

Lloydstraat 138c
3024 EA Rotterdam
t +31 10 221 01 90

kvk 24300171
btw NL8186.35.629.B.01

info@hoppinger.com
www.hoppinger.com

hoppinger▲

internetconcepten
websites
applicaties

het hoppinger▲ html5 handboek

Auteur: Tim Huisman
Editor: Pieter Beekman

Versie: 1.0



Inhoudsopgave

Inleiding	5
Opmaak.....	5
Bronverwijzingen	6
Aanleiding	7
Bronnen.....	7
Hoofdstuk 1: Wat is de essentie van HTML5 en wat wil men ermee bereiken?	8
1.1 - Van 'HTML Tags' tot XHTML 1.1.....	8
1.1.1 - HTML.....	8
1.1.2 - XHTML	8
1.2 - En toen was er HTML5... ..	9
1.2.1 - Workshop on Web Applications and Compound Documents	9
1.2.2 - Web Hypertext Application Technology Working Group (WHATWG).....	10
1.3 - Conclusie.....	10
Bronnen	11
Hoofdstuk 2: Wat zijn de nieuwe mogelijkheden van HTML5 in vergelijking met XHTML?	12
2.1 - Doctype, <html> en <head> element.....	12
2.1.1 - Doctype	12
2.1.2 - <html> element.....	12
2.1.3 - <head> element.....	13
2.2 - Semantische elementen	13
2.2.1 - <header> element	14
2.2.2 - <nav> element.....	15
2.2.3 - <section> element	15
2.2.4 - <article> element.....	16
2.2.5 - <aside> element	17
2.2.6 - <footer> element	17
2.2.7 - <hgroup> element	18
2.2.8 - <time> element.....	19
2.2.9 - <mark> element.....	19
2.2.10 - <figure> element.....	20
2.3 – Forms	21
2.3.1 - Input types.....	21
2.3.2 - Attributen	24
2.3.3 - Elementen	26
2.3.4 - Input validatie	27
2.4 - Video en Audio	28
2.4.1 - Video	28
2.4.2 - Audio	29
2.5 - Canvas	29



2.6 - Microdata.....	30
2.7 - Web Storage en Offline Storage.....	31
2.7.1 - Web Storage.....	31
2.7.2 - Offline Storage.....	32
2.8 - Drag and Drop.....	32
2.9 - Geolocation	33
2.10 - Conclusie.....	34
2.10.1 - Voordelen	34
Bronnen	35
Hoofdstuk 3 : Ligt het succes van HTML5 bij het aanpassingsvermogen van de grote browsers?.....	37
3.1 - Semantische elementen	37
3.1.1 - HTML5 Shiv.....	37
3.1.2 - Display: Block.....	38
3.2 - Forms	38
3.3 - Video, audio en canvas(text)	39
3.4 - Microdata.....	40
3.5 - Web Storage, Offline Storage, Drag and Drop en Geolocation	40
3.6 - Conclusie.....	41
Bronnen	41
Hoofdstuk 4: Zal HTML5 voordelen opleveren op het gebied van SEO in vergelijking met XHTML?	42
4.1 - Page segmentation.....	42
4.2 - Zoekmachines beïnvloeden.....	43
4.2.1 - <time> element.....	43
4.2.2 - Microdata.....	43
4.3 - Zoekmachines en video's	44
4.4 - Conclusie.....	44
Bronnen	45
Hoofdstuk 5: Is HTML5 een goed alternatief om Flash mee te vervangen?.....	46
5.1 - Browser plug-ins.....	46
5.1.1 - Adobe Flash	46
5.1.2 - Alternatieven.....	47
5.2 - Prestaties	47
5.2.1 - CPU gebruik	47
5.2.2 - Frames per second.....	48
5.3 - Search Engine Optimization (Flash vs. Canvas)	50
5.4 - Conclusie.....	51
Bronnen	52
Hoofdstuk 6: Nu overstappen op HTML5 of toch nog even wachten?	53
6.1 - Backwards compatibility	53



6.2 - Browser compatibility.....	55
6.3 - Conclusie.....	55
Bronnen.....	56
Hoofdstuk 7: Wat heeft HTML5 te bieden?	57
7.1 - Wat is de essentie van HTML5 en wat wil men ermee bereiken?	57
7.1.1 - Samenvatting.....	57
7.2 - Wat zijn de nieuwe mogelijkheden van HTML5 in vergelijking met XHTML?	57
7.2.1 - Samenvatting.....	57
7.2.2 - Conclusie.....	58
7.3 - Ligt het succes van HTML5 bij het aanpassingsvermogen van de grote browsers?	59
7.3.1 - Samenvatting.....	59
7.3.2 - Conclusie.....	60
7.4 - Zal HTML5 voordelen opleveren op het gebied van SEO in vergelijking met XHTML?	60
7.4.1 - Samenvatting.....	60
7.4.2 - Conclusie.....	61
7.5 - Is HTML5 een goed alternatief om Flash mee te vervangen?	61
7.5.1 - Samenvatting.....	61
7.5.2 - Conclusie.....	61
7.6 - Nu overstappen op HTML5 of toch nog even wachten?.....	62
7.6.1 - Samenvatting.....	62
7.6.2 - Conclusie.....	62
Bijlagen.....	63
B1 - Compatibiliteit browsers.....	64
B1.1 - Opera 10.62.....	64
B1.2 - Chrome 6.0.472.63.....	65
B1.3 - Safari 4.0.5 (531.22.7)	66
B1.4 - Firefox 3.6.10	67
B1.5 - Internet Explorer 9.....	68
B1.6 - Internet Explorer 8.0.6001.....	69
B1.7 - iPhone Browser & iPad Browser	70
B1.8 - Android Browser.....	71
B1.9 – Opera Mini.....	72
B2 - XHTML2.....	73
B2.1 - Design Aims.....	73
B2.2 - Ontwikkeling XHTML2 stopgezet	74
B2.3 - Veranderingen in XHTML2.....	74
Bronnen.....	76
B3 - 'Thoughts on Flash'.....	77
Bronnen.....	79
Bronnenlijst.....	80



Inleiding

Dit onderzoek richt zich op de nieuwe webstandaard HTML5, dat momenteel in ontwikkeling is door het *Web Hypertext Application Technology Working Group (WHATWG)*. HTML5 zal naar alle waarschijnlijkheid de nummer één markup language worden en hiermee deze rol overnemen van HTML4 en XHTML1. We zullen in dit onderzoek gaan kijken naar de gedachtegang achter HTML5 en de nieuwe en verbeterde functionaliteiten. Om dit zo goed mogelijk te belichten zal dit gedaan worden vanuit een drietal perspectieven: eindgebruiker, ontwikkelaar en *Hoppering*.

- Eindgebruiker: hiermee worden de bezoekers van websites en applicaties bedoeld. Deze mensen hebben/hoeven geen kennis te hebben van de achterliggende code. Vanuit dit perspectief wordt er vooral gekeken naar de visuele en gebruiksvriendelijke mogelijkheden van HTML5.
- Ontwikkelaar: hiermee worden de bouwers van websites en applicaties bedoeld. Deze mensen hebben/willen juist kennis van de achterliggende code opdoen. Daarnaast willen zij de voordelen van HTML5 weten in vergelijking met HTML4 en/of XHTML1.
- *Hoppering*: spreekt redelijk voor zich, maar waar wordt vooral naar gekeken vanuit dit perspectief? *Hoppering* wil graag weten wanneer ze moeten overstappen naar HTML5, wat HTML5 te bieden heeft en welke voordelen er zijn voor hun klanten.

Aan het eind van het onderzoek zal er antwoord gegeven worden op de hoofdvraag: **“Wat heeft HTML5 te bieden?”** Om dit zo volledig en onderbouwd mogelijk te kunnen doen zal er eerst antwoord gegeven worden op een achttal deelvragen. Elke deelvraag verdiept zich in een bepaald gebied van HTML5, namelijk: achtergronden, vernieuwingen, compatibiliteit, semantiek en SEO, HTML5 vs. Flash en toekomstbeeld.

- Achtergronden: **“Deelvraag #1 - Wat is de essentie van HTML5 en wat wil men ermee bereiken?”**
- Vernieuwingen: **“Deelvraag #2 - Wat zijn de nieuwe mogelijkheden van HTML5 in vergelijking met XHTML?”**
- Compatibiliteit: **“Deelvraag #3 - Ligt het succes van HTML5 bij het aanpassingsvermogen van de grote browsers?”**
- Semantiek en SEO: **“Deelvraag #4 - Zal HTML5 voordelen opleveren op het gebied van SEO in vergelijking met XHTML?”**
- HTML5 vs. Flash: **“Deelvraag #5 - Is HTML5 een goed alternatief om Flash mee te vervangen?”**
- Toekomstbeeld: **“Deelvraag #6 - Nu overstappen op HTML5 of toch nog even wachten?”**

Opmaak

Om extra aandacht te vestigen op stukken tekst, bronnen of stukken code zijn deze opvallend opgemaakt. Hieronder een overzicht:

- Hierin staan codevoorbeelden in XHTML
- Hierin staan codevoorbeelden in HTML5
- *“Hierin staan officiële WHATWG HTML5 specificaties”*
- Hierin staan afbeeldingen
- **“Bronnen die in de lopende tekst staan worden op deze manier weergegeven, zodat ze meer opvallen.”**
- *Namen van personen, bedrijven, browsers, technieken en literatuur worden cursief weergegeven. Literaire werken staan ook nog eens tussen ‘ ’.*
- *Elementen, attributen en functies worden weergegeven met een ander lettertype: Myriad Pro*



Bronverwijzingen

In dit onderzoek worden een groot aantal externe bronnen gebruikt. Om plagiaat te voorkomen is het belangrijk dat de herkomst van de bron wordt gedocumenteerd. Dit vind achteraan in het hoofdstuk plaats (indien er bronnen zijn gebruikt). Om de leesbaarheid van deze verwijzingen te vergroten volgt hier een uitleg.

Verwijzingen naar bronnen uit boekwerken:

Template: #Nummer# : #Titel# | #Hoofdstuknummer# : #Hoofdstuk# – #Kopje# (Wanneer aanwezig) | #Bladzijde(n)# | #Auteur# | #Druk# | #Drukkerij# | #Datum dat bron is geraadpleegd#

Voorbeeld: ¹ : HTML5: Up and Running | Hfst. 2 : Detecting HTML5 Features – Placeholder Text | Blz. 27 | Mark Pilgrim | August 2010 – First Edition | O'Reilly Media, Inc. | 20-09-2010

Verwijzingen naar bronnen van het internet:

Template: #Nummer# : #Titel# | #URL van bron# | #Kopje(s)# (Wanneer aanwezig) | #Auteur# (Wanneer bekend) | #Datum van publicatie# (Wanneer bekend) | #Datum dat bron is geraadpleegd#

Voorbeeld: ¹ : A Brief History of Markup | <http://www.alistapart.com/articles/a-brief-history-of-markup/> | XHTML 1: HTML as XML | Jeremy Keith | 04-05-2010 | 21-09-2010



Aanleiding

Terwijl *HTML5* nog steeds in volle ontwikkeling is zijn er op dit moment al een groot aantal nieuwe functionaliteiten beschikbaar voor gebruik. Denk bijvoorbeeld aan de nieuwe semantische elementen (o.a.: <section>, <article>, <header>, <footer>), canvas, video, audio en uitbreidingen voor forms. Het is dan ook niet vreemd dat showcases, handleidingen, artikelen en websites volledig gemaakt met *HTML5* als paddenstoelen uit de grond schieten. Met de huidige ontwikkelingen rond *HTML5* lijkt het erop dat *HTML5* blijft. De kans dat het de nieuwe webstandaard zal worden is groot, maar wanneer is de vraag...

Wanneer moeten webbureaus hun producten met *HTML5* gaan bouwen en/of oude producten ombouwen? Er is namelijk nog geen enkele browser die alle functionaliteiten van *HTML5* ondersteund.¹ En de specificaties zoals ze nu staan geschreven kunnen nog makkelijk veranderen. Dit wordt ook nadrukkelijk aangegeven door het *Web Hypertext Application Technology Working Group (WHATWG)*: *"This is a work in progress! This document is changing on a daily if not hourly basis in response to comments and as a general part of its development process."*² Het *WHATWG* is een samenwerkingsverband tussen browserontwikkelaars en webtechnologie geïnteresseerden, die de ontwikkeling van *HTML5* op hun schouders hebben genomen. Naar aanleiding van bovenstaande quote kan men zich afvragen of het eigenlijk al wel zin heeft om als webbureau volledig over te stappen naar *HTML5*? Of is het beter om nog even te wachten tot de ondersteuning verbeterd is?

In deze 'wanneer moeten we overstappen'-vraag is *Hopperinger* ook geïnteresseerd en dit is aanleiding geweest om een onderzoek naar *HTML5* op poten te zetten. Via dit onderzoek wil *Hopperinger* antwoord krijgen op de 'wanneer'-vraag en een beeld krijgen van de mogelijkheden van *HTML5*. De onderzoeker wil zelf doormiddel van dit onderzoek een beeld krijgen van de functionaliteiten en werking van *HTML5* en de toekomst van (front-end) web development.

Bronnen

¹: *HTML5 & CSS3 Support* | <http://www.findmebyip.com/litmus#target-selector> | *HTML5 Web Applications t/m HTML5 Forms Attributes* | | 20-09-2010

²: *HTML5 (including next generation additions still in development) Draft Standard* | <http://www.whatwg.org/specs/web-apps/current-work/multipage/> | *Status of this document* | | 10-09-2010 | 20-09-2010



Hoofdstuk 1: Wat is de essentie van HTML5 en wat wil men ermee bereiken?

Voordat we kijken naar het heden is het goed om eerst terug te gaan in het verleden. Hierin ligt namelijk één van de hoofdredenen voor de ontwikkeling van een nieuwe HTML versie.

1.1 - Van 'HTML Tags' tot XHTML 1.1

1.1.1 - HTML

De term *HyperText Markup Language* dook voor het eerst op in 1991 toen *Tim Berners-Lee* een document publiceerde genaamd 'HTML Tags'. Hierin stonden een twintigtal elementen beschreven, die de basis zouden vormen voor het eerste concept van HTML. Deze zag in 1993 het daglicht toen *Tim Berners-Lee* samen met en *Daniel Connolly* de 'Hypertext Markup Language (HTML) Internet Draft'¹ opstelde en publiceerde. In hetzelfde jaar kwam *Dave Raggett* met een eigen HTML concept onder de naam HTML+. Beide concepten bleven echter liggen en waren na een aantal maanden verlopen, maar de toon was gezet. In 1995 werd door de *Internet Engineering Task Force (IETF)* het eerste ontwerp van HTML voltooid onder de naam HTML 2.0.

Een jaar later, in 1996, hief IETF zijn HTML werkgroep op, maar onder leiding van *Tim Berners-Lee* nam de *World Wide Web Consortium (W3C)* hun werkzaamheden over. HTML 3.0 kwam niet verder dan de conceptfase, maar begin 1997 was er dan toch de opvolger voor HTML 2.0 in de vorm van versie 3.2. Aan het eind van datzelfde jaar werd HTML 4.0 op de markt gebracht en verscheen er nog een update naar versie 4.01 in 1999. Vanaf dat moment kwam de ontwikkeling van HTML tot een halt.

1.1.2 - XHTML

De reden voor het stopleggen van de ontwikkeling van HTML kwam door de ontevredenheid van de W3C met de markup language: ontwikkelaars hadden naar hun mening teveel vrijheid in de opmaak van hun documenten. Daarom ontwikkelde ze XHTML, dat begin 2000 werd doorgevoerd: *"The content of the XHTML 1.0 specification was identical to that of HTML 4.01. No new elements or attributes were added. The only difference was in the syntax of the language. [...] XHTML required authors to follow the rules of XML, a stricter markup language [...]"*² Er werden geen nieuwe functionaliteiten toegevoegd, maar wel strikte regels om zo meer uniformiteit te krijgen: *"It encouraged authors to use a single writing style. Whereas previously tags and attributes could be written in uppercase, lowercase, or any combination thereof, a valid XHTML 1.0 document required all tags and attributes to be lowercase."*²

In mei 2001 kwam W3C met de opvolger van XHTML 1.0, onder de naam XHTML 1.1. En met deze versie kwam het probleem van XHTML bovendien: *"Use something that looks kind of like XHTML syntax, but keep serving it with the text/html MIME type. [...] Unless you're serving your pages with a MIME type of application/xhtml+xml, your so-called "XHTML" is XML in name only."*³ Het komt erop neer dat ontwikkelaars de syntax van XHTML gebruikten, maar het document wel een HTML MIME type meegaven. Is dit erg? Vergelijk het met een potje basketbal (= XHTML), maar met de regels van voetbal (= HTML). Overtredingen worden vanuit het voetbal perspectief bekeken en bestraft. Dit betekent in het kort dat je strikt genomen niet aan het basketballen bent.

Maar waarom wordt er dan niet op goede manier van XHTML gebruik gemaakt? De MIME type veranderen is namelijk zo gedaan. Het probleem zit hem in het feit dat XHTML met zogeheten draconion error handling werkt: *"If there was even a single error in your XHTML page, web browsers would have no choice but to stop processing and display an error message to the end user."*³ Eén klein foutje en de website zou niet meer weergegeven worden. Samen met de wetenschap dat er 99% kans is dat bestaande webpagina's een fout bevatten³, hadden ontwikkelaars snel de keuze gemaakt. Met de text/html MIME type werd de webpagina goed weergegeven, terwijl het veranderen naar de application/xhtml+xml MIME type hoogstwaarschijnlijk alleen errors zou opleveren.



Er zijn twee belangrijke vragen die naar aanleiding hiervan gesteld kunnen worden.

1. **Wil je als bedrijf extra tijd, mankracht en geld inzetten om de fouten op te lossen of wil je twee woorden veranderen waardoor de fouten vanzelf 'verdwijnen'?**
Het antwoord op deze vraag ligt aan wat de ontwikkelaar/het bedrijf belangrijker vindt. Als de voorkeur ligt bij een website waarvan de code volledig valide is, zullen projecten automatisch langer gaan duren. We hebben immers kunnen lezen dat (zelfs de kleinste) fouten in documenten met het application/xhtml+xml MIME type onverbiddeijk worden afgestraft en deze moeten allemaal weggewerkt worden. Als de voorkeur ligt bij het maken van een website die goed wordt weergegeven, met echter enkele oneffenheden in de code bied het veranderen van de MIME type naar text/html MIME type de uitkomst. Het voordeel is dat projecten minder lang duren in vergelijking met de eerste optie. Hoe sneller het product wordt opgeleverd, hoe sneller men met een nieuw product kan beginnen.
2. **Is het wel zinvol om XHTML te gebruiken als je het technisch gezien niet goed gebruikt?**
De reden waarvoor XHTML is ontwikkelt (striktere syntax voor meer uniformiteit) is over het algemeen goed ontvangen bij ontwikkelaars. Er zullen nog maar weinig webbedrijven zijn die hun websites met de HTML syntax schrijven. Het zou haast de omgekeerde wereld zijn om nu terug over te stappen naar een syntax, die de uniformiteit op het web meer kwaad dan goed kan doen. Daarom is en blijft het gebruik van XHTML zinvol en om deze reden is HTML5 tegelijk ook XHTML5. De standaarden uit XHTML zijn doorgevoerd in HTML5, maar de ontwikkelaar is niet verplicht om deze te gebruiken.

1.2 - En toen was er HTML5...

De titel doet vermoeden dat er in het diepste geheim ergens onder een steen jaren aan HTML5 is gewerkt, en dat men als een donderslag bij heldere hemel van onder de steen sprong. Maar niets is minder waar: HTML5 werd ruim zes jaar geleden voor het eerst genoemd en men is waarschijnlijk nog jaren bezig met de ontwikkeling.

1.2.1 - Workshop on Web Applications and Compound Documents

In 2004 organiseerde de W3C een workshop op het gebied van *Web Applications and Compound Documents*.⁴ Van deze mogelijkheid maakte de *Mozilla Foundation* en *Opera Software* gebruik om hun gezamenlijke visie te geven over de toekomst van het web. Zij kwamen met een zevental vereisten voor moderne web applicaties:⁵

- Web applicaties moeten gemaakt kunnen worden met technieken, die bekend zijn bij ontwikkelaars (*HTML*, *CSS*, *DOM* en *JavaScript*). Het gebruik van externe plugins om de applicaties te laten werken moet voorkomen worden.
- Error afhandeling moet gedetailleerd worden gedefinieerd om te voorkomen dat browsers zelf mechanismen moeten bedenken.
- Gebruikers van web applicaties mogen niks merken van fouten, die zijn gemaakt door ontwikkelaars.
- Alle functies (die opgenomen worden in de web applicaties specificaties) moeten een praktisch en functioneel doel dienen.
- Web applicaties moeten op elk apparaat en in elke vorm van presentatie identiek zijn.
- Web applicaties moeten dezelfde functionaliteiten hebben in zowel de desktop als mobiele versie.
- Het ontwikkeltraject van web applicaties moet open zijn; iedereen moet toegang hebben tot de mailinglijsten, archieven en specificaties.

Tijdens een stemronde bleek echter dat men niet volledig achter deze ideeën stond; elf stemmen tegen en acht voor. In de samenvatting van de workshop stond de volgende uitleg: ***“At present, W3C does not intend to put any resources into the third straw-poll topic: extensions to HTML and CSS for Web Applications, other than technologies being developed under the charter of current W3C Working Groups.”***⁶ De W3C koos ervoor om toentertijd geen enkele ondersteuning te bieden voor hetgeen dat nu onder de noemer HTML5 valt. Zij zagen meer brood in versie 2.0 van XHTML, maar daarover is meer te lezen in bijlage twee. De partijen die de presentatie hadden gegeven verenigde zich en zo ontstond de *Web Hypertext Application Technology Working Group (WHATWG)*.



1.2.2 - Web Hypertext Application Technology Working Group (WHATWG)

Maar wat wil de WHATWG bereiken? Hier wijden zij zelf over uit in een nieuwsbericht met betrekking tot hun plannen voor de toekomst, geschreven in 2005: *“The group aims to develop specifications based on HTML and related technologies to ease the deployment of interoperable Web Application [...]The working group intends to ensure that all its specifications address backwards compatibility concerns, clearly provide reasonable transition strategies for authors, and specify error handling behavior to ensure interoperability even in the face of documents that do not comply to the letter of the specifications.”*⁷

In hetzelfde nieuwsbericht wordt aangegeven dat de WHATWG in ieder geval aan drie documenten gaat werken: *Web Forms 2.0* (Uitbreiding van HTML4 formulieren), *Web Apps 1.0* (Functionaliteiten voor applicatie ontwikkeling) en *Web Controls 1.0* (Mechanismen voor ontwikkeling interactieve widgets). Sinds de eerste specificatielijst in 2005, bekend onder de naam ‘*Web Applications 1.0 Early Working Draft*’⁸, zijn de ideeën voor *Web Forms 2.0* en *Web Apps 1.0* samengevoegd tot één document en hernoemd tot *HTML5*. Op het moment van schrijven is de laatste specificatielijst van HTML5 gepubliceerd op 10 september 2010.⁹

Waar nogal discussie en verwarring over is ontstaan is de datum waarop *HTML5* officieel klaar zal zijn. *Ian Hickson* heeft namelijk in een interview gezegd dat *HTML5* in 2022 de proposed recommendation zal worden.¹⁰ Veel geïnteresseerden in het onderwerp concludeerden dat het geen nut zou hebben om te verdiepen in *HTML5*. Aangezien er rond dat tijdstip andere technieken of nieuwere versies ten tonele zouden zijn verschenen. Deze conclusie was echter te voorbarig. Met een proposed recommendation wordt namelijk het volgende bedoeld: de techniek is in zijn volledigheid geïmplementeerd in twee verschillende browsers. De belangrijkste fase van ontwikkeling voor webontwikkelaars is de candidate recommendation. Dit komt in het kort neer op de versie die de WHATWG beschouwt als de voltooide versie. Deze kandidaat versie zou in 2012 klaar zijn.

De specificaties van *Cascading Style Sheets 2.1 (CSS2.1)* zit in dezelfde situatie als *HTML5*. Vanaf 2007 zit deze revisie op *CSS2* in de candidate recommendation fase, terwijl het al wel grootschalig wordt gebruikt. Men verwacht dat de status van de specificaties in 2011 zal worden omgezet naar de proposed recommendation en in hetzelfde jaar naar de definitieve recommendation. Hetzelfde zal gebeuren met *HTML5*: ontwikkelaars werken er al mee terwijl in 2022 de specificaties pas officieel klaar zullen zijn.

1.3 - Conclusie

HTML5 is een verhaal van het verleden, heden en toekomst. Zo wordt er gekeken naar het verleden om hieruit lering te trekken, wil men het heden vergemakkelijken en voorbereid zijn op de toekomst.

Elementen uit het verleden zijn voornamelijk van toepassing op de error afhandeling: *HTML* was hier te los in en *XHTML* bleek uiteindelijk te strikt. Daarom is één van de kernpunten om de error afhandeling zo gedetailleerd mogelijk te documenteren om zo de gouden tussenweg te creëren. Fouten van ontwikkelaars zullen worden opgevangen terwijl de eindgebruiker niets zal merken van de fout.

De *WHATWG* wil het heden vergemakkelijken door uitbreidingen te bedenken voor bestaande functionaliteiten en daarmee externe plugins en/of uitgebreide scripts onnodig maken. Een voorbeeld hiervan is de validatie van formulier inhoud; hier moesten tot nu toe zelf stukken code in *JavaScript* voor geschreven worden. Nu voldoet het aangeven van het juiste input type en de browser valideert de inhoud automatisch. Meer hierover in het volgende hoofdstuk.

De wijze waarop het internet tegenwoordig wordt gebruikt is aan het veranderen. Vroeger werd het voornamelijk gebruikt om tekstdocumenten te delen, maar nu wil men ook video, audio en games kunnen zien, horen en spelen. Het web is interactiever geworden en daar zijn volgens de *WHATWG* de huidige generatie markup languages niet voor gebouwd. De `<video>`, `<audio>` en `<canvas>` functionaliteiten zijn enkele voorbeelden van hoe men dit wil gaan bereiken.



Maar de gedachten waar de WHATWG met elke specificatie weer op terugkomt zijn als volgt: specificaties moeten volledig achterwaarts compatibel met zijn voorgangers zijn, functionaliteiten moeten op elk platform en in elke browsers werken en alle functies moeten een praktisch en functioneel doel dienen. Het ander doel van de WHATWG is om het mogelijk te maken om web applicaties te bouwen met bekende technieken (*HTML*, *CSS*, *DOM* en *JavaScript*), zonder dat hiervoor externe plugins gebruikt hoeven te worden.

Bronnen

¹ : Hypertext Markup Language (HTML) Internet Draft | <http://www.w3.org/MarkUp/draft-ietf-iiir-html-01.txt> | | Tim Berners-Lee & Daniel Connolly | ??-06-1993 | 21-09-2010

² : A Brief History of Markup | <http://www.alistapart.com/articles/a-brief-history-of-markup/> | XHTML 1: HTML as XML | Jeremy Keith | 04-05-2010 | 21-09-2010

³ : HTML5: Up and Running | Hfst. 1 : How Did We Get Here? – Everything You Know About XHTML Is Wrong | Blz. 10 | Mark Pilgrim | August 2010 – First Edition | O'Reilly Media, Inc. | 21-09-2010

⁴ : The W3C Workshop on Web Applications and Compound Documents | <http://www.w3.org/2004/04/webapps-cdf-ws/> | | Dean Jackson | 13-01-2005 | 21-09-2010

⁵ : Position Paper for the W3C Workshop on Web Applications and Compound Documents | <http://www.w3.org/2004/04/webapps-cdf-ws/papers/opera.html> | Design Principles for Web Application Technologies | The Mozilla Foundation & Opera Software, ASA | | 21-09-2010

⁶ : Summary of The W3C Workshop on Web Applications and Compound Documents | <http://www.w3.org/2004/04/webapps-cdf-ws/summary> | Next steps | Dean Jackson | 21-07-2004 | 21-09-2010

⁷ : WHAT open mailing list announcement | <http://www.whatwg.org/news/start> | | Ian Hickson | 04-05- 2004 | 21-09-2010

⁸ : Web Applications 1.0 Early Working Draft | <http://www.whatwg.org/specs/web-apps/2005-09-01/> | | Ian Hickson (Editor) | 01-09-2005 | 21-09-2009

⁹ : HTML5 (including next generation additions still in development) Draft Standard | <http://www.whatwg.org/specs/web-apps/current-work/> | | Ian Hickson (Editor) | 10-09-2010 | 21-09-2010

¹⁰ : HTML 5 Editor Ian Hickson discusses features, pain points, adoption rate, and more | <http://blogs.techrepublic.com.com/programming-and-development/?p=718> | | Justin James | 27-08-2008 | 21-09-2010



Hoofdstuk 2: Wat zijn de nieuwe mogelijkheden van HTML5 in vergelijking met XHTML?

Dit hoofdstuk verdiept zich in de vernieuwingen en veranderingen, die met *HTML5* worden geïntroduceerd. Aangezien *Hopper* zijn websites in *XHTML* maakt, zal er met deze markup language worden vergeleken. Voorbeelden zijn allemaal gebaseerd op de indexpagina van de *Hopper* website, tenzij anders aangegeven. Elementen en attributen die niets bijdragen aan het voorbeeld worden weggelaten. Onderstreepte elementen in de codevoorbeelden worden later in dit hoofdstuk toegelicht.

De volgende functionaliteiten zullen in dit hoofdstuk besproken worden:

- Doctype, <html> en <head> element
- Semantische elementen
- Forms
- Video en Audio
- Canvas
- Microdata
- Web Storage en Offline Storage
- Drag and Drop
- Geolocation

2.1 - Doctype, <html> en <head> element

2.1.1 - Doctype

Elk *HTML* document begint met een doctype en daarom zal deze als eerste worden toegelicht. De gedachtegang van *Web Hypertext Application Technology Working Group (WHATWG)* om het web ontwikkelaars makkelijk te maken is hier duidelijk doorgedrongen. *XHTML* had een doctype dat onmogelijk was om te onthouden. Er moesten templates of CTRL-C/CTRL-V aan te pas komen om de deze in elk document hetzelfde te hebben. Voorbeeld:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Zoals in het vorige hoofdstuk is geconcludeerd wil de *WHATWG* met elke specificatie volledig achterwaartse compatibiliteit met voorgaande *HTML* versies. En dit is duidelijk te zien in de nieuwe doctype: alle versie specifieke informatie is weggelaten. Het is alsof alle onderscheid is verdwenen en dat er nog maar één overkoepelende variant over is gebleven:

```
<!DOCTYPE html>
```

2.1.2 - <html> element

Een zelfde soort verandering heeft plaatsgevonden met het root element. Deze is aanzienlijk ingekort om *XHTML* specifieke attributen weg te laten. Het enige attribuut dat is overgebleven is het lang attribuut. Hiermee wordt aangegeven in welke taal de inhoud van het document is geschreven. Browsers kunnen de waarde bijvoorbeeld gebruiken voor correcte spellingscontrole of het activeren van het juiste taalpakket van de screenreader.

Voorbeeld:

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="nl" lang="nl">
```

```
<html lang="nl">
```



2.1.3 - <head> element

2.1.3.1 - <meta> element

Aan het <head> element zelf is weinig te veranderen, het gaat in dit geval meer om de inhoud. Zo heeft het <meta> element een metamorfose ondergaan. Sinds *HTML5* is het vereist om minimaal één van de volgende attributen op te nemen: name, http-equiv, charset, en itemprop attributes. Dit ligt er uiteraard aan in hoeverre het <meta> element is uitgewerkt (op de *Hoppinger* site staan er namelijk zes). Maar in de simpelste vorm zou het er als volgt uit kunnen zien:

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
```

```
<meta charset="UTF-8" />
```

2.1.3.2 - <link> element & <script> element

In het <html> element staan ook referenties naar externe en interne bestanden zoals *CSS* en *JavaScript* bestanden. Het is in *HTML5* niet meer nodig om hier een type aan toe te voegen. Het is in de meeste gevallen duidelijk om welk soort bestanden het gaat. Voorbeeld:

```
<link rel="stylesheet" type="text/css" href="/stylesheets/sifr.css" />  
<script type="text/JavaScript" src="/JavaScript/menu.js"></script>
```

```
<link rel="stylesheet" href="/stylesheets/sifr.css" />  
<script src="/JavaScript/menu.js"></script>
```

2.2 - Semantische elementen

*"In essence, someone queried a few Google indexes for common web developer practices, and decided to invent new tags for what people were already doing."*¹ De auteur van deze bron beweert dat de elementen, die in dit hoofdstuk zullen worden besproken, zijn toegevoegd omdat deze veel worden gebruikt. Het geeft figuurlijk alleen maar een andere naam aan hetzelfde voorwerp. Maar zo eenvoudig als de bron het doet voorkomen is het niet. Het vervangen van <div> elementen levert ten eerste meer structuur en uniformiteit op. Ten tweede is het veel gunstiger voor machines (in dit geval browsers): het wordt gemakkelijker om de informatie te lezen en te verwerken.

In dit onderdeel worden de volgende nieuwe (semantische) elementen besproken:

- <header>
- <nav>
- <section>
- <article>
- <aside>
- <footer>
- <hgroup>
- <time>
- <mark>
- <figure>

Van sommige elementen uit de lijst kan direct bedacht worden waarvoor ze zouden kunnen dienen. En dat is precies wat men met de elementen wil bereiken: zowel mens als machine moet de context van de inhoud kunnen ontdekken doormiddel van de tags waar het tussen staat. Zoekmachines weten bijvoorbeeld dat de belangrijke informatie nu hoogstwaarschijnlijk in een <article> element zal staan. Voorheen kon elke <div> de informatie bevatten en werden daarom allemaal als gelijkwaardig gezien. Het was zoeken naar een naald in een hooiberg.



2.2.1 - <header> element

*"The header element represents a group of introductory or navigational aids. Note: A header element is intended to usually contain the section's heading (an h1–h6 element or an hgroup element), but this is not required. The header element can also be used to wrap a section's table of contents, a search form, or any relevant logos."*³

Het probleem met de huidige manier om een header aan te geven (bijvoorbeeld; `<div id="header">`), is dat het semantisch geen betekenis heeft. Tussen een `<div>` kan elk soort informatie worden geplaatst en de browsers mogen geen betekenis uit de id afleiden.⁴ Daarom is er in *HTML5* het `<header>` element opgenomen. Qua inhoud verandert er weinig, maar het is vooral gunstig voor de algehele structuur en voor Search Engine Optimization (SEO). Meer over dit laatste onderwerp in hoofdstuk vier. Voorbeeld:

```
<div id="topContainer">
  <ul>
    <li>home</li> <li>contact</li>
  </ul>
  <a href="/"></a>
</div>
```

```
<header>
  <ul>
    <li>home</li> <li>contact</li>
  </ul>
  <a href="/"></a>
< /header >
```

Maar het `<header>` element hoeft niet alleen gebruikt te worden voor de header van de gehele webpagina, maar kan bijvoorbeeld ook gebruikt worden om de titel en/of datum van nieuwsartikelen in te plaatsen (zie codevoorbeelden hieronder). Machines hoeven enkel naar het parent element (het element waar het `<header>` element in wordt gebruikt) te kijken om te bepalen waar de header op van toepassing is. Een header binnen een `<article>` element zal hoogstwaarschijnlijk niet de header van de hele webpagina zijn, maar bijvoorbeeld van een nieuwsbericht. De kans dat de header binnen een `<section>` element de header van de hele pagina is, is aanzienlijk groter.

```
<div class="block_content">
  <h4>Adwords workshop op 21 september</h4>
  <span class="blockDate">17 augustus 2010</span>
  ...
</div>
```

```
<article>
  <header>
    <h4> Adwords workshop op 21 september</h4>
    <time datetime="2010-08-17" pubdate="pubdate">17 augustus 2010</time>
  </header>
  ...
</ article >
```



2.2.2 - <nav> element

*"The nav element represents a section of a page that links to other pages or to parts within the page: a section with navigation links. Not all groups of links on a page need to be in a nav element — only sections that consist of major navigation blocks are appropriate for the nav element."*⁵

Net als in het geval van het <header> element heeft de oude variant van een navigatiemenu (bijvoorbeeld; <div id="navigation">) geen semantische waarde. De ontwikkelaar weet uiteraard de functie van deze <div>, maar de browsers kunnen dit er niet uithalen. Deze zien alleen een lijst met hyperlinks. Bovendien is de gebruiksvriendelijkheid voor visueel en/of motoriek gehandicapten laag. Er bestaan plugins en screenreaders die het toelaten om direct naar de navigatie te springen of juist over te slaan.⁶ Als de id en/of class naam van de <div> op elke website verschilt wordt het functioneren van deze hulpmiddelen bemoeilijkt. Om dit op te lossen is het in *HTML5* het <nav> element geïntroduceerd. Voorbeeld:

```
<div class="menu">
  <ul>
    <li>over hoppers</li>
    ...
    <li>weblog</li>
  </ul>
</div>
```

```
<nav>
  <ul>
    <li>over hoppers</li>
    ...
    <li>weblog</li>
  </ul>
</ nav >
```

Het visuele verschil is klein, maar qua semantiek en gebruiksvriendelijkheid maakt het een groot verschil. Browsers weten dat in het <nav> element de belangrijkste links naar andere webpagina's/websites staat. Ook hulpmiddelen voor gehandicapten hebben er baat bij; deze kunnen ervan uitgaan dat het hoofdmenu binnen dit blok staat. Het is trouwens mogelijk om meerdere <nav> blokken op één pagina te plaatsen. En het is niet noodzakelijk om dit alleen in de header te doen, zelfs de footer kan een <nav> element bevatten.

2.2.3 - <section> element

*"The section element represents a generic section of a document or application. A section, in this context, is a thematic grouping of content, typically with a heading. Note: The section element is not a generic container element. When an element is needed for styling purposes or as a convenience for scripting, authors are encouraged to use the div element instead. A general rule is that the section element is appropriate only if the element's contents would be listed explicitly in the document's outline."*⁷

Het <section> element wordt gebruikt om gerelateerde stukken informatie te groeperen. Het is te vergelijken met een <div>, maar in dit geval weet de browser dat de inhoud van de <section> bij elkaar hoort. Zo kan een <section> bestaan uit een reeks nieuwsberichten of juist de reacties op een nieuwsbericht. Voorbeeld:

```
<div class="wideContentBlock">
  <div class="block_content">
    ...
  </div>
  <div class="block_content">
    ...
  </div>
</div>
```



Het codevoorbeeld in *XHTML*, op de vorige pagina, zou in HTML5 er als volgt uit kunnen zien:

```
<section>
  <article>
    ...
  </article>
  <article>
    ...
  </article>
</section>
```

2.2.4 - <article> element

“The article element represents a self-contained composition in a document, page, application, or site and that is, in principle, independently distributable or reusable, e.g. in syndication. This could be a forum post, a magazine or newspaper article, a blog entry, a user-submitted comment, an interactive widget or gadget, or any other independent item of content.”⁸

Tot nu toe werden stukken informatie in een <div> element geplaatst, maar welke informatie hierin stond kon de browser niet detecteren. Het kan de navigatie zijn, maar net zo goed de header of een nieuwsbericht? De eerste twee onderdelen zijn besproken (<nav> en <header>), maar hoe zit het met de laatste? Hiervoor is er in *HTML5* het <article> element toegevoegd. Zoals bovenstaande quote vermeld wordt er binnen een <article> stukken informatie geplaatst die niet afhankelijk zijn van anderen. Voorbeeld:

```
<div class="wideContentBlock">
  <div class="block_content">
    Op dinsdagavond 21 september organiseren we weer een workshop voor Google Adwords. Hoe kun je
    deze effectief en efficiënt inzetten?
  </div>
  ...
  <div class="block_content">
    ...
  </div>
</div>
```

```
<section>
  <article>
    Op dinsdagavond 21 september organiseren we weer een workshop voor Google Adwords. Hoe kun je
    deze effectief en efficiënt inzetten?
  </ article >
  ...
  <article>
    ...
  </ article >
</ section >
```

Naast het feit dat browsers weten waar de informatie te vinden is, is dit ook handig voor zoekmachines. Ze weten immers precies waar de belangrijke informatie (de content) van de website is opgeslagen. Meer over dit onderwerp in hoofdstuk vier.



2.2.5 - <aside> element

*"The aside element represents a section of a page that consists of content that is tangentially related to the content around the aside element, and which could be considered separate from that content. Such sections are often represented as sidebars in printed typography."*⁹

Het <aside> element wordt gebruikt om informatie gerelateerd aan het omliggende element (waar het <aside> element in staat) in te plaatsen. Zo hebben bepaalde websites aan de zijkant een verzameling van links naar andere websites of een weergave van de laatste tweets. Beide informatiebronnen zijn gerelateerd aan de inhoud/onderwerp van de website. Als het <aside> in een <article> element staat kan het bijvoorbeeld gebruikt worden om een quote mee aan te geven (zoals vaak in tijdschriften wordt gedaan) of om de bronverwijzing in te plaatsen. Voorbeeld (overgenomen uit de WHATWG specificatie⁹ – context: een quote uit het artikel):

```
<article>
  <p>He later joined a large company, continuing on the same work.
  I love my job. People ask me what I do for fun when I'm not at
  work. But I'm paid to do my hobby, so I never know what to
  answer. Some people wonder what they would do if they didn't have to
  work... but I know what I would do, because I was unemployed for a
  year, and I filled that time doing exactly what I do now.</p>

  <aside>
    <q> People ask me what I do for fun when I'm not at work. But I'm
    paid to do my hobby, so I never know what to answer. </q>
  </aside>
</article>
```

2.2.6 - <footer> element

*"The footer element represents a footer for its nearest ancestor sectioning content or sectioning root element. A footer typically contains information about its section such as who wrote it, links to related documents, copyright data, and the like."*¹⁰

Veel websites hebben helemaal onderaan de pagina contact informatie, copyright rechten en dergelijke staan. Zoals eigenlijk alle bovenstaande elementen wordt dit met behulp van een <div> gedaan, maar zoals we ook gezien hebben heeft dit geen enkele semantische waarde. Daarom is met HTML5 het <footer> element geïntroduceerd. Voorbeeld:

```
<div id="footer">
  <div id="footerContainer">
    <ul>
      <li>home</li>
      ...
      <li>weblog</li>
    </ul>
    <div id="footerAdress">
      Lloydstraat 138c, 3024 EA Rotterdam, 010 2210190
    </div>
  </div>
</div>
```

```
<footer>
  <nav>
    <ul>
      <li>home</li>
      ...
      <li>weblog</li>
    </ul>
  </nav>
  <div id="footerAdress">
    Lloydstraat 138c, 3024 EA Rotterdam, 010 2210190
  </div>
</ footer >
```



<div id="footerAdress"> blijft hier een <div> aangezien het in dit geval nadrukkelijk wordt gebruikt voor opmaak doeleinden (*"When an element is needed for styling purposes [...] authors are encouraged to use the div element instead."*⁷). Uiteraard zijn er andere alternatieven voor dit stukje tekst: <p> of <address>. Net als het <header> element kan het <footer> element op meerdere plaatsen gebruikt worden. Voorbeeld waarin de 'Lees verder' link in de footer van een artikel staat:

```
<div class="wideContentBlock">
  <div class="block_content">
    Op dinsdagavond 21 september organiseren we weer een workshop voor Google Adwords. Hoe kun je
    deze effectief en efficiënt inzetten?
    <a href="hyperlink">Lees verder</a>
  </div>
</div>
```

```
<section>
  <article>
    Op dinsdagavond 21 september organiseren we weer een workshop voor Google Adwords. Hoe kun je
    deze effectief en efficiënt inzetten?
    <footer> <a href="hyperlink">Lees verder</a> </footer>
  </ article >
</ section >
```

2.2.7 - <hgroup> element

*"The hgroup element represents the heading of a section. The element is used to group a set of h1–h6 elements when the heading has multiple levels, such as subheadings, alternative titles, or taglines."*¹¹

Boeken hebben vaak een titel en hoofdstukken, maar het boek kan ook een subtitel hebben. We gebruiken het boek 'HTML5: Up and Running' als voorbeeld. De titel is 'HTML5' (wordt aangegeven met <h1>), terwijl alle hoofdstukken direct onder de titel komen (dus <h2>). Maar hoe zit het dan met de subtitel 'Up and Running'? Als deze in een <h2> wordt gezet, zit het op hetzelfde niveau in de DOM als de hoofdstukken. Als de hoofdstukken omgezet worden naar <h3>, worden de hoofdstukken 'kinderen' van de subtitel. Beide handelingen leveren niet het beoogde resultaat op. Voorheen zou dit probleem opgelost worden door de subtitel in een <p> of te plaatsen. In HTML5 is hier het <hgroup> element voor, waarmee de titel en subtitel als het ware worden samenvoegt. Voorbeeld (met het boek):

```
<h1>HTML5</h1>
<p>Up and Running</p>

<h2>1. How Did We Get Here?</h2>
  <h3>Diving In</h3>
  <h3>MIME Types</h3>
  ...
...
```

```
<hgroup>
  <h1>HTML5</h1>
  <h2>Up and Running</h2>
</hgroup>

<h2>1. How Did We Get Here?</h2>
  <h3>Diving In</h3>
  <h3>MIME Types</h3>
  ...
...
```



2.2.8 - <time> element

"The time element represents either a time on a 24 hour clock, or a precise date in the proleptic Gregorian calendar, optionally with a time and a time-zone offset. This element is intended as a way to encode modern dates and times in a machine-readable way so that, for example, user agents can offer to add birthday reminders or scheduled events to the user's calendar."¹²

Het komt er op neer dat het <time> element vooral bedoelt is om de browsers bepaalde tijdstippen kenbaar te maken. Zo kunnen er, zoals in de quote staat, bijvoorbeeld dagen van verjaardagen mee worden aangegeven. Maar het kan ook gebruikt worden om de publiceerdatum van een nieuwsbericht in op te slaan. De ontwikkelaar moet zelf de datum toevoegen; bijvoorbeeld hardcoded of doormiddel van PHP. Voorbeeld:

```
<div class="block_content">
  <h4>Adwords workshop op 21 september</h4>
  <span class="blockDate">17 augustus 2010</span>
  ...
</div>
```

```
<article>
  <header>
    <h4> Adwords workshop op 21 september</h4>
    <time datetime="2010-08-17" pubdate="pubdate">17 augustus 2010</time>
  </header>
  ...
</ article >
```

Via het pubdate attribuut wordt aangegeven dat de datetime de publicatiedatum is van het element waar het in staat. Als het binnen een <article> element staat (zoals nu) is het de publicatiedatum van het artikel. Als het buiten een <article> staat is het de publicatiedatum van het gehele document. In hoofdstuk vier wordt er dieper op ingegaan, maar het <time> element heeft voordelen voor SEO: zo kan er met behulp van de publicatiedatum bijvoorbeeld de relevantie van een webpagina/website berekend worden. Hoe recenter het document, hoe meer relevantie het zal hebben.

2.2.9 - <mark> element

"The mark element represents a run of text in one document marked or highlighted for reference purposes, due to its relevance in another context. When used in a quotation or other block of text referred to from the prose, it indicates a highlight that was not originally present but which has been added to bring the reader's attention to a part of the text that might not have been considered important by the original author when the block was originally written, but which is now under previously unexpected scrutiny. When used in the main prose of a document, it indicates a part of the document that has been highlighted due to its likely relevance to the user's current activity."¹³

Bovenstaande uitleg maakt het er niet duidelijker op. Misschien levert de uitleg *"Think of the <mark> element as a highlighter. A string wrapped within this tag should be relevant to the current actions of the use."*¹⁴ een beter beeld op. Het <mark> element kan kortom gezien worden als een digitale markeerstift, waarmee belangrijke informatie kan worden 'aangestreept'. Het kan bijvoorbeeld gebruikt worden om de meest relevante stukken tekst/zoekresultaten mee aan te geven. De gebruiker kan in één oogopslag zien wat er belangrijk is of welk zoekresultaat het meest van toepassing is.

Voorbeeld (overgenomen uit de WHATWG HTML5 specificaties ¹³ – Context: zoekresultaten op 'kitten' worden gemarkeerd):

```
<p>I also have some <mark>kitten</mark>s who are visiting me these days. They're really cute. I think they like my garden! Maybe I should adopt a <mark>kitten</mark>.</p>
```



2.2.10 - <figure> element

"The element can thus be used to annotate illustrations, diagrams, photos, code listings, etc, that are referred to from the main content of the document, but that could, without affecting the flow of the document, be moved away from that primary content, e.g. to the side of the page, to dedicated pages, or to an appendix."¹⁵

Het <figure> element is een container waarin onafhankelijke informatie kan worden opgeslagen. Een goed voorbeeld hiervan zijn één of meer afbeeldingen met een uitleg erbij (<figcaption>). De afbeeldingen hebben verband met een tekst waarin naar de afbeeldingen wordt gerefereerd. Er staat bijvoorbeeld ergens in de tekst: "**zie afbeelding 1.3.**" De afbeelding zou nog steeds dezelfde waarde hebben mocht de tekst/referentie wegvallen. Dit wordt bereikt door een uitleg te koppelen aan de afbeeldingen en deze erbij weer te geven. De afbeelding en de uitleg worden als het ware door het <figure> element aan elkaar gekoppeld. Voorbeeld^{15/16} (Niet met de *Hoppinger* website):

```
<p>This case centered on some sort of "intellectual property" infringement related to a comic (see Exhibit A). The suit started after a trailer ending with these words:
```

```
...
```

```

```

```
<p>Exhibit A. The alleged rough copycomic.</p>
```

```
...
```

```
<p>This case centered on some sort of "intellectual property" infringement related to a comic (see Exhibit A). The suit started after a trailer ending with these words:
```

```
...
```

```
<figure>
```

```
  
```

```
  <figcaption>Exhibit A. The alleged <cite>rough copy</cite> comic.</figcaption>
```

```
</figure>
```

```
...
```



2.3 – Forms

Het formulier speelt een belangrijke rol op het web; het is namelijk het medium om als gebruiker informatie te verzenden naar de beheerder/website/database. Zonder formulieren zou het internet een groot deel van zijn functionaliteit verliezen, aangezien veel werkzaamheden gebruik maken van formulieren: inloggen/registreren, een bericht op Facebook posten en wat te denken van zoekmachines. Het meest essentiële onderdeel van een zoekmachine is het formulier waar de gebruiker de zoekterm invult.

Het formulier is sinds de begindagen steeds verder uitgebreid, zodat er bijvoorbeeld validatie kan plaatsvinden op de ingevoerde waarden. De WHATWG gaf in het nieuwsartikel over hun oprichting¹⁷ aan dat de hoofdwerkzaamheden tot dan aan toe voornamelijk bij het uitbreiden van formulieren lag: *“The main focus up to this point has been extending HTML4 Forms to support features requested by authors, without breaking backwards compatibility with existing content.”*¹⁷ Deze ontwikkelingen vonden eerst plaats in het ‘Web Forms 2.0: An incremental improvement of HTML4.01's forms’ document, maar werd later samengevoegd met het ‘Web Apps 1.0: Features for Application Development in HTML’ document tot HTML5.

Het testen van de besproken input types, attributen en elementen in dit hoofdstuk heeft in de volgende browsers plaatsgevonden:

- Opera 10.62
- Chrome 6.0.472.63
- Internet Explorer 8.0.6001
- Safari 4.0.5 8.0.6001
- Firefox 3.6.10

2.3.1 - Input types

2.3.1.1 - Email Address & Web Address

Deze twee input types zijn ontwikkeld met mobiele browsers in het achterhoofd, en voornamelijk voor smartphones waar tekst invoer op het scherm plaatsvindt. Als voorbeeld nemen we de *iPhone* browser. Zodra de gebruiker de focus in een invoerveld zet, zal de weergave van het toetsenbord veranderen:

- Email address (zie afbeelding 1): elk e-mailadres bestaat uit minimaal een punt en @ teken en spaties zijn niet toegelaten. Het toetsenbord zal daarom de twee tekens weergeven en maakt de spatiebalk kleiner dan in de standaard weergave.¹⁸
- Web address (zie afbeelding 2): de spatiebalk is weggelaten, de punt en slash zijn weergegeven en tevens een knop om een achtervoegsel (.com, .org, .net) mee toe te voegen.

In desktop browsers is deze functionaliteit uiteraard niet aanwezig en zal er geen visueel verschil zijn in vergelijking met een standaard invoerveld. Het is echter wel zo dat het vervangen van een tekstveld naar een email of url invoerveld voordelen heeft met validatie. Meer over valideren van invoervelden in hoofdstuk 2.3.4.

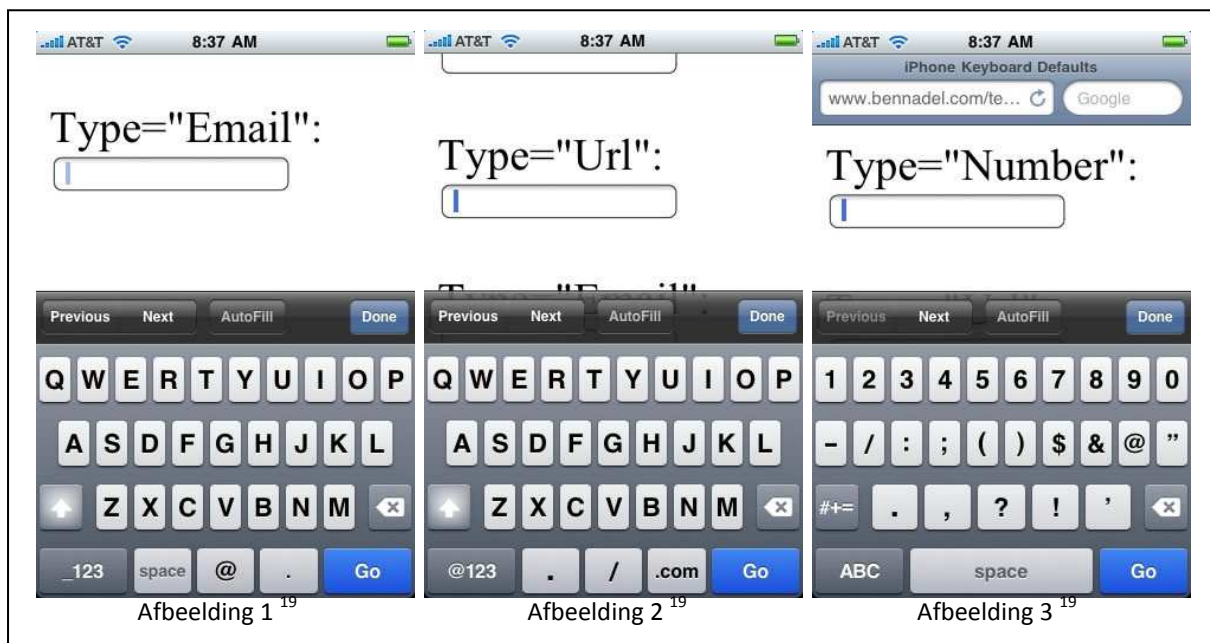
```
<input type="email">
```

```
<input type="url">
```

2.3.1.2 - Telephone Number

Dit input type kan, zoals de naam al doet vermoeden, gebruikt worden om telefoonnummers in te voeren. Qua uiterlijk verschilt er niets met een standaard tekstveld en validatie is standaard niet mogelijk. Elk land heeft namelijk andere regels, lengtes als het om telefoonnummers gaat. In onderdeel 2.3.2.6 wordt echter een attribuut toegelicht waarmee het valideren of limiteren van inhoud alsnog kan plaatsvinden.

```
<input type="telephone">
```



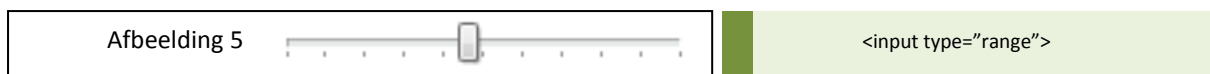
2.3.1.3 - Numbers

Voor het aangeven van nummers zijn er een tweetal input types in HTML5 toegevoegd: `<input type="number">` en `<input type="range">`. Voorheen werd hier een standaard tekstveld voor gebruikt.

Wanneer een mobiele browser (in ieder geval die op de *iPhone* – zie afbeelding 3) dit input type tegenkomt wordt het toetsenbord aangepast: cijfers in de plaats van letters. Desktop browsers geven het op het moment van schrijven op twee manieren weer: *Opera* (zie afbeelding 4) en *Chrome* voegen spinboxes (pijl omhoog en omlaag) aan de rechterkant van het invoerveld toe. *Internet Explorer*, *Firefox* en *Safari* geven het als een standaard tekstveld weer.



Het verschil tussen number en range zit hem voornamelijk in de weergave: *Opera* (zie afbeelding 5), *Safari* en *Chrome* geven in de plaats van een invoerveld een schuifbalk. *Internet Explorer* en *Firefox* geven het als een standaard tekstveld weer.



Aan beide input types kunnen de volgende attributen worden meegegeven:

- Min : de laagst mogelijke waarde
- Max : de hoogst mogelijke waarde
- Step: met elke muisklik wordt deze waarde bij de huidige waarde opgeteld of afgetrokken.
- Value: de beginwaarde die wordt weergegeven.



2.3.1.4 - Date Picker

Op dit moment ondersteunt alleen *Opera* deze input types, de overige browsers geven standaard tekstvelden weer. Er zijn totaal een zestal input types waarmee data gekozen kan worden:

- **Date:** De dropdown button geeft een kalender weer waarin een maand, jaar en dag gekozen kan worden. (Afbeelding 6)

```
<input type="date">
```

- **Month:** Dezelfde kalender wordt weergegeven als bij date, maar in het invoerveld wordt alleen het jaar en maand weergegeven. (Afbeelding 7)

```
<input type="month">
```

- **Week:** Dezelfde kalender wordt weergegeven als bij date, maar in het invoerveld wordt alleen het jaar en week weergegeven. (Afbeelding 8)

```
<input type="week">
```

- **Time:** Met een spinbox of door het zelf in te voeren kan de tijd worden aangegeven. (Afbeelding 9)

```
<input type="time">
```

- **Datetime en Datetime-local:** Beide input types zijn een samenvoeging van het date en time input type. Met de datetime input type gaat het om de tijd in de UTC tijdzone²⁰, terwijl datetime-local de tijd aangeeft in tijdzone van de ontwikkelaar (in dit geval CET). (Afbeelding 10)

```
<input type="datetime(-local)">
```

Afbeelding 6

Afbeelding 7

Afbeelding 8

Afbeelding 9

Afbeelding 10

2.3.1.5 - Search Box

Als je het search invoerveld vergelijkt met standaard tekstvelden, zal er weinig verschil te merken zijn. De context van de invoer blijft namelijk hetzelfde, alleen de weergave in sommige browsers verschilt: *Safari* en *Chrome* tonen een 'X'-icoontje aan de rechterkant van het invoerveld. Hiermee kan in één klik de inhoud van het invoerveld leeggemaakt worden.

Afbeelding 11

```
<input type="search">
```



2.3.2 - Attributen

2.3.2.1 - Placeholder

Met het placeholder attribuut kan standaard inhoud meegegeven worden aan een invoerveld (afbeelding 12). Zodra er in het invoerveld geklikt wordt zal de tekst verdwijnen (afbeelding 13). Als de gebruiker niks invoert en het invoerveld verlaat komt de standaard inhoud weer terug. Als de gebruiker wel data invoert zal de nieuwe waarde worden weergegeven.



```
<input type="text" placeholder="Typ hier je tekst..." />
```

2.3.2.2 - Autofocus

Met autofocus is het mogelijk om aan te geven naar welk invoerveld de browser moet gaan, zodra de pagina is geladen. Stel dat een pagina alleen een zoekbalk bevat (zoals *Google* bijvoorbeeld), dan is het handig om autofocus te gebruiken. Dit scheelt één extra handeling, aangezien de gebruiker niet het invoerveld hoeft aan te klikken. Als er op meerdere invoervelden het autofocus attribuut wordt geplaatst, zal de functionaliteit alleen op het laatst aangegeven veld worden uitgevoerd.

Tot nu toe moest er *JavaScript* code aan te pas komen om deze functionaliteit aan te bieden. In het onderstaande voorbeeld is het formulier weggelaten, maar het belangrijk om te weten dat aan het invoerveld `id="element"` wordt meegegeven):

```
<script type="text/JavaScript">
    function formfocus() {
        document.getElementById('element').focus();
    }
    window.onload = formfocus;
</script>
```

```
<input type="text" autofocus="autofocus" />
[Het is ook mogelijk om het als volgt aan te geven: <input type="text" autofocus />]
```

2.3.2.3 - Required

Voor het registreren op een website zijn er altijd een aantal waarden, die verplicht zijn; o.a. gebruikersnaam, e-mailadres en wachtwoord. Voor dit soort situaties is er in *HTML5* het `required` attribuut toegevoegd. Als dit attribuut aan een `input type` wordt meegegeven betekent dit dat de gebruiker verplicht is om het in te voeren. Op het moment van schrijven lijkt alleen *Chrome* dit te ondersteunen: als het veld niet is ingevuld en de gebruiker drukt op de submit knop, zal de focus op het niet ingevulde invoerveld worden gezet. Waarden in andere invoervelden blijven ingevuld. Voorheen moest de ontwikkelaar zelf een stuk code in *JavaScript* of *PHP* schrijven om te kijken of de verplichte velden ingevuld waren. In *HTML5* wordt het als volgt aangegeven:

```
<input type="text" required=" required " />
[Het is ook mogelijk om het als volgt aan te geven: <input type="text" required />]
```



2.3.2.4 - List

Met behulp van het list attribuut koppel je een lijst met data aan een bepaald invoerveld toe. Wat het doet is een normaal invoerveld samenvoegen met een drop-down lijst (oftewel het <select> input type). Het list attribuut is ideaal om waarden voor te leggen aan de gebruiker. Bijvoorbeeld; top 10 zoektermen of alle e-mailadressen van de werknemers.

Als voorbeeld een standaard invoerveld met het list attribuut. De gebruiker kan zelf een waarde intypen, maar kan ook één van de waarden uit de lijst kiezen.

Afbeelding 14	<input type="text" value="Cerv"/>
	<input type="text" value="Gazelle"/>
	<input type="text" value="Batavus"/>
	<input type="text" value="Giant"/>

Code voor bovenstaand voorbeeld is als volgt:

```
Text (+ List): <input type="text" list="fietsmerken" /></p>
<datalist id="fietsmerken">
  <option value="Gazelle" />
  <option value="Batavus" />
  <option value="Giant" />
</datalist>
```

2.3.2.5 - Multiple

*"The multiple attribute is a boolean attribute that indicates whether the user is to be allowed to specify more than one value."*²¹

Met dit attribuut wordt aangegeven dat het mogelijk is om meerdere waarden in één invoerveld in te voeren. Dit is bijvoorbeeld handig bij het file input type, waarmee er bestanden op de computer kan geselecteerd worden. Om de functionaliteit van dit attribuut te gebruiken moet het in het input type element worden geplaatst:

```
<input type="text" multiple="multiple" />
[Het is ook mogelijk om het als volgt aan te geven: <input type="text" multiple />]
```

2.3.2.6 - Pattern

Met het pattern attribuut kan aangegeven worden waar een invoerveld aan moet voldoen. Stel dat men een formulier heeft waar de gebruiker een postcode moet invullen. Een Nederlandse postcode bestaat altijd uit vier cijfers en twee (hoofd)letters. Aan het invoerveld zou dan meegegeven kunnen worden dat er totaal vier cijfers verwacht worden of juist twee letters. Validatie van de waarden wordt hierdoor een stuk makkelijk. Voorbeeld:

```
Text (+ Pattern):
<input type="text" pattern="[0-9]{4}" title="Vier letters" />
<input type="text" pattern="[A-Z]{2}" title="Twee cijfers" /></p>
```



2.3.3 - Elementen

Opmerking: deze drie elementen staan in de WHATWG specificaties in het hoofdstuk met betrekking tot forms. Maar alleen bij het <output> element is er sprake van interactie met een formulier. De overige twee elementen kunnen afzonderlijk van het <form> element gebruikt worden. Toch worden ze hier besproken aangezien ze ook op deze manier in de specificaties zijn gedocumenteerd.

2.3.3.1 - Output

"The output element represents the result of a calculation."²²

Dit element is geïmplementeerd in HTML5 om de volgende rol te vervullen: het tonen van het resultaat van een berekening van twee invoervelden. Het meest voor de hand liggende voorbeeld is een simpele rekenmachine;

```
<form onsubmit="return false">
  <input name=a type=number step=any> +
  <input name=b type=number step=any> =
  <output onforminput="value = a.valueAsNumber + b.valueAsNumber"></output>
</form>
```

Bovenstaand voorbeeld levert het onderstaande resultaat op: een tweetal invoervelden (met input type number), die worden opgeteld zonder het gebruik van een submit knop. Het is te vergelijken met een AJAX request, maar in dit geval hoeft de ontwikkelaar zelf geen achterliggende code met JavaScript te schrijven. Het gebruik van dit element kan de ontwikkelaar een hoop werk schelen.

Afbeelding 15 + = 8

2.3.3.2 - Progress

"The progress element represents the completion progress of a task. The progress is either indeterminate, indicating that progress is being made but that it is not clear how much more work remains to be done before the task is complete, or the progress is a number in the range zero to a maximum, giving the fraction of work that has so far been completed."²³

Met dit element kan aangegeven worden hoever men is met een bepaalde taak. Zo kan er bijvoorbeeld worden weergegeven hoever een download is:

```
<progress value="250" max="1000">
  <span id="downloadProgress">25</span>%
</progress>
```

Afbeelding 16

De bovenstaande afbeelding is de standaard weergave van een progress bar (werkt alleen in Chrome). De progress bar bevat ook een animatie; witte veeg die van links naar rechts over het groene deel beweegt. De minimaal vereiste attributen voor dit element zijn het value en max attribuut. In het geval van het voorbeeld is de tekst statisch, maar met gebruik van JavaScript of PHP kan de tekst dynamisch worden en wordt het functioneel.



Er kan bijvoorbeeld met het `<progress>` element aangeven worden hoe ver de lezer is in een boek:

```

<?php
    $max = 35; // aantal pagina's
    $value = $_GET['PAGE']; // huidige pagina
    $progress = (100 / $max) * $value;
?>

<progress value="<?php echo $value; ?>" max="<?php echo $max; ?>"
    <span id="readingProgress"><?php echo progress' ?></span>%
</progress>

```

2.3.3.3 - Meter

*"The meter element represents a scalar measurement within a known range, or a fractional value; for example disk usage, the relevance of a query result, or the fraction of a voting population to have selected a particular candidate. This is also known as a gauge."*²⁴

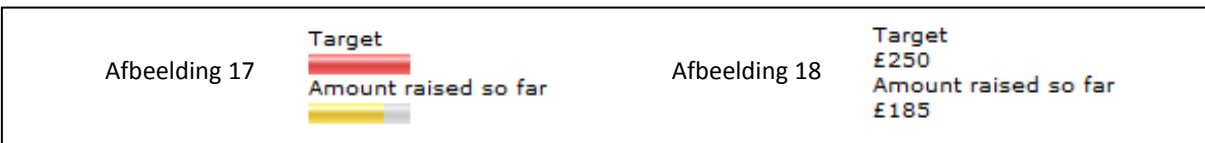
Het `<meter>` element wordt gebruikt om aan te geven hoeveel eenheden er behaald zijn van het totaal. Denk bijvoorbeeld aan een goede doelen actie waar van te voren een doel is vastgesteld. Met het `<meter>` element kan worden aangegeven wat het doel is en hoever men nu is in het behalen van dit doel. Voorbeeld:²⁴

```

<dl>
    <dt>Target</dt>
    <dd><meter max="250" value="250" title="pounds">£250</meter></dd>
    <dt>Amount raised so far</dt>
    <dd><meter min="0" max="250" low="50" high="200" value="185" optimum="225" title="pounds">£185</meter></dd>
</dl>

```

De gegevens die in het `<meter>` element staan worden vervolgens op een visuele manier weergegeven: standaard wordt dit gedaan met balkjes (werkt alleen in *Chrome* - zie afbeelding 17). Sommige browsers ondersteunen dit element echter niet en daarom zijn de bedragen nogmaals toegevoegd tussen de `<meter>` tags. In de niet ondersteunende browser zullen de balkjes vervangen worden met de daadwerkelijke waarden (zie afbeelding 18).



2.3.4 - Input validatie

In bovenstaande tekst wordt een aantal keer het woord 'valideren' gebruikt en dat de nieuwe input types dit makkelijker maken. Het probleem is echter dat het ligt aan de browser, die men gebruikt. Zo is het tot nu toe alleen mogelijk geweest om bepaalde invoervelden te laten valideren in *Opera*. Op de volgende input types zal automatische validatie plaatsvinden wanneer men op de submit knop drukt: email, url en number. Zodra de waarde niet klopt met hetgeen dat er verwacht wordt zal er een foutmelding uit het invoerveld schuiven (zie afbeelding 19²⁵).



Het automatisch valideren van waarden voor deze drie input types staat altijd aan. Mocht een ontwikkelaar echter willen dat er geen validatie plaatsvindt kan er *novalidate* toegevoerd worden aan het `<form>` element.



2.4 - Video en Audio

2.4.1 - Video

Eén van de doelen van de WHATWG met *HTML5* was om externe plugins verleden tijd te laten zijn. Dit wil men onder andere bereiken met het `<video>` element, dat plugins als *Flash*, *Silverlight* en *QuickTime* onnodig maakt. In het afgelopen jaar zijn een aantal organisaties al overgestapt op het *HTML5* `<video>` element; *TEDX*²⁶ en *Youtube*²⁷. Het is geen toeval dat beide bronnen mobiel gebruik betreffen: tot voor kort was het onmogelijk om filmpjes op het internet op mobiele telefoons (lees: smartphones) af te spelen. Er was namelijk geen *Flash* (sinds versie 2.2 wel op *Android* telefoons), *Silverlight* of *Quicktime* ondersteuning. Het was op sommige toestellen wel mogelijk om een video van het internet in een externe videospeler af te spelen, maar dit leverde wel extra handelingen op.

Het probleem met het video element, op het moment van schrijven, is de ondersteuning voor video containers. Er zijn namelijk drie grote containers (OGG, H.264 en WebM) en geen van deze drie worden door alle browsers ondersteund. Waarom dan niet gewoon blijven werken met hetgeen dat er nu is? Dit onderwerp zal aangesneden worden in hoofdstuk vijf. Ontwikkelaars zullen dus genoodzaakt zijn om alternatieven te bieden als ze de video in elke browser willen af laten spelen. Als de ontwikkelaar bijvoorbeeld tevreden zou zijn met het feit dat de video alleen in Safari en *Chrome* werkt, kan het `<video>` element als volgt worden gebruikt:

```
<video width="600" height="400" src="hoppweekend2010.webm" type='video/webm; codecs="vp8, vorbis"'></video>
```

Als de ontwikkelaar wil dat de video in elke browser kan worden afgespeeld, moet het als volgt gebruikt worden:

```
<video width="600" height="400">
  source src="hoppweekend2010.mp4" type='video/mp4; codecs="avc1.42E01E, mp4a.40.2"'>
  <source src="hoppweekend2010.webm" type='video/webm; codecs="vp8, vorbis"'>
  <source src="hoppweekend2010.og4" type='video/ogg; codecs="theora, vorbis"'>
</video>
```

In elk `<video>` element moeten in iedere geval de breedte, hoogte, bron en type staan. Het type attribuut bestaat uit het bestandstype van de video en de video (VP8) en audio (Vorbis) codecs. Daarnaast kunnen de volgende attributen worden gebruikt:

- **Controls** : Hiermee worden een standaard set van controls aangeboden: play/pause, volume en progressiebalk (zie afbeelding 20 voor de standaard weergave in Opera – standaard weergave verschilt per browser).
- **Preload** : Hiermee wordt aangegeven dat de video gedownload moet worden zodra de pagina aan het laden is.
- **Autoplay** : Hiermee wordt aangegeven dat de video moet gaan spelen zodra de pagina aan het laden is.
- **Poster** : Hiermee kan een afbeelding worden weergegeven zolang de video niet beschikbaar is.
- **Loop** : De video begint opnieuw met spelen, zodra deze volledig is afgespeeld.

De browser speelt altijd het eerste videobestand uit de lijst af die ondersteund wordt. Overige bestanden uit de lijst zullen negeert worden.





2.4.2 - Audio

Het `<audio>` element lijkt erg veel op het `<video>` element. De redenen voor de ontwikkeling zijn hetzelfde en de problemen met containers zijn hetzelfde. Op dit moment zijn er een viertal containers die in verband worden gebracht met het `<audio>` element: OGG/Vorbis, MP3, WAV, ACC. Net als bij het `<video>` element wordt geen enkele container door alle browsers ondersteund. Het verschil tussen beide media gerelateerde elementen zit hem vooral in de weergave. De simpelste vorm van gebruik (in dit geval ondersteund door *Safari* en *Chrome*) is als volgt:

```
<audio src="horse.mp3" controls="controls"></audio>
```

Attributen die voor het `<audio>` van toepassing zijn:

- **Controls** : Controls : Hiermee worden een standaard set van controls aangeboden: play/pause, volume en progressiebalk (zie afbeelding 21 voor de standaard weergave in Opera).
- **Preload** : Hiermee wordt aangegeven dat het geluidsfragment gedownload moet worden zodra de pagina aan het laden is.
- **Autoplay** : Hiermee wordt aangegeven dat het geluidsfragment moet gaan spelen zodra de pagina aan het laden is.
- **Loop** : Het geluidsfragment begint opnieuw met spelen, zodra deze volledig is afgespeeld.

Afbeelding 21



2.5 - Canvas

“The canvas element provides scripts with a resolution-dependent bitmap canvas, which can be used for rendering graphs, game graphics, or other visual images on the fly.”²⁸

Als er strikt vanuit een *HTML* standpunt naar canvas gekeken wordt, kan er worden geconcludeerd dat het enkel een container is, een leeg element. De enige vereiste attributen dat het heeft zijn `height` en `width`. Het `<canvas>` element is alleen bedoeld om ruimte beschikbaar te stellen in een webpagina om te tekenen. Het tekenen zelf wordt gedaan met behulp van *JavaScript*.

Canvas is net als video en audio bedoeld om externe plugins onnodig te maken. Op het moment worden veel reclame banners en animaties met Flash vervaardigd. Om dit te laten werken moet de eindgebruiker de *Flash Player* plugin hebben geïnstalleerd. Canvas is in het leven geroepen om onder andere de bovenstaande functionaliteiten over te nemen. Tevens zal canvas ondersteund worden op mobiele platformen en zal daardoor nieuwe mogelijkheden voor mobiele gebruikers bieden.

Maar hoe werkt het `<canvas>` element dan? Als voorbeeld gebruik ik een ingekorte versie van het *Hoppinger Logo* experiment. Als eerste reserveren we ruimte in de webpagina om te tekenen en daarna vullen we deze ruimte m.b.v. *JavaScript*.

```
<canvas id='canvas' width='1200' height='800'></canvas>
```

```
<script type="text/JavaScript">
  window.onload = function()
  {
    canvas = document.getElementById('canvas');
    context = canvas.getContext('2d');

    logo = new Image();
    logo.src = 'hoppinger.png';

    context.drawImage(logo, 0, 0);
  }
</script>
```



Wanneer de pagina aan het laden is wordt de onload functie doorlopen. Hierin wordt eerst het canvas element opgeslagen (met behulp van het id) en daarna aangegeven dat er met de 2d canvas API zal worden gewerkt. In dit geval tekenen we een afbeelding op het canvas; dit wordt gedaan door een variabele aan te maken en hierin de bron van de afbeelding te plaatsen. Het enige dat nog overblijft is om aan te geven dat de afbeelding getoond moet worden. Dat wordt gedaan via de drawImage functie. Parameters die in dit geval worden meegegeven zijn de lokatie van de afbeelding en de x- en y positie.

Het resultaat is het *Hoppinger* logo in de linkerbovenhoek van de canvas. Dit had uiteraard ook bereikt kunnen worden met een element. Maar stel dat de afbeelding geanimeerd zou moeten worden? Dat kan niet met het element en daar is het <canvas> element ideaal voor.

2.6 - Microdata

*"Sometimes, it is desirable to annotate content with specific machine-readable labels, e.g. to allow generic scripts to provide services that are customised to the page, or to enable content from a variety of cooperating authors to be processed by a single script in a consistent manner. For this purpose, authors can use the microdata features described in this section. Microdata allows nested groups of name-value pairs to be added to documents, in parallel with the existing content."*²⁹

Waar microdata op neerkomt is dat gerelateerde informatie wordt opgeslagen in een virtueel 'dossier'. In dit 'dossier' kan bijvoorbeeld informatie van een 'about me'-pagina of contactinformatie van een bedrijf worden opgeslagen. Deze data kan vervolgens door zoekmachines gebruikt worden om de zoekresultaten specifieker te maken. In praktijk is het alsof de ontwikkelaar eigen semantische elementen bedenkt en meegeeft aan de data. Een voorbeeld: de regel <dd itemprop="name">Mark Pilgrim</dd> (uit onderstaand codevoorbeeld) zou bijvoorbeeld gelijk staan aan <person>Mark Pilgrim</person>. Zowel mens als machine weet in één oogopslag wat dit element voor waarde zal bevatten: de naam van een persoon. Op dit moment zijn er soortgelijke technieken beschikbaar (zoals *RDFa* en *microformats*), maar dit zijn uitbreidingen voor *(X)HTML*. In tegenstelling tot microdata maken deze technieken niet deel uit van een *HTML* of *XHTML* specificatie, maar hebben eigen specificaties.³⁰

Voorbeeld (overgenomen uit *'HTML5: Up and Running'*³¹):

```
<section itemscope itemtype="http://data-vocabulary.org/Person">
...
<h1>Contact Information</h1>
<dl>
  <dt>Name</dt>
  <dd itemprop="name">Mark Pilgrim</dd>

  <dt>Position</dt>
  <dd>
    <span itemprop="title">Developer advocate</span> for
    <span itemprop="affiliation">Google, Inc.</span>
  </dd>

  <dt>Mailing address</dt>
  <dd itemprop="address" itemscope itemtype="http://data-vocabulary.org/Address">
    <span itemprop="street-address">100 Main Street</span><br>
    <span itemprop="locality">Anytown</span>,
    <span itemprop="region">PA</span>
    <span itemprop="postal-code">19999</span><br>
    <span itemprop="country-name">USA</span>
  </dd>
</dl>
...
</section>
```



De eerste stap in het proces is om 'dossier' aan te maken om de informatie in op te slaan. In dit geval wordt dat gedaan in het <section> element. Het element waar het 'dossier' in is geopend bevat normalerwijs alle informatie, dat zal worden opgenomen. Het is dus niet toegestaan om in het 'dossier Person' informatie op te nemen uit een <header> element. Met het itemtype wordt een locatie aangegeven waar het 'dossier' te vinden is. Data wordt aangegeven doormiddel van het itemprop attribuut (komt in het element te staan waar de data zich in bevind). Het address wordt in een apart dossier gestopt, maar wel met een koppeling naar het 'dossier Person'.

Wat er gecreëerd is, is een 'dossier' met daarin informatie dat leesbaar is voor machines. Zoals eerder aangegeven kan deze data door zoekmachines gebruikt worden om de zoekresultaten specifiek te maken. Hieronder het resultaat van bovenstaande microdata:

Afbeelding 21	About Mark Pilgrim Anytown PA - Developer advocate - Google, Inc. Excerpt from the page will show up here. Excerpt from the page will show up here. diveintohtml5.org/examples/person-plus-microdata.html - Cached - Similar pages
---------------	--

Wat er in het resultaat te zien is, is dat de tweede regel (in het lichtgrijs) bestaat uit informatie uit het 'dossier'. Stel dat er ook een pasfoto was opgenomen in het dossier: als deze zou worden weergegeven zal het zoekresultaat nog meer opvallen. En aangezien mensen vaak als eerst afbeeldingen registreren en daarna tekst kan dit een belangrijk detail zijn om gevonden te worden op het web. Als er gezocht wordt op *Mark Pilgrim* toont de zoekmachine (in dit geval *Google*) 495.000 zoekresultaten. Net dat beetje extra informatie kan helpen om het juiste resultaat te vinden.

2.7 - Web Storage en Offline Storage

2.7.1 - Web Storage

Het is belangrijk om te melden dat Web Storage (ook bekend als Local Storage) geen deel meer uitmaakt van de *HTML5* specificaties. In plaats daarvan is het opgenomen in een afzonderlijke specificatie³². Desalniettemin wordt het nog steeds beschouwd en bestempeld als *HTML5*.

Web Storage is een nieuwe (en verbeterde) variant op cookies: ***"A cookie can be used for authentication, storing site preferences, shopping cart contents, the identifier for a server-based session, or anything else that can be accomplished through storing text data."***³³ Het gebruik van cookies is echter allesbehalve ideaal te noemen.³⁴

- Met elk HTTP request worden de cookies meegezonden ook al zijn deze al eens verzonden. Er wordt kortom continue dubbele data verzonden en levert extra en onnodig dataverkeer op.
- Data in cookies is ongecodeerd en kan dus een gevaar opleveren: wanneer de cookies in verkeerde handen komen kan de data uit een cookie worden gehaald. In het geval van inloggegevens kan dit voor grote problemen en/of ongemak zorgen.³⁵
- Het formaat van een cookie is gelimiteerd aan 4KB. Toekomstige web applicaties (in *HTML5*) hebben meer ruimte voor dataopslag nodig dan dit.

Met Web Storage hebben ze deze tekortkomingen van cookies trachten op te lossen. Zo zal de data worden opslagen in de browser van de gebruiker en normalerwijs zal deze data niet uitgewisseld worden met de web server. De maximale opslagruimte per bron is opgevoerd naar 5MB. Net als bij cookies zal de data bewaard blijven, wanneer de gebruiker de browser/pagina afsluit.

Interessant is de besteedde aandacht aan privacy en veiligheid van Web Storage in de specificaties. Zo wordt er onder andere melding gemaakt van het feit dat derden (als voorbeeld werd een adverteerbureau gebruikt) met de local storage data een profiel van gebruikers kan maken en hier advertenties op aanpassen. Dit staat bekend als user tracking. Vooral bedrijven kunnen hier hun voordeel mee doen: ***"For example, a user's shopping wishlist on one domain could be used by another domain for targeted advertising."***³⁶ Het is daarentegen de vraag of de eindgebruiker blij zal zijn met de 'misbruik' van hun opgeslagen webgedrag.



2.7.2 - Offline Storage

*"In order to enable users to continue interacting with Web applications and documents even when their network connection is unavailable [...] authors can provide a manifest which lists the files that are needed for the Web application to work offline and which causes the user's browser to keep a copy of the files for use offline."*³⁷

Zou het niet ideaal zijn als een website/applicatie beschikbaar blijft, terwijl de internetverbinding niet optimaal is? De oplossing voor dit probleem wordt met *HTML5* geïntroduceerd: Offline Storage. Het idee is dat de ontwikkelaar de juiste bestanden aangeeft en dat deze vervolgens door de browser automatisch worden gedownload en up to date gehouden worden. Zodra de internetverbinding wegvalt zal de browser de bestanden aanspreken die lokaal staan opgeslagen. Het resultaat is een werkende website, maar dan wel offline. Het ligt aan de ontwikkelaar in hoeverre de functionaliteiten en weergave hetzelfde zijn als de online versie.

Voorbeeld: als eerst geven we aan welke bestanden er opgeslagen moeten worden. De bestanden (drie in totaal) worden opgeslagen in een .manifest bestand:

```
CACHE MANIFEST
index.html
stylesheet.css
images/hoppingerlogo.jpg
```

Deze drie bestanden zullen worden gedownload zodra de website geopend wordt. Het enige dat nog gedaan moet worden is aangeven dat het manifest bestand gekoppeld is aan de webpagina. Dit wordt als een attribuut meegegeven aan `<html>` element:

```
<html lang="en" manifest="hoppindex.manifest">
```

2.8 - Drag and Drop

Over het algemeen wordt *Internet Explorer* door ontwikkelaars niet als een voorloper op het gebied van implementatie bestempeld. Ook in het geval van *HTML5* loopt IE ver achter bij de overige browsers. Hier komt echter verandering in met versie negen, maar daarover is meer te lezen in hoofdstuk drie. In het geval van drag and drop is *Internet Explorer* echter het voorbeeld geweest voor de WHATWG. In IE5, dat in 1999 op de markt kwam, werd de drag and drop functionaliteit geïntroduceerd. In de *HTML5* specificaties is deze functionaliteit letterlijk overgenomen: *"Since IE already supports drag and drop, and since Safari implements the IE API for drag and drop, it makes a lot of sense to simply use that API."*³⁸

Met drag and drop kunnen elementen tussen een tweetal elementen verslept worden. In het voorbeeld (zie volgende bladzijde) wordt het toegelaten om afbeelding van de 'start' `<div>` naar de 'target' `<div>` te verslepen. Aan de 'start' `<div>` geven we een *ondragstart* event mee (regel #1): het element dat verslepen wordt zal worden opgeslagen (regel #10). Het `` en `<a>` element zijn standaard versleepbaar, maar andere elementen vereisen het *draggable="true/false"* attribuut in de tag (regel #2). Aan de 'target' `<div>` worden de *ondragover* en *ondrop* events meegegeven (regel #4). Met het *ondragover* effect wordt aangegeven hoe de cursor eruit ziet wanneer de gebruiker over het element sleept (regel #15). Als er losgelaten wordt op een element met het *ondrop* event, zal de opgeslagen waarde worden toegevoegd aan dit element (regel #22). Met regel #23 wordt aangegeven dat het drag and drop event voltooid is en kan worden afgesloten.



```
#1 <div id="start" ondragstart="dragStart(event);">
#2   
  </div>
#4 <div id="target" ondragover="return dragOver(event);" ondrop="return drop(event);" > </div>

  <script>
    function dragStart(event)
    {
#10      event.dataTransfer.effectAllowed = 'copy';
      event.dataTransfer.setData("Text", event.target.getAttribute('id'));
    }

    function dragOver(event)
#15    {
      event.dataTransfer.dropEffect = 'move';
      return false;
    }

    function drop(event)
#22    {
#23      var element = event.dataTransfer.getData("Text");
      event.target.parentNode.appendChild(document.getElementById(element));
      event.stopPropagation();
      return false;
    }
  </script>
```

2.9 - Geolocation

Strikt genomen is Geolocation geen onderdeel van *HTML5* en is het, in tegenstelling tot Web Storage, ook nooit onderdeel geweest van *HTML5*. Toch blijkt ook in dit geval dat veel mensen het als *HTML5* beschouwen. Geolocation wordt ontwikkeld door de *Geolocation Working Group*, dat onderdeel is van de W3C en niet door een werkgroep van de WHATWG. Op het moment worden de specificaties bestempeld als de candidate recommendation.³⁹ Dit komt er op neer dat de W3C gelooft dat deze specificaties als voltooid kunnen worden beschouwd.

De functionaliteit van deze Application programming interface (API) is redelijk simpel: met behulp van *JavaScript* worden in ieder geval de longitude (lengtegraad), latitude (breedtegraad) en nauwkeurigheid opgehaald van de gebruikers huidige positie. De API ondersteunt een tweetal functies: `getCurrentPosition` (éénmalig de positie ophalen – voorbeeld: lokatie weergeven op *Google Map*) en `watchPosition` (zodra de positie veranderd wordt deze verandering opgeslagen – voorbeeld: wandeltocht weergeven op een *Google Map*). Het is aan de ontwikkelaar om te beslissen wat er met de data gebeurt.⁴⁰



2.10 - Conclusie

Wat zijn de nieuwe mogelijkheden van HTML5 in vergelijking met XHTML? De mogelijkheden die besproken zijn kunnen worden opgedeeld in een drietal categorieën; (semantische) elementen, formulieren en web applicaties. Een samenvatting van het hoofdstuk is wellicht op zijn plaats gezien de kwantiteit aan informatie.

Als eerste zijn (semantische) elementen besproken die ofwel veranderd zijn of zijn toegevoegd. Veranderen hebben voornamelijk plaats gevonden op de doctype, <html> en <head> elementen. Deze zijn toegepast omdat veel attributen overbodig zijn aangezien deze XHTML specifiek zijn. In andere gevallen kunnen bepaalde attributen weggelaten worden, aangezien de waarde van dit attribuut al is op te halen uit het element of ander attribuut. Daarnaast zijn er een groot aantal nieuwe elementen toegevoegd waarmee de semantiek van HTML documenten aanzienlijk verbeterd wordt. Het <div> element, dat tegenwoordig in overvloed wordt gebruikt, heeft geen enkele semantische waarde: de context van de inhoud kan niet uit het element gehaald worden. Hiervoor zijn in HTML5 een aantal nieuwe elementen geïntroduceerd die allemaal een eigen rol vervullen: header, navigatie, artikelen en footers.

Als tweede is er gekeken naar de vernieuwingen aan webformulieren. Het probleem op het ogenblik is dat de ondersteuning voor deze vernieuwingen gelimiteerd is. Zo blijkt dat de meeste toevoegingen nu vooral nog ondersteund worden in Opera. Echter het doel van de nieuwe input types is duidelijk; validatie van inhoud automatiseren, aanpassen toetsenborden op smartphones en het gebruikersgemak verhogen.

De attributen hebben als hoofddoel om het de ontwikkelaars een stuk gemakkelijker te maken. Het is bijvoorbeeld niet meer nodig om met JavaScript een stuk code te schrijven om de focus op een bepaald invoerveld te plaatsen of om een standaard tekst in een invoerveld weer te geven. Tevens kan er met een tweetal attributen (required en pattern) al een zekere mate van validatie worden toegevoegd aan invoervelden.

Een drietal elementen (in de specificaties gekoppeld aan forms) kunnen helpen in het weergeven van berekeningen (<output>) en progressie (<progress> en <meter>). De progressie wordt weergegeven met balken, maar alleen Chrome ondersteund dit op het moment van schrijven.

Als laatste zijn een aantal functionaliteiten besproken, die gebruikt zullen worden voor web applicaties en het onnodig maken van plugins. Zo werd er voorheen veel Flash gebruikt als het ging om animaties, video- en audio fragmenten. In HTML5 kan dit nu ook allemaal via de <canvas>, <video> en <audio> elementen. En in vergelijking met Flash hoeft de gebruiker zich geen zorgen te maken of hij/zij wel de juiste plug-ins heeft geïnstalleerd.

Stel dat iemand een spel met canvas maakt met meerdere levels. Het zou ideaal zijn als de progressie van de speler opgeslagen wordt en dat er na het herstarten van de browser gewoon doorgespeeld kan worden vanaf het punt waar de speler gestopt was. Hiervoor is Web Storage geïntroduceerd (was eerst deel van HTML5, maar heeft nu een eigen specificatie). En stel dat je ook het spel wilt blijven spelen terwijl de internetverbinding is uitgevallen. Hiervoor is Offline Storage ontwikkelt, waarmee de bronbestanden van een website lokaal bij de gebruiker worden opgeslagen.

2.10.1 - Voordelen

Wat zijn de belangrijkste voordelen van de nieuwe HTML5 mogelijkheden, zoals deze in dit hoofdstuk zijn besproken?

- Informatie in de header (bv. doctype) is drastisch ingekort, zodat het gemakkelijker te onthouden is en minder typewerk/knip- en plakwerk vereist.
- Semantische elementen zorgen voor meer structuur en maken het document leesbaarder voor zowel mens als machine. (Zoek)machines kunnen gemakkelijker de informatie op de pagina indexeren (qua context) en gebruiken in zoekresultaten.
- Nieuwe form input types zijn specifiek bedacht om bepaalde waardes mee aan te geven, zoals bijvoorbeeld data, getallen, email of web adressen.
 - Dit maakt het gemakkelijk om de inhoud te kunnen valideren aangezien de browser weet wat de waarde waarschijnlijk zal bevatten. Een email adres bevat bijvoorbeeld altijd een @ en ‘.’



- Dit maakt het mogelijk om het toetsenbord van mobiele toestellen aan te passen aan de inhoud die verwacht wordt. Bijvoorbeeld alleen cijfers als de focus op een number invoerveld ligt.
- Nieuwe form attributen kunnen gebruikt worden om stukken *Javascript* mee te vervangen: van vijf regels (in het geval van autofocus) naar één woord.
- De <video> en <audio> element maken het mogelijk om deze mediavormen op een webpagina te plaatsen met het gemak van een afbeelding. De gebruiker hoeft verder geen plugins te installeren om de bestanden te zien en/of te luisteren.
- Canvas kan gebruikt worden om animaties te maken met web gebaseerde technieken (*JavaScript*). Ook hier hoeft de gebruiker geen plugin voor te hebben/installeren.
- Met microdata kan de ontwikkelaar als het ware belangrijke data aanvinken om deze op te slaan en leesbaar te maken voor machines. Zoekmachines kunnen deze data gebruiken om zoekresultaten specifieker mee te maken.
- Een website die offline net zo functioneert als wanneer er wel een internetverbinding is, is met *HTML5* te realiseren door de offline storage mogelijkheden.
- Web storage kan gebruikt worden om data van een webpagina lokaal op te slaan: het kan worden gezien als de grotere broer van cookies.
- Met geolocation, ook al maakt dit niet deel uit van de *HTML5* specificaties, kan de locatie van de gebruiker worden opgevraagd.

Bronnen

¹ : The HTML5 Semantics Debate | <http://visitmix.com/opinions/The-HTML5-Semantics-Debate> | Tags are Too Final | Joshua Allen | 31-08-2009 | 22-09-2010

² : HTML 5 Reference | <http://html5tutorial.net/html-5-reference/html-5-reference.html> | | | 22-09-2010

³ : HTML5 (including next generation additions still in development) Draft Standard | <http://www.whatwg.org/specs/web-apps/current-work/multipage/sections.html#the-header-element> | 4.4.8 The header element | Ian Hickson (Editor) | 10-09-2010 | 22-09-2010

⁴ : HTML5: Up and Running | Hfst. 3 : What Does It All mean? – Headers | Blz. 45 | Mark Pilgrim | August 2010 – First Edition | O'Reilly Media, Inc. | 22-09-2010

⁵ : HTML5 (including next generation additions still in development) Draft Standard | <http://www.whatwg.org/specs/web-apps/current-work/multipage/sections.html#the-nav-element> | 4.4.3 The nav element | Ian Hickson (Editor) | 10-09-2010 | 22-09-2010

⁶ : HTML5: Up and Running | Hfst. 3 : What Does It All mean? – Navigation | Blz. 51 | Mark Pilgrim | August 2010 – First Edition | O'Reilly Media, Inc. | 22-09-2010

⁷ : HTML5 (including next generation additions still in development) Draft Standard | <http://www.whatwg.org/specs/web-apps/current-work/multipage/sections.html#the-section-element> | 4.4.2 The section element | Ian Hickson (Editor) | 10-09-2010 | 22-09-2010

⁸ : HTML5 (including next generation additions still in development) Draft Standard | <http://www.whatwg.org/specs/web-apps/current-work/multipage/sections.html#the-article-element> | 4.4.4 The article element | Ian Hickson (Editor) | 10-09-2010 | 22-09-2010

⁹ : HTML5 (including next generation additions still in development) Draft Standard | <http://www.whatwg.org/specs/web-apps/current-work/multipage/sections.html#the-aside-element> | 4.4.5 The aside element | Ian Hickson (Editor) | 10-09-2010 | 23-09-2010

¹⁰ : HTML5 (including next generation additions still in development) Draft Standard | <http://www.whatwg.org/specs/web-apps/current-work/multipage/sections.html#the-footer-element> | 4.4.9 The footer element | Ian Hickson (Editor) | 10-09-2010 | 23-09-2010

¹¹ : HTML5 (including next generation additions still in development) Draft Standard | <http://www.whatwg.org/specs/web-apps/current-work/multipage/sections.html#the-hgroup-element> | 4.4.7 The hgroup element | Ian Hickson (Editor) | 10-09-2010 | 23-09-2010

¹² : HTML5 (including next generation additions still in development) Draft Standard | <http://www.whatwg.org/specs/web-apps/current-work/multipage/sections.html#the-address-element> | 4.4.10 The address element | Ian Hickson (Editor) | 10-09-2010 | 23-09-2010

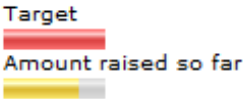
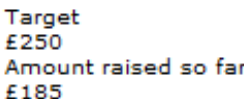


- ¹³ : HTML5 (including next generation additions still in development) Draft Standard | <http://www.whatwg.org/specs/web-apps/current-work/multipage/sections.html#the-address-element> | 4.4.10 The address element | Ian Hickson (Editor) | 10-09-2010 | 23-09-2010
- ¹⁴ : 28 HTML5 Features, Tips, and Techniques you Must Know | <http://net.tutsplus.com/tutorials/html-css-techniques/25-html5-features-tips-and-techniques-you-must-know/> | 22. Mark Element | Jeffrey Way | 01-08-2010 | 23-09-2010
- ¹⁵ : HTML5 (including next generation additions still in development) Draft Standard | <http://www.whatwg.org/specs/web-apps/current-work/multipage/grouping-content.html#the-figure-element> | 4.4.11 The figure element | Ian Hickson (Editor) | 10-09-2010 | 23-09-2010
- ¹⁶ : 28 HTML5 Features, Tips, and Techniques you Must Know | <http://net.tutsplus.com/tutorials/html-css-techniques/25-html5-features-tips-and-techniques-you-must-know/> | 2. The Figure Element | Jeffrey Way | 01-08-2010 | 23-09-2010
- ¹⁷ : WHAT open mailing list announcement | <http://www.whatwg.org/news/start> | | Ian Hickson | 04-05-2004 | 24-09-2010
- ¹⁸ : HTML5: Up and Running | Hfst. 8 : A Form of Madness? – Email Addresses | Blz. 151 | Mark Pilgrim | August 2010 – First Edition | O'Reilly Media, Inc. | 24-09-2010
- ¹⁹ : Default To The Numeric, Email, And URL Keyboards On The iPhone | <http://www.bennadel.com/blog/1721-Default-To-The-Numeric-Email-And-URL-Keyboards-On-The-iPhone.htm> | | 22-09-2009 | 04-10-2009
- ²⁰ : Coordinated Universal Time | http://en.wikipedia.org/wiki/Coordinated_Universal_Time | | 17-09-2010 | 24-09-2010
- ²¹ : HTML5 (including next generation additions still in development) Draft Standard | <http://www.whatwg.org/specs/web-apps/current-work/multipage/common-input-element-attributes.html#the-multiple-attribute> | 4.10.7.2.6 The multiple attribute | Ian Hickson (Editor) | 10-09-2010 | 24-09-2010
- ²² : HTML5 (including next generation additions still in development) Draft Standard | <http://www.whatwg.org/specs/web-apps/current-work/multipage/the-button-element.html#the-output-element> | 4.10.15 The output element | Ian Hickson (Editor) | 26-09-2010 | 27-09-2010
- ²³ : HTML5 (including next generation additions still in development) Draft Standard | <http://www.whatwg.org/specs/web-apps/current-work/multipage/the-button-element.html#the-progress-element> | 4.10.16 The progress element | Ian Hickson (Editor) | 26-09-2010 | 27-09-2010
- ²⁴ : HTML5 (including next generation additions still in development) Draft Standard | <http://www.whatwg.org/specs/web-apps/current-work/multipage/the-button-element.html#the-meter-element> | 4.10.17 The meter element | Ian Hickson (Editor) | 26-09-2010 | 27-09-2010
- ²⁵ : Dive into HTML5 | <http://diveintohtml5.org/forms.html#validation> | №9. A Form of Madness - Form Validation | Mark Pilgrim | | 24-09-2010
- ²⁶ : TED.com now available in HTML5, serving many mobile platforms, including iPhone, iPad | <http://www.wired.com/epicenter/2010/07/youtube-launches-new-html5-mobile-site/> | | 31-03-2010 | 24-09-2010
- ²⁷ : YouTube Launches New HTML5 Mobile Site | <http://www.wired.com/epicenter/2010/07/youtube-launches-new-html5-mobile-site/> | | Michael Calore | 08-07-2010 | 24-09-2010
- ²⁸ : HTML5 (including next generation additions still in development) Draft Standard | <http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html#the-canvas-element> | 4.8.11 The canvas element | Ian Hickson (Editor) | 10-09-2010 | 24-09-2010
- ²⁹ : HTML5 (including next generation additions still in development) Draft Standard | <http://www.whatwg.org/specs/web-apps/current-work/multipage/links.html#microdata> | 5 Microdata | Ian Hickson (Editor) | 10-09-2010 | 24-09-2010
- ³⁰ : RDFa Primer Bridging the Human and Data Webs | <http://www.w3.org/TR/xhtml-rdfa-primer/> | | Ben Adida & Mark Birbeck (Editors) | 14-10-2008 | 02-11-2010
- ³¹ : HTML5: Up and Running | Hfst. 10 : “Distributed,” “Extensibility,” and Other Fancy Words – Marking Up People | Blz. 168-174 | Mark Pilgrim | August 2010 – First Edition | O'Reilly Media, Inc. | 27-09-2010
- ³² : Web Storage Editor's Draft | <http://dev.w3.org/html5/webstorage/> | | Ian Hickson (Editor) | 25-09-2010 | 27-09-2010
- ³³ : HTTP cookie | http://en.wikipedia.org/wiki/HTTP_cookie | | 26-09-2010 | 27-09-2010
- ³⁴ : HTML5: Up and Running | Hfst. 7 : The Past, Present, and Future of Local Storage for Web Applications | Blz. 127 | Mark Pilgrim | August 2010 – First Edition | O'Reilly Media, Inc. | 27-09-2010
- ³⁵ : Hackers stelen Twitter-cookies via XSS-lek | http://www.security.nl/artikel/34382/1/Hackers_stelen_Twitter-cookies_via_XSS-lek.html | | 07-09-2010 | 27-09-2010
- ³⁶ : Web Storage Editor's Draft | <http://dev.w3.org/html5/webstorage/#implementation-risks> | 7.3 Implementation risks | Ian Hickson (Editor) | 25-09-2010 | 27-09-2010
- ³⁷ : HTML5 (including next generation additions still in development) Draft Standard | <http://www.whatwg.org/specs/web-apps/current-work/#offline> | 6.6.1 Introduction | Ian Hickson (Editor) | 26-09-2010 | 27-09-2010
- ³⁸ : Hixie's Natural Log | <http://ln.hixie.ch/?start=1115899732&count=1> | Beneath The Surface | Ian Hickson | 12-05-2005 | 11-10-2010
- ³⁹ : Geolocation API Specification W3C Candidate Recommendation | <http://www.w3.org/TR/geolocation-API/> | | Andrei Popescu (Editor) | 07-09-2010 | 27-09-2010
- ⁴⁰ : Geolocation API Specification W3C Candidate Recommendation | <http://www.w3.org/TR/geolocation-API/#usecases> | 6 Use-Cases and Requirements | Andrei Popescu (Editor) | 07-09-2010 | 27-09-2010



Hoofdstuk 3 : Ligt het succes van HTML5 bij het aanpassingsvermogen van de grote browsers?

In het vorige hoofdstuk zijn een groot aantal van de nieuwe functionaliteiten van *HTML5* doorgenomen. Op het moment van schrijven is het mogelijk om als ontwikkelaar de meeste specificaties al toe te passen. Er is echter één probleem: de browsers en dan met name de mate waarin browsers ondersteuning bieden voor *HTML5*. Hiervan zijn al voorbeelden langsgesproken in het vorige hoofdstuk, bijvoorbeeld: het `<meter>` element, dat alleen in *Chrome* wordt weergegeven als een balk (zie afbeelding 1). Alle overige browsers geven de harde waarden weer, die binnen het element staan (zie afbeelding 2).

Afbeelding 1		Afbeelding 2	
--------------	---	--------------	---

In dit hoofdstuk wordt er gekeken naar de nieuwe functionaliteiten, zoals besproken in hoofdstuk twee, en in hoeverre deze worden ondersteund. Dit staat echter in de lopende tekst en verspreid over het hele hoofdstuk. Voor een duidelijk overzicht kan er gekeken worden naar bijlage één; een grafische weergave van de ondersteuning per functionaliteit voor de vijf grote browsers (*Opera*, *Chrome*, *Safari*, *Firefox*, *Internet Explorer*). De makers van *Modernizr* (een *JavaScript* library waarmee de ondersteuning voor *CSS3* en *HTML5* functionaliteiten wordt bekeken - www.modernizr.com) hebben een overzicht gemaakt van oplossingen om de nieuwe functionaliteiten in oudere browsers te laten werken.¹ In dit hoofdstuk gaan we echter uit van de standaard ondersteuning die de browsers bieden.

3.1 - Semantische elementen

In het geval van het `<meter>` element moeten we echter rekening houden met het feit dat dit een bijzonder element is. In vergelijking tot het `<section>` element, maakt het `<meter>` element van harde waarden een grafische weergave (net als het `<progress>` element). Elementen die specifiek worden gebruikt om structuur te geven (semantische elementen) aan een document zullen weinig last hebben van problemen met compatibiliteit. Deze elementen hebben namelijk geen eigen opmaak (in tegenstelling tot `<p>` en `<h1>`) en veranderen ook de weergave van de inhoud niet. Er zijn een tweetal handelingen om er voor te zorgen dat de nieuwe elementen zich in elke browser gelijkwaardig gedragen.

3.1.1 - HTML5 Shiv

De nieuwe semantische elementen zijn voor alle browser onbekend. Alle browser, behalve *Internet Explorer (IE)*, accepteren dit zonder problemen te veroorzaken. IE staat echter niet toe om onbekende elementen op te maken (doormiddel van *CSS*) en in het *DOM* worden de elementen als lege nodes gezien, zonder children. Inhoud binnen het nieuwe element worden in de *DOM* buiten het element gezet als siblings.² Er zijn echter een aantal manieren om dit te omzeilen door de elementen zelf 'te maken' met behulp van *JavaScript*:

```
<script> document.createElement("article"); </script>
```

Door deze regel code zal IE het element (in dit geval het `<article>` element) herkennen en wordt het gezien als elk ander element. Via *CSS* kan de opmaak veranderd worden en in de *DOM* zal het element en de children goed worden weergegeven. Het is uiteraard niet gewenst om in elk document een lijst te hebben met alle nieuwe elementen van *HTML5*. Daarom is er een script beschikbaar via *Google* project hosting³ dat alle elementen creëert, zodat IE deze herkent. Nu kan er met één regel code bereikt worden, wat anders twintig regels had gekost. Er wordt hier gewerkt met een conditionele opmerking, zodat het stuk code alleen uitgevoerd wordt als er in IE gewerkt wordt.



```
<!--[if lt IE 9]> <script src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script> <![endif]-->
```

Door deze regel toe te voegen in het <head> element worden de nieuwe semantische elementen (zoals besproken in hoofdstuk twee) volledig ondersteund in alle browsers. Het is uiteraard aan de ontwikkelaar om te beslissen of hij/zij al gebruik wil maken van de elementen. Het levert wel de voordelen op, die de *Web Hypertext Application Technology Working Group (WHATWG)* voor ogen had: het document wordt overzichtelijker en de semantische waarde van het document vergroot aanzienlijk.

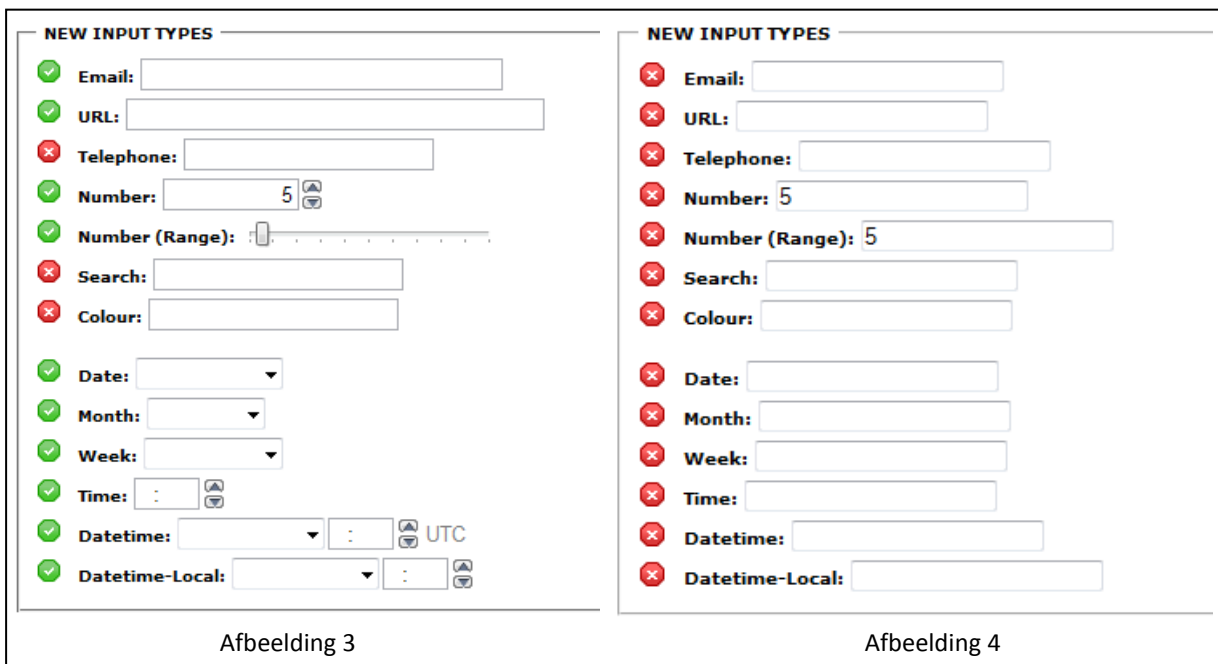
3.1.2 - Display: Block

Elementen, die bekend zijn bij de browser, hebben veelal een standaard opmaak. Het kan verschillen per browser, maar in ieder geval geeft *Firefox* de opmaak *display: block* standaard mee aan het <div> element. De nieuwe semantische elementen nemen in een aantal gevallen de rol van het <div> element over: ze zouden daarom dezelfde opmaak als het <div> element moeten hebben. Echter, aangezien ze onbekend zijn voor de browsers zal de ontwikkelaar dit zelf moeten aangeven. Zolang de elementen niet op deze manier zijn opgemaakt, zal de overige opmaak niet worden toegepast op de elementen.

```
section, header, nav, aside, article, footer
{
    display: block;
}
```

3.2 - Forms

We zullen later in het hoofdstuk zien dat ontwikkelaars vaak zelf een alternatief moeten aanbieden, om de ervaring in elke browser zo goed mogelijk hetzelfde te krijgen. In het geval van forms ligt dit echter iets anders. Zo maken de browsers van de nieuwe input types (zie hoofdstuk 2), die niet ondersteund worden standaard een normaal invoerveld. Als voorbeeld de testpagina waar in het vorige hoofdstuk mee is gewerkt. Op afbeelding 3 is het formulier te zien in *Opera* (biedt de beste ondersteuning voor *HTML5* forms): alle input types worden op de manier weergegeven, zoals de *WHATWG* voor ogen heeft. Op afbeelding 4 is hetzelfde formulier te zien in *Firefox*: hier is duidelijk te zien dat de invoervelden van input type number t/m datetime-local omgezet zijn naar standaard invoervelden. De iconen voor de invoervelden geven aan of de browser het input typ ondersteunt.





Kortom, de nieuwe input types kunnen gebruikt worden, ook al worden ze nog niet door alle browsers ondersteund. Op een puur visueel vlak is er altijd een alternatief, dat wordt aangeboden vanuit de browser. Waar wel een probleem in zit is het valideren van de input. Als voorbeeld nemen we het input type `number`: *Opera* geeft een invoerveld weer met een tweetal pijlen. Eén van deze pijlen (omhoog bij max en omlaag bij min) zal niet klikbaar zijn zodra deze waarde is bereikt. Als er handmatig een getal boven de 10 wordt ingevuld, zal er tijdens het verzenden een foutmelding worden getoond.⁴ [in *Opera*] In *Firefox* is er alleen de mogelijkheid om handmatig de waarde in te vullen. Als er nu een waarde boven de 10 wordt ingevuld en verzonden, zal er geen validatie plaatsvinden.⁴ [in *Firefox*] Hiervoor zou de ontwikkelaar zelf een stuk code moeten schrijven om de validatie te verrichten.

Daarnaast hebben we nog een aantal nieuwe attributen, die meegeleverd worden in de form specificaties: `placeholder`, `autofocus`, `required`, `pattern`, `list` en `multiple`. Het is mogelijk om deze attributen al te gebruiken, alleen passen niet alle browsers de functionaliteit toe. Maar voor elk attribuut kan er een alternatief aangeboden worden. Zo worden op dit ogenblik de eerste vier afgehandeld met *JavaScript*. Kan `list` worden nagebootst door zowel een invoerveld als een drop-down lijst aan te bieden. In het geval van `multiple` kan het volgende alternatief geboden worden; zodra er een invoerveld is ingevuld, wordt er een nieuw invoerveld toegevoegd aan het form (met behulp van *JavaScript*).

Net als semantische elementen kunnen de nieuwe functionaliteiten van forms al toegepast worden. Er zijn echter wel drie grote aandachtspunten waar rekening mee gehouden moet worden:

- Het valideren van ingevoerde data moet door een eigen gemaakt stuk code gedaan worden. Uitzondering op deze regel is *Opera*, waar in ieder geval de input velden *email*, *url* en *number* automatisch worden gevalideerd.
- Functionaliteiten van attributen vereisen op dit moment nog een alternatief voor browsers die het attribuut niet ondersteunen. Aangezien de attributen juist als doel hebben om deze alternatieven te vervangen, zou dit geen probleem mogen zijn.
- De ondersteuning voor input types, elementen en attributen verschilt per browser. Zo biedt *Opera* de meeste ondersteuning met 73% werkende functionaliteiten, terwijl *Internet Explorer 8* geen enkele ondersteuning biedt. *Firefox* scoort niet veel beter met één werkende functionaliteit.

Het is vooral het laatste punt dat een doorslaggevende factor speelt in de vraag of men op dit moment gebruik wil maken van *HTML5* forms. *Internet Explorer* en *Firefox* zijn de twee grootste browsers⁴ en voor deze browsers (en *Safari*) levert een *HTML5* form hetzelfde resultaat op als een *XHTML* form. Een argument om wel *HTML5* forms te gebruiken is dat de website klaar zal zijn voor de toekomst. Zodra de desbetreffende browsers wel ondersteuning bieden, hoeft de website en/of het formulier niet herschreven te worden.

3.3 - Video, audio en canvas(text)

Als er in een gesprek *HTML5* ten sprake komt, zullen de eerste woorden hoogstwaarschijnlijk over `canvas` of `video` gaan. Het is dan ook allesbehalve vreemd dat deze functionaliteiten in alle browsers werken. Vooral `canvas`, het stokpaardje van de WHATWG, was één van de eerste vernieuwingen die in alle browsers ondersteund werd. Er moet wel opgemerkt worden dat *Internet Explorer* deze elementen pas sinds IE9 ondersteund. IE9 is op het moment van schrijven nog in de bèta fase van ontwikkeling. Het `<canvas>` element zal verder niet worden toegelicht aangezien deze door alle browsers ondersteund wordt.

Het `<video>` en `<audio>` element zijn echter wel belangrijk om toe te lichten. Hier zit namelijk een probleem in dat veroorzaakt wordt door een bepaald onderdeel van deze mediabronnen: de codecs. Een videobestand bestaat uit een tweetal codecs (één audio en één video), samengevoegd in een container. Voorbeeld: de Theora video codec en de Vorbis audio codec maken samen de Ogg container. In de *HTML5* specificatie wordt niet aangegeven welke video codecs en containers er gebruikt moeten worden en daardoor wordt de keuze bij de browsers gelegd.



Op dit moment zijn er een drietal containers, die gebruikt worden in *HTML5*: Ogg, MP4 en WebM:

- Ogg: deze container bestaat uit de Theora video codec en de Vorbis audio codec.
- MP4: deze container bestaat uit de H.264 video codec en AAC audio codec.
- WebM: deze container bestaat uit de VP8 video codec en Vorbis audio codec.

Er is geen enkele browser die al deze containers ondersteund en men verwacht dat dit in de komende jaren zo zal blijven. Het is tevens zo dat WebM nog helemaal niet ondersteund wordt, maar dat er wel verwacht wordt dat dit binnen het jaar zal veranderen. *Internet Explorer*, *Firefox*, *Chrome*, *Opera* en de browser op de *Android* zouden er dan gebruik van kunnen maken. Als bovenstaande browsers daadwerkelijk de ondersteuning implementeren zal het de best ondersteunde container worden. Maar tot het zover is (en wellicht daarna ook nog) zullen ontwikkelaars video's in meerdere vormen moeten aanbieden om zo alle browsers van beeld en geluid te voorzien. Zie hoofdstuk 2.4.1 voor een voorbeeld.

Ook het audio element heeft nu nog last van bovenstaand probleem, aangezien de specificaties niet aangeven welke containers er gebruikt moeten worden. De keuze ligt daarom nog bij de browsers en die bieden ondersteuning voor een viertal containers: Ogg, MP3, Wav en ACC. Om iedere eindgebruiker het audio bestand te laten beluisteren zou op dit moment het bestand in minimaal twee formaten aangeboden moeten worden. Zie bijlage 1 voor de ondersteuning per browser.

3.4 - Microdata

Microdata is een verhaal met twee kanten: het komt er op neer dat Microdata een selectie van zelfgemaakte semantische elementen is. En zoals we eerder in dit hoofdstuk gezien hebben zullen elementen, die alleen uitwerking hebben op de structuur van een document, geen problemen moeten opleveren. En nu de andere kant van het verhaal: het is echter wel zo dat geen enkele browser microdata ondersteund.⁶ Waar deze tegenstrijdigheid op neerkomt is dat de browser de elementen ziet, maar niet weet wat er mee gedaan moet/kan worden. Er is nog niks in de browser toegevoegd dat zegt: "haal de microdata op van de site en sla de evenementen op in mijn kalender."

Daarnaast hebben we in hoofdstuk twee gezien dat microdata ook gebruikt kan worden door zoekmachines. Aangezien werknemers van Google betrokken zijn bij de ontwikkeling van *HTML5* is het niet wonderlijk dat *HTML5* functionaliteiten in de producten van Google terecht komen. Uit een nieuwsbericht uit maart van dit jaar is te halen dat Google al microdata gebruikt: "*By using microdata markup in your web pages, you can specify reviews, people profiles, or events information on your web pages that Google may use to improve the presentation of your pages in Google search results.*"⁷

Er is geen reden om niet met microdata te werken. Op het moment voegt het nog geen functionaliteit toe vanuit de browser, maar in relatie tot zoekmachines kan het net dat beetje extra opleveren. Het zoekresultaat van *Hoppinger* (met adresgegevens en een recensie) kan dan net iets aantrekkelijker lijken dan andere internetspecialisten.

3.5 - Web Storage, Offline Storage, Drag and Drop en Geolocation

Deze functionaliteiten worden beschouwd als onderdelen van de oorspronkelijke '*Web Apps 1.0*' specificaties. Opvallend genoeg zijn de browsers vooral werkzaam geweest om de functionaliteiten van deze specificaties te ondersteunen. Dit is duidelijk te merken als er gekeken wordt naar de mate van ondersteuning voor de vernieuwingen uit bovenstaande kop:⁸

- Web Storage is volledig ondersteund.
- Offline Storage wordt door alle browser ondersteund, behalve IE.
- Drag and Drop wordt door alle browsers ondersteund, behalve *Opera*.
- Geolocation (geen *HTML5*) wordt niet ondersteund door IE en *Safari 4*. Overige browsers ondersteunen deze API wel.



3.6 - Conclusie

Alle functionaliteiten zoals die in de *HTML5* specificaties zijn opgenomen kunnen op dit moment al gebruikt worden door web ontwikkelaars. Echter kunnen niet alle eindgebruikers deze functionaliteiten (optimaal) ervaren. De reden hiervoor is de mate waarin browsers de *HTML5* specificaties ondersteunen. De populaire en alom bekende `<canvas>` en `<video>` elementen worden sinds de bèta van IE9 in elke browser ondersteunt. Nieuwe input types en attributen worden op dit moment nog voornamelijk in *Opera* ondersteunt.

Deze wisselvalligheid zorgt ervoor dat web ontwikkelaars continu moeten nadenken over alternatieven: graceful degradation. Althans, als men wil dat de website in de vijf grote browsers dezelfde functionaliteiten hebben. Het bedenken en implementeren van alternatieven kost extra tijd en mankracht. Binnen een bedrijfsomgeving komt hier ook nog eens extra kosten bij. Wanneer nieuwe browserversies betere ondersteuning bieden en de alternatieven niet meer nodig zijn, is al het extra werk voor niks geweest. Een uitzondering hierop is wanneer de ontwerper voordeel van deze werkzaamheden heeft gehad. Het was daarnaast de bedoeling dat deze alternatieven juist onnodig werden met een aantal simpele elementen/attributen.

Daarnaast kan volgens de volgende bron *HTML5* pas gebruikt worden (op grote schaal) wanneer alle oude browsers niet meer/heel weinig worden gebruikt: *"So we have to wait for IE6, IE7, Safari 3, Firefox 2 & 3, all to die off before [using any new element from] HTML5 is viable on the corporate web. We're going to have to wait for browsers that aren't even out yet to become statistically irrelevant before we can use [the new elements of] HTML5 without either duplicating the site in XHTML or throwing away customers/visitors/revenue."*⁹ In de laatste zin wordt gezegd dat wanneer toekomstige browsers in de vergetelheid raken men pas kan beginnen met *HTML5*, zonder een alternatief in *XHTML* te maken.

Uiteraard kleven er altijd nadelen aan nieuwe ontwikkelingen. Dit wordt in het geval van *HTML5* versterkt aangezien het product al beschikbaar is tijdens de ontwikkeling. Het is voor de browsers daardoor al wel mogelijk om ondersteuning in te bouwen aan de hand van de specificaties. Het is op dit moment echter nog steeds mogelijk dat een specificatie kan veranderen. Het compleet verwijderen van een specificatie of een totaal andere functie aan een element geven zal hoogstwaarschijnlijk niet meer voorkomen.

Zoals alle showcase sites van de grote browsers ons laten zien is *HTML5* een verrijking voor web development. Als men eenmaal grootschalig met *HTML5* gaat werken is er geen weg meer terug. Maar om de ervaring van deze nieuwe mogelijkheid zo optimaal mogen aan te bieden ligt de sleutel bij de bedrijven die de browsers maken. Zonder ondersteuning van de browsers kunnen de bedachte specificaties nooit gezien en/of ervaren worden. Het antwoord op deze deelvraag is: ja, het succes van *HTML5* ligt bij het aanpassingsvermogen van de browsers.

Bronnen

¹: HTML5 Cross browser Polyfills | <http://github.com/Modernizr/Modernizr/wiki/HTML5-Cross-browser-Polyfills> | Edited and maintained by Paul Irish | 18-10-2010 | 18-20-2010

²: HTML5: Up and Running | Hfst. 3 : What Does It All mean? – A Long Digression into How Browsers Handle Unknown Elements | Blz. 42-43 | Mark Pilgrim | August 2010 – First Edition | O'Reilly Media, Inc. | 28-09-2010

³: HTML5 IE enabling script | <http://code.google.com/p/html5shiv/> | Remy Sharp | 08-09-2010 (Last Update) | 28-09-2010

⁴: HTML5 New Input Types | http://www.w3schools.com/html5/tryit.asp?filename=tryhtml5_form_number | TryIt Editor – Number input type | 28-09-2010

⁵: Browser Statistics | http://www.w3schools.com/browsers/browsers_stats.asp | ??-08-2010 | 04-10-2010

⁶: The HTML5 Test – How well does your browser support HTML5? | <http://html5test.com/> | Niels Leenheer | 28-09-2010

⁷: Microdata support for Rich Snippets | <http://googlewebmastercentral.blogspot.com/2010/03/microdata-support-for-rich-snippets.html> | Google medewerkers | 11-03-2010 | 28-09-2010

⁸: HTML5 & CSS3 Support | <http://www.findmebyip.com/litmus#target-selector> | HTML5 Web Applications | 28-09-2010.

⁹: HTML5 is not backward compatible | <http://mattwilcox.net/archive/entry/id/957/> | And by way of clarifying why HTML5 is a dead donkey for the foreseeable future | Matt Wilcox | 24-09-2008 | 28-09-2010



Hoofdstuk 4: Zal HTML5 voordelen opleveren op het gebied van SEO in vergelijking met XHTML?

Google is ooit begonnen als een (kleinschalige) zoekmachine, maar is sindsdien uitgegroeid tot de nummer één zoekmachine.¹ Web ontwikkelaars houden daarom tijdens het ontwikkelen en na ontwikkeling rekening met de kracht van *Google*. Het belangrijkste van een website is dat deze gevonden kan worden. En dit is niet gemakkelijk in een wereld met miljarden websites. Om dit voor elkaar te krijgen zijn er technieken bedacht, die onder de noemer Search Engine Optimization (SEO)² zijn geschaard. Hiermee kunnen web ontwikkelaars hun producten zo ontwikkelen dat het voor zoekmachines gemakkelijker is om een profiel van de website te maken.

Zoals in hoofdstuk één te lezen is, is de *Web Hypertext Application Technology Working Group (WHATWG)* een gezamenlijk initiatief van *Mozilla Foundation* en *Opera Software*. Echter, dit was de situatie in 2004 toen de WHATWG werd opgericht. Sindsdien zijn er nieuwe mensen op het toneel verschenen en zijn sommigen van werkgever veranderd. Een goed voorbeeld hiervan is *Ian Hickson*, editor van de *HTML5* specificaties. In 2004 werkte hij nog voor *Opera Software*, maar maakte in 2005 de overstap naar *Google*. Sinds 2005 is *Google* steeds meer betrokken geraakt in de ontwikkeling en implementatie van *HTML5*. De vraag is in hoeverre *HTML5* is gemaakt vóór *Google*. In dit hoofdstuk zal gekeken worden hoe nieuwe functionaliteiten voordelen opleveren voor SEO en zoekmachines in het algemeen.

4.1 - Page segmentation

*“Essentially page segmentation is when a search engine looks to break a given web page down into its component parts. They could analyze a web page and assign various relevance or importance scoring for different regions of a page.”*³ Waar page segmentation op neer komt is dat een website uit elkaar wordt gehaald en elk stukje wordt gescand naar de inhoud: wat is relevant of belangrijk voor de zoekmachine om te weten. Aangezien *Google* de meest relevante resultaten als eerste weergeeft, is het belangrijk om het voor zoekmachines makkelijk te maken om de website te segmenteren.

Het is moeilijk en/of onmogelijk voor zoekmachines om uit een `<div>` element (met of zonder id of class naam) de context van de inhoud te halen. Daarom wordt er tijdens het segmenteren van een webpagina gekeken naar bepaalde (visuele) patronen om in te schatten waar de belangrijke content staat. Deze patronen zijn⁴:

- Het aantal afbeeldingen dat gebruikt wordt binnen een block element.
- Het formaat van afbeeldingen.
- Het aantal links.
- De lengte van een link.
- Het aantal woorden.
- Lengte van een formulier.
- Elementen voor tekst opmaak (``, `<text>`, `<i>`, ``).
- Andere code elementen (`<table>`, `<p>`, `<hr>`, ``, `<td>`).
- De achtergrond kleur van een node (of child node).
- Het formaat en positie van een block element.

Deze aanpak vereist veel denkwerk van zoekmachines: elke regel inhoud moet gecontroleerd worden om te kunnen bepalen waar de belangrijkste inhoud van de betreffende pagina kan zitten. Ook bestaat er de kans dat de zoekmachine een fout maakt: de patronen zijn gebaseerd op websites met een standaard opmaak. Als een website hier sterk van afwijkt kunnen de resultaten van de analyse sterk verschillen met de realiteit.

In hoeverre kan *HTML5* page segmentation makkelijk maken voor zoekmachines? Over het algemeen zal het niet langer meer nodig zijn om met de bovengenoemde patronen te werken. Er kan nu immers met de nieuwe semantische elementen worden aangegeven waar alle informatie staat. De belangrijkste informatie (nieuwsartikel of projectinformatie) zal hoogstwaarschijnlijk opgenomen zijn in een `<article>` element. Het zal een stuk makkelijker worden voor zoekmachines om een webpagina te segmenteren, aangezien elk element een specifieke rol vervult.



4.2 - Zoekmachines beïnvloeden

Het draait bij zoekmachines allemaal om relevante informatie. Dit is de informatie waar zoekmachines mee willen werken, en is tevens hetgeen dat gebruikers voorgeschoteld willen krijgen. Eén methode om de zoekmachine te laten weten waarom een webpagina relevant zou kunnen zijn, is door keywords aan te geven in het <meta> element. Meta informatie wordt gebruikt om machines informatie over de website en webpagina aan te reiken. Hoe meer waarden je hierin plaatst, hoe relevanter de webpagina is en hoe hoger deze in de zoekresultaten staat. Deze manier van zoekmachine beïnvloeding werkt echter niet meer, aangezien *Google* in 2009 heeft laten weten niet de inhoud van het <meta> element te gebruiken om de relevantie te meten: *"De reden dat Google de meta keywords tag al jaren niet meer gebruikt [...] is eenvoudig: te veel mensen hebben de meta keywords tag misbruikt. Zo plaatste men te veel irrelevante zoektermen in de meta keywords tag om zoekmachines te misleiden."*⁵

Overige zoekmachines gebruiken de meta keywords nog wel, maar ook bij deze heeft het weinig invloed op de ranking. Er zijn echter nog meer methoden om de zoekmachine te beïnvloeden. En in *HTML5* zijn er hier een tweetal bijgekomen: het <time> element en microdata.

4.2.1 - <time> element

Het komt nogal eens voor dat er op een zoekterm nieuwsberichten opduiken die allang gedateerd zijn: het is goed dat men deze nog terug kan vinden, maar de gebruiker heeft er vaak niets aan. Om dit soort situaties te voorkomen is er in *HTML5* een nieuwe functionaliteit toegevoegd: het <time> element. Met behulp van het <time> element (zie gebruik in hoofdstuk 2) kan de ontwikkelaar tijdsaanduidingen op een webpagina leesbaarder maken voor machines.

Onder andere zoekmachines kunnen de inhoud van het <time> element gebruiken om de relevantie van een resultaat te beoordelen. Hoe nieuwer het artikel, hoe relevanter het moet zijn. Het logische is dat de nieuwste bronnen hoger in de zoekresultaten worden weergegeven. Mogelijke verwarringen, zoals in de volgende bron, zouden daarmee voorkomen kunnen worden: *"[...] in the recent heavy snow in the UK, I searched for school closures in my town and discovered my kids' school was closed. After receiving a phone call from the school asking me where my kids were, I re-examined the webpage. In small print at the bottom of the page were the words "Published 5 January 2008". [...]"*⁶

4.2.2 - Microdata

Met behulp van microdata kunnen ontwikkelaars aangeven welke gegevens belangrijk zijn op een webpagina. Deze data wordt opgeslagen in een 'dossier' (zie hoofdstuk 2) en kan gebruikt worden door zoekmachines om zoekresultaten mee uit te breiden. Je kunt de zoekmachine dus beïnvloeden in wat belangrijk is om weer te geven in het zoekresultaat. Op het ogenblik zijn er een viertal soorten 'dossiers' ondersteund: individueel (afbeelding 1), bedrijf, evenement (afbeelding 2) en een beoordeling (afbeelding 3).⁷ De regels tekst in het lichtgrijs zijn direct overgenomen uit de microdata.

<p>Afbeelding 1</p>	<p>Google search preview</p> <p>John Cox Louisville KY - writer Excerpt from the page will show up here. Excerpt from the page will show up here. Excerpt from the page will show up here. Excerpt from the page will show up here. wyome.com/microdata.php - Cached - Similar pages</p>
<p>Afbeelding 2</p>	<p>Google search preview</p> <p>The Devil Makes Three Excerpt from the page will show up here. Excerpt from the page will show up here. Excerpt from the page will show up here. Excerpt from the page will show up here. The Devil Makes Three Sun, Jun 13 Bonnaroo wyome.com/microdata4.php - Cached - Similar pages</p>
<p>Afbeelding 3</p>	<p>Google search preview</p> <p>R-195F Review ★★★★★ Review by John Cox - Jun 10, 2010 Excerpt from the page will show up here. Excerpt from the page will show up here. Excerpt from the page will show up here. Excerpt from the page will show up here. wyome.com/microdata3.php - Cached - Similar pages</p>



Het zijn kleine toevoegingen, maar ze kunnen desalniettemin het verschil maken. Een voorbeeld: Stel dat een bedrijf gespecialiseerd is in het beoordelen van webspecialisten in Rotterdam. Al de beoordelingspagina's zijn opgemaakt met een <time> element op de publicatiedatum en de inhoud opgenomen in microdata. We gaan ervan uit dat een bedrijf een nieuwe website wil en zoekt op *Google* naar 'beste webbureaus omgeving Rotterdam'. In de zoekresultaat staan een tiental resultaten (allemaal beoordeeld in de afgelopen maand), weergegeven zoals afbeelding 3. Voorheen werd alleen de titel, inhoud en link van de beoordeling weergegeven. Om van elke beoordeling het resultaat te zien, moest elke pagina afzonderlijk geopend worden. In dit geval kan de gebruiker direct de beoordeling aanschouwen en dat is waar het uiteindelijk om draait bij beoordelingen. Het kan de gebruiker een hoop werk uit handen nemen.

4.3 - Zoekmachines en video's

Een aantal zoekmachines bieden naast de standaard zoekresultaten ook resultaten in de vorm van afbeeldingen en video's aan. Er zit echter een groot verschil tussen deze twee vormen en dat zit hem voornamelijk in de bron van het resultaat. Zo zijn afbeeldingen van alle websites geplukt, die ook in de standaard resultaten zouden kunnen staan. Video's zijn echter van websites afgenomen, die één specifiek doel dienen; het uploaden en weergeven van video's. Voorbeelden hiervan zijn: *YouTube*, *Metacafe* en *Dailymotion*.

Dit verschijnsel komt doordat 'normale' websites tot nu toe geen video's in de browser konden afspelen, zonder gebruik te maken van externe plugins en/of videospelers. De meeste plugins (zoals *Flash*) stoppen het bronbestand in een container, zodat deze wel af te spelen is maar niet te indexeren door zoekmachines. Het probleem hierbij is dat de locatie van de bron niet meer zichtbaar is voor de browser: *"Because Flash obfuscates text [in dit geval video] inside SWF files, search engines have no means of reading and indexing the content, because they rely on HTML text to not only see the actual words, but also how those words are organized into a meaningful structure, like headers, paragraphs, and links."*⁸ Veel video's die door 'normale' websites worden aangeboden staan in een onleesbare vorm voor zoekmachines.

Dit zal echter drastisch gaan veranderen met *HTML5* en het <video> element. Hier worden de video's direct in geplaatst en is te vergelijken met een element. Het bronbestand staat hier namelijk ook aangegeven in de source. En zoals we in de voorgaande quote hebben kunnen zien is dit leesbaar voor zoekmachines. Video's zullen weldra op dezelfde manier door zoekmachines worden verwerkt als afbeeldingen. Het zal een stijging in het aantal zoekresultaten opleveren, aangezien er steeds meer video's geïndexeerd kunnen worden.

4.4 - Conclusie

Zoekmachines werken met crawlers of spiders om websites te doorlopen en te ontleden naar relevante inhoud. Tot nu toe moesten hier moeilijke algoritmes aan te pas komen om deze content uit een webpagina te filteren. Met de nieuwe semantische elementen kan de ontwikkelaar zelf aangeven waar de relevante informatie staat. Zo zal de belangrijkste data van de webpagina binnen een <article> element staan. Meta data over dit artikel kan bijvoorbeeld in een <header> en/of <footer> binnen het artikel staan. Voor meer informatie over de semantische elementen kan er teruggekeken worden naar hoofdstuk 2.2. Het wordt de zoekmachines dus een stuk gemakkelijker gemaakt en het zal de foutmarge doen dalen.

Daarnaast maken een aantal functionaliteiten het mogelijk om zoekmachines te beïnvloeden. Dit kan met het <time> element gedaan worden, waarmee onder andere publicatiedata kan worden aangegeven. Het voordeel van dit element is dat de inhoud leesbaar is voor machines. Deze kunnen de inhoud gebruiken om hiermee de relevantie te bepalen: nieuwe artikelen hebben meer relevantie dan oudere. Daarnaast kunnen ontwikkelaars met behulp van microdata aangeven welke informatie belangrijk kan zijn om weer te geven in het zoekresultaat. Bijvoorbeeld een adres bij een bedrijf, beoordeling bij een review of evenementen bij een evenementenhal.



Als laatste is er gekeken naar het nieuwe <video> element dat, in tegenstelling tot video's in een *Flash* container, wel geïndexeerd kan worden door zoekmachines. Bij *Flash* kon deze indexatie niet plaatsvinden, aangezien zoekmachines alleen met *HTML* kunnen werken. Het bronbestand en uitleg van de inhoud staan nu direct in het element en kunnen daarom wel worden opgevangen door zoekmachines.

In de wereld van zoekmachines draait het allemaal om de content en de nieuwe semantische elementen zorgen dat deze beter te bereiken is voor zoekmachines. Daarom zal voornamelijk hier voordeel uitgehaald kunnen worden op het gebied van SEO. Er mag echter niet worden onderschat dat microdata en het <time> element er voor kunnen zorgen dat een website hoger in de zoekresultaten kan komen te staan. De eindconclusie is als volgt: een aantal van de functionaliteiten in *HTML5* leveren inderdaad voordeel op, op het gebied van SEO.

Bronnen

- ¹ : Top Ten Search Engines - Top 10 SEs | <http://www.seoconsultants.com/search-engines/> | Search Engine Statistics - What Market Share? | | 08-09-2010 | 30-09-2010
- ² : SEO Starter Guide | <http://www.google.com/webmasters/docs/search-engine-optimization-starter-guide.pdf> | | Google | 28-09-2010 | 30-09-2010
- ³ : SEO implications of Page Segmentation concepts | <http://www.huomah.com/Search-Engines/Search-Engine-Optimization/SEO-implications-of-Page-Segmentation-concepts.html> | How search engines could get granular | | 08-01-2009 | 30-09-2010
- ⁴ : SEO implications of Page Segmentation concepts | <http://www.huomah.com/Search-Engines/Search-Engine-Optimization/SEO-implications-of-Page-Segmentation-concepts.html> | Segmenting the page | | 08-01-2009 | 30-09-2010
- ⁵ : Google: meta keywords & description niet gebruikt in ranking | <http://www.edwords.nl/2009/09/21/google-meta-keywords-meta-description-tag-niet-gebruikt-ranking/> | Waarom negeert Google de meta keywords tag? | | 21-09-2009 | 30-09-2010
- ⁶ : The time element (and microformats) | <http://html5doctor.com/the-time-element/> | Where could time be used? | Bruce Lawson | 09-02-2010 | 30-09-2010
- ⁷ : HTML5 Microdata: Welcome to the Machine | <http://net.tutsplus.com/tutorials/html-css-techniques/html5-microdata-welcome-to-the-machine/> | | John Cox | 14-06-2010 | 30-09-2010
- ⁸ : SEO Debate: HTML vs. Flash | <http://www.rickydeez.com/web-design/seo-debate-html-vs-flash/> | SEO / Search Engine Hindrances | | Circa 2008 | 30-09-2010



Hoofdstuk 5: Is HTML5 een goed alternatief om Flash mee te vervangen?

Aan het begin van 2010 werd door *Apple* de *iPad* geïntroduceerd aan het publiek. Het bleek dat, net als de *iPhones* en *iPods*, er geen ondersteuning voor *Flash* op zou zitten. Volgens *Apple* kan *Flash* namelijk nooit optimaal werken op mobiele toestellen, omdat deze hier niet voor gemaakt zijn. *Flash* zou veel (reken)kracht nodig hebben om te draaien en dat is nu juist wat mobiele toestellen niet hebben. Bovendien zou touchscreen niet goed samenwerken met *Flash* dat gebouwd is voor computers met muisbesturing. Het ongenoegen of bovenstaande zaken kwamen in april 2010 tot uiting toen *Steve Jobs* een brief met de titel '*Thoughts on Flash*' (zie bijlage drie) schreef. Hierin concludeerde *Apple*, één van de pioniers in de ontwikkeling van *HTML5*, dat *HTML5* de toekomst zal zijn voor mobiele en desktop platformen. Maar in hoeverre kan *Flash* vervangen worden door *HTML5*? In dit hoofdstuk zal er getracht worden om antwoord op deze vraag te geven.

5.1 - Browser plug-ins

Eén van de speerpunten van *HTML5* is dat het third party browser plug-ins onnodig maakt: *"The new standard incorporates features like video playback [...] that have been previously dependent on third-party browser plug-ins such as Adobe Flash, Microsoft Silverlight, [...]"*² Om dit te realiseren moeten er in *HTML5* functionaliteiten worden opgenomen die als alternatief kunnen dienen.

Met third-party browser plug-ins zit het grootste probleem in de kleine ongemakken dat het oplevert:

- Een video wil niet afspelen, omdat de juiste plug-in versie niet is geïnstalleerd.
- Sommige video's worden in één extensie aangeboden (bijvoorbeeld alleen in *Silverlight*) en het duurt even voordat de juiste plug-in is geïnstalleerd.
- Elk geïnstalleerde plug-in neemt ruimte in op de gebruikers computer.
- Continu meldingen krijgen wanneer er een nieuwe versie van de plugin beschikbaar is.
- De noodzaak om continu de nieuwste versie van de plug-in te installeren.
- Op elke computer moeten de plug-ins worden geïnstalleerd: twee computers, twee keer installeren en/of updaten.

5.1.1 - Adobe Flash

*"Adobe® Flash® Player is a cross-platform browser-based application runtime that delivers uncompromised viewing of expressive applications, content, and videos across screens and browsers."*³ Een aantal functionaliteiten die tegenwoordig door *Flash* worden afgehandeld zijn games, rijkelijk geanimeerde websites, geanimeerde reclamebanners, audiospelers en videospelers. In *HTML5* zijn een drietal elementen die gebruikt kunnen worden om deze functionaliteiten over te nemen: `<canvas>`, `<video>` en `<audio>`.

Het `<canvas>` element kan gebruikt worden om te animeren en zou de rol over kunnen nemen als het om games, rijkelijk geanimeerde websites en reclamebanners gaat. Het `<video>` en `<audio>` element kunnen gebruikt worden om een alternatief te bieden voor de Search Engine Optimization (SEO) onvriendelijke video- en audiospelers (zie hoofdstuk 4.3). Het alternatief biedt echter een probleem. Volgens de chief technical officer van *Adobe* worden op dit moment 75% van de video's afgespeeld met behulp van *Flash*.⁴ Dit wil men gaan vervangen met een videospeler, dat op elk platform kan werken maar niet met alle videobestanden overweg kan. Dit ligt niet zozeer aan *HTML5*, maar meer aan de mate waarin de browsers ondersteuning bieden (zie hoofdstuk 3.3 en bijlage 2).



5.1.2 - Alternatieven

Het komt er vaak op neer dat er third-party browser plug-ins nodig zijn voor het afspelen van video- en/of audiobestanden (bv. *Silverlight* en *QuickTime*). Door het originele bestand in een container te plaatsen wordt het pas mogelijk om de video op een webpagina af te spelen. We hebben in hoofdstuk 2.4 en 4.3 gezien dat deze container in *HTML5* verdwijnt en dat video- en audiobestanden rechtstreeks kunnen worden afgespeeld. Plugin-ins om dit mogelijk te maken zouden dus niet meer nodig zijn.

HTML en *XHTML* zijn niet gemaakt om animaties mee te maken. Hiervoor zijn andere technieken bedacht zoals *Java*, *Flash*, *Silverlight* en *JavaScript* (bv. via *jQuery* of *MooTools*). Met de eerste drie kunnen spellen en/of rijkelijk geanimeerd websites worden gemaakt. De gebruiker moet echter wel software installeren om het resultaat te zien (zie 6.1). In *HTML5* zouden animaties met behulp van het `<canvas>` element en de achterliggende canvas API gemaakt kunnen worden. Men moet er echter wel rekening mee houden dat (in ieder geval) *Flash* al jaren ontwikkeld en geoptimaliseerd is voor dit soort taken. Het is daarom nog niet mogelijk om te zeggen of alle functionaliteiten en het gemak van *Flash* (het animeren met behulp van de tijdlijn) geëvenaard kunnen worden door *HTML5*.

5.2 - Prestaties

5.2.1 - CPU gebruik

Sinds de rel tussen *Adobe* en *Apple* (en daarmee *Flash* vs. *HTML5*) is er steeds geroepen dat *Flash* een grootgebruiker van de CPU is. Daarom zou het nadelig zijn voor mobiele toestellen die uiteraard minder geheugen hebben dan desktop platformen (waar *Flash* oorspronkelijk voor is gemaakt). Tevens zou *Flash* ervoor zorgen dat de batterij van mobiele telefoons nog sneller leeg zijn. Op mobiele platformen zou *HTML5* dus beter presteren en daarom worden video's op *YouTube* nu ook in *HTML5* aangeboden. Vooral voor mobiele platformen die geen *Flash* ondersteunen zou dit een uitkomst moeten gaan bieden.

De implementatie van *HTML5* elementen in *YouTube* leverde een ideale testcase op: er worden immers twee identieke bestanden aangeboden, één in *Flash* en de ander in *HTML5*. In februari van dit jaar heeft een onderzoek plaatsgevonden waarin gekeken werd naar het CPU gebruik van de *Flash* videospeler en de *HTML5* videospeler. De conclusie die hieruit kon worden gehaald was niet als verwacht: de één is niet duidelijk beter dan de ander. Het verschilt namelijk per besturingssysteem, browser en/of *Flash*-versie.⁵

5.2.1.1 - Besturingssystemen

Van nature uit ondersteunt het *Mac* besturingssysteem *Flash* slechter dan *Windows*. Dit is duidelijk te merken als er gekeken wordt naar het CPU gebruik tussen beiden: In *Safari* op de *Mac* werd er ongeveer 37% CPU gebruikt, terwijl dezelfde browser in *Windows* maar 23% gebruikte. Dit zijn de resultaten met *Flash* v.10.0. Als er getest wordt met versie 10.1 van *Flash* wordt het verschil tussen de besturingssystemen pas echt duidelijk. In *Flash* v.10.1 is namelijk hardware acceleration toegevoegd: *"Hardware video decoding allows Flash Player to offload H.264 video decoding tasks from the CPU to deliver smooth, high quality video with minimal overhead, improving video playback performance, reducing system resource utilization, and extending battery life."*⁶

Deze functionaliteit was op het moment van het onderzoek alleen beschikbaar op *Windows* platformen (sinds augustus beschikbaar op nieuwe *Mac*'s). Het CPU gebruik tussen v.10.0 en v.10.1 in *Safari* daalde maar met 14% op de *Mac*. In *Windows* daalde het CPU gebruik met maar liefst 68%. Bij alle browsers die op *Windows* draaien daalt het CPU gebruik explosief (minimaal 35% in *IE*), terwijl het op *Mac*'s het verschil niet hoger komt dan 14%. Hieruit valt te concluderen dat *Flash* een stuk beter draait op *Windows*. Sinds v.10.1 van *Flash* is het verschil tussen beiden alleen maar groter geworden.



5.2.1.2 - Browsers

Ten tijde van de test was het <video> element niet in alle browsers ondersteund. Daarom zijn er alleen resultaten van *Chrome* te gebruiken, aangezien deze op zowel *Windows* en *Mac* werkte. Het CPU gebruik op de *Mac* was tijdens de *HTML5* test praktisch gelijk als in *Flash*: 50%. In *Windows* was er echter wel verschil: *HTML5* gebruikte 26% van het CPU, terwijl *Flash* op een gebruik van 20% (v.10.0) en 11% (v.10.1) uitkwam. Hieruit valt te concluderen dat *Flash* juist minder CPU gebruikt dan *HTML5*. Dit is precies het omgekeerde met wat men oorspronkelijk dacht. Echter, als er wordt gekeken naar *Safari* op de *Mac* wordt dit weer tegengesproken: daar gebruikte *HTML5* namelijk 12% tegenover 37% (v.10.0) en 32% (v.10.1) in *Flash*. Maar de reden hiervoor is dat *Safari* zijn eigen hardware acceleration toepast op video's met H.264 codec.

De overkoepelende conclusie is dat in februari van dit jaar *HTML5 Flash* nog niet kon evenaren qua CPU gebruik. Het zat in alle gevallen vast op hardware acceleration: *Windows* doet dit voor *Flash* en maakt een aanzienlijk verschil, terwijl *Mac's* dit niet hadden. *Safari* maakte toentertijd als enige browser gebruik van hardware acceleration op video's met H.264 codec. Hierdoor werd alleen *HTML5* sneller. Het was een strijd van ongelijke partijen vanaf het begin. Als deze test op dit moment opnieuw zou worden gedaan, zouden de resultaten een stuk definitiever zijn. Zowel *Windows* als *Mac's* hebben nu hardware acceleration voor *Flash* en ondersteunen alle browser tegenwoordig het <video> element.

5.2.2 - Frames per second

De meeste eindgebruikers zullen niet kijken naar hoeveel CPU er gebruikt wordt tijdens het afspelen van een video. Zij verwachten dat de video's, spellen en animaties soepel worden weergegeven. En dit heeft te maken met het aantal frames per second (fps). Hoe hoger het fps, hoe soepeler de applicatie wordt weergegeven. Hiervoor is er op de website *craftymind* een onderzoek gestart: drie applicaties, alle drie met een *Flash* en *HTML5* versie, waarvan de fps wordt berekend. De precieze datum van de resultaten is onbekend en daarom zal de auteur van dit onderzoek zijn eigen resultaten ook toevoegen.



5.2.2.1 - Vector Test ⁸

In deze test worden een aantal geanimeerde lijnen getekend (zie afbeelding 1), dat doet denken aan een seismograaf. Uit deze test blijkt dat op *Windows* computers het gemiddelde fps van de *HTML5* versie de helft minder is dan bij *Flash*. Er moet echter wel worden opgemerkt dat er veel verschil zit tussen de browsers: 15.73fps in *Firefox*, 6.41fps in *Chrome* en 24.77fps in *Opera*. Terwijl alle browsers in *Flash* rond de 29fps (met uitzondering van *Chrome*) zitten. Het testen op de *Mac* verliep niet zo soepel als op de *Windows*. Maar ook hier uit bleek dat *Flash* beter presteerde in deze test dan *HTML5*.

Browser	HTML5 fps	Flash fps
Opera 10.53	24.77	29.9
Chrome 4.1.249	6.41	26
Internet Explorer 8.0.6001	-	30.7
Safari 4.0.5	-	29.5
Firefox 3.6.3	15.73	29.65



5.2.2.1.1 - Eigen resultaten

Ook uit de eigen bevindingen blijkt dat *Flash* een hogere fps heeft dan *HTML5*. Ook al zit er een groot verschil tussen de waarden van boven- en onderstaande tabel, de conclusie die eruit gehaald kan worden blijft hetzelfde: *Flash* wordt een stuk soepeler weergegeven.

Browser	HTML5 fps	Flash fps
Opera 10.62	13.25	19.24
Chrome 6.0.472.63	2.25	12.12
Internet Explorer 8.0.6001	-	15.77
Safari 4.0.5 (531.22.7)	-	18.18
Firefox 3.6.10	8.86	13.34

5.2.2.2 - Bitmap Gaming test ⁹

In deze test worden monsters neergeschoten vanaf een vuurtoren met een laser (zie afbeelding 2). Wederom zijn de resultaten *Flash* gunstig: de fps van de *Flash* versie is twee keer zo hoog als de *HTML5* versie. Zo geeft *Opera* de *HTML5* versie het beste weer met 13,5fps terwijl de *Flash* versie 17 frames per seconde toont.

Browser	HTML5 fps	Flash fps
Opera 10.53	13.59	17.23
Chrome 4.1.249	10.1	15.98
Internet Explorer 8	-	17.34
Safari 4.0.5	-	17.29
Firefox 3.6.3	5.78	17.7

5.2.2.2.1 - Eigen resultaten

Uit eigen bevindingen blijkt dat het verschil tussen beide versies kleiner is geworden. Desalniettemin blijft *Flash* beter presteren dan de *HTML5* variant.

Browser	HTML5 fps	Flash fps
Opera 10.62	7.65	8.31
Chrome 6.0.472.63	6.06	8.28
Internet Explorer 8.0.6001	-	7.48
Safari 4.0.5 (531.22.7)	-	8
Firefox 3.6.10	3.23	8.21

5.2.2.3 - Text Column test ¹⁰

In deze test worden een aantal tekstvlakken geanimeerd, zodat ze groter en kleiner worden. De tekst past zich continue aan, aan de grootte van het tekstveld (afbeelding 3). De testresultaten zijn dit keer van de *Mac*, aangezien de resultaten op de *Windows* computer corrupt waren. Echter op beide besturingssystemen komt *HTML5* het beste uit de test.

Browser	HTML5 fps	Flash fps
Opera 10.10	22.72	15.22
Chrome 5.0.342	26.07	22.85
Safari 4.0.5	27.26	18.71
Firefox 3.6.3	23.61	18.71



5.2.2.3.1 - Eigen resultaten

In tegenstelling tot bovenstaande resultaten moet uit mijn eigen resultaten worden geconcludeerd dat *Flash* beter presteert. Behalve in *Firefox* waar de *HTML5* versie soepeler werd afgespeeld dan de *Flash* versie. Uit de bron en reacties op het artikel kunnen een aantal redenen hiervoor gevonden worden:

- Het aanpassen van een tekst aan de grootte van het veld waar het instaat (wordwrap), wordt standaard door browsers gedaan. In *Flash* gebeurt dit niet automatisch en moet de ontwikkelaar dit zelf aangeven.¹⁰
- De test is niet gemaakt met de laatste versie van een bepaald framework voor text layout in *Flash*.¹¹
- Hardware acceleration, dat in *Flash* 10.1 is geïntroduceerd, zou volgens één van de reacties de fps van 1.48 naar 24.62 hebben opgeschroefd.¹²

Browser	HTML5 fps	Flash fps
Opera 10.62	11.17	13.19
Chrome 6.0.472.63	11.49	11.7
Internet Explorer 8.0.6001	10.18	12.38
Safari 4.0.5 (531.22.7)	-	12.52
Firefox 3.6.10	13.39	12.47

Uit de eerste twee tests (zowel van de onderzoeker als mijzelf) blijkt duidelijk dat *Flash* qua prestaties beter is dan het <canvas> element. Het is interessant om te vermelden dat de eerste twee testen ook op een aantal mobiele platformen zijn uitgevoerd. Hieruit kwam dat de mobiele toestellen (*Nokia N900* en *HTC HD2*), die *Flash* ondersteunen dit soepeler weergaven dan de *HTML5* variant.¹³ De laatste test is echter tegenstrijdig aangezien de resultaten van de onderzoeker geen twijfel mogelijk laat, terwijl mijn resultaten een nipte overwinning van *Flash* aangeven. Samengevat kunnen we echter stellen dat *Flash* qua prestaties voorligt op *HTML5*: het loopt soepeler en het CPU gebruik ligt ook een stuk lager.

5.3 - Search Engine Optimization (Flash vs. Canvas)

In hoofdstuk 4.3 zijn de voordelen besproken van het <video> element voor Search Engine Optimization (SEO). Hieruit kon geconcludeerd worden dat *Flash* video's niet te lezen zijn door zoekmachines en dat *HTML5* video's op dezelfde manier als afbeeldingen verwerkt kunnen worden. Naast het weergeven van video's wordt *Flash* ook gebruikt om geanimeerde websites te maken. In deze websites wordt de inhoud tekstueel weergegeven en dat is waar zoekmachines mee kunnen werken. De vraag is echter in hoeverre zoekmachines tekst in *Flash* applicaties kunnen verwerken? En hoe zit het met tekst dat is weergegeven op een canvas?

Flash applicaties zijn voornamelijk grafisch (o.a. movieclips en sprites) en de elementen die hier verantwoordelijk voor zijn kunnen niet door zoekmachines worden doorlopen. Tekst elementen en links kunnen echter wel verwerkt worden, maar met één essentieel probleem: in vergelijking tot *HTML* documenten is tekst in *Flash* niet gestructureerd.¹⁴ Er zijn bijvoorbeeld geen methoden aanwezig om titels te onderscheiden van een paragraaf. Het tekstvlak in *Flash* kan een beetje vergeleken worden met het <div> element: er kan van alles instaan. Bij <div> elementen is het echter nog mogelijk om de relevantie te berekenen door naar de structuur van de gehele pagina te kijken (zie hoofdstuk 4.1). De zoekmachine zal in een *Flash* bestand bijvoorbeeld de copyright evenveel relevantie toekennen als de daadwerkelijk inhoud.

Als we kijken naar het <canvas> element en SEO moet er eigenlijk gekeken worden naar hoe zoekmachines om kunnen gaan met *JavaScript*. Het element is namelijk niet meer dan een container. De inhoud dat zal worden weergegeven wordt met *JavaScript* vervaardigd. In het geval van *JavaScript* zijn zoekmachines streng: "Google Site Search isn't able to index content contained in JavaScript. [...] JavaScript is meant for scripting only, if you add text in JavaScript thinking you would get top ranks in organic results you will not get that."¹⁵ Alle tekst op het canvas zal dus niet geïndexeerd worden. Bovendien zou bij canvas hetzelfde probleem optreden als bij *Flash*: de ongestructureerdheid van de pagina en de moeilijkheden om de relevantie te berekenen.



Als men toch zou willen dat de inhoud volledig geïndexeerd wordt, zou er gewerkt moeten worden met alternatieven. Voorbeelden:

- *HTML en Flash*: als het ware maakt de ontwikkelaar een tweetal weergaven aan: één *HTML* en één *Flash*. Standaard is de pagina *HTML* en kan daarom worden geïndexeerd door zoekmachines. Er kan bijvoorbeeld een sectie zijn opgenomen met daarin de tekstuele inhoud/uitleg van de *Flash* applicatie. Met hulp van *JavaScript* kan er gekeken worden of de gebruiker *Flash* kan afspelen. Zo ja, zal de sectie vervangen worden door het *Flash* element: ***“The content of that page can now be indexed because search engines can read the HTML-coded content, while visitors with JavaScript and Flash can view enhanced visual content.”***¹⁶
- *HTML en Canvas*: het `<canvas>` element is geen inhoudsloos element en kan daarom inhoud verwerken die tussen de tags is geplaatst. Tussen de tags zou de inhoud en/of uitleg worden geplaatst. De zoekmachines kunnen deze informatie wel indexeren. Het kan vergeleken worden met het alt attribuut van afbeeldingen.

5.4 - Conclusie

Met *HTML5* wilde de WHATWG onder andere ervoor zorgen dat gebruikers geen plug-ins, zoals *Flash Player*, van derden hoeven te downloaden om een webpagina in zijn volledigheid te kunnen zien. Deze techniek en de bijbehorende plug-in wordt vaak gebruikt om rijkelijk geanimeerde websites, geanimeerde onderdelen zoals banners, video en audio weer te geven. *HTML5* biedt hiervoor de `<canvas>`, `<video>` en `<audio>` elementen aan om deze functionaliteiten over te nemen, zonder dat de gebruiker een plug-in hoeft te downloaden.

Het is echter wel zo dat *Flash* al jarenlang in de wereld zit en in die tijd de werkwijze, prestaties en programma's geoptimaliseerd heeft. Terwijl *HTML5* figuurlijk gezien net om de hoek komt kijken. Dit is duidelijk te merken als er gekeken wordt naar de prestaties van beide. Zo is het CPU gebruik tijdens het afspelen van *Flash* video's lager dan wanneer dezelfde video wordt afgespeeld in de *HTML5* videospeler. Dit wordt nogmaals bevestigd met versie 10.1 van *Flash* waarin het CPU gebruik op *Windows* platformen gemiddeld met 50% daalde.

Als er gekeken zou worden naar het `<canvas>` element moet er tevens de conclusie getrokken worden dat *Flash* beter presteert. In dit geval werd er gekeken naar het frames per second (fps). Hoe hoger dit is, hoe soepeler de video/animatie zal worden weergegeven. En in bijna elke browser op zowel *Mac* als *Windows* computers waren de resultaten in het voordeel van *Flash*. Zelfs op mobiele toestellen, waarop *Flash* nog in zijn kinderschoenen loopt, geven de resultaten aan dat *Flash* beter presteert dan *HTML5*.

Op het gebied van Search Engine Optimization heeft *Flash* een kleine voorsprong: uit *Flash* applicaties kan tekstuele inhoud worden gefilterd, terwijl *JavaScript* (gebruikt voor canvas) helemaal niet wordt doorlopen. Van de inhoud uit *Flash* applicaties kan echter niet de relevantie worden berekend. Voor volledige indexatie van de inhoud (van zowel *Flash* als canvas) zal er gewerkt moeten worden met alternatieven:

- *Flash*: informatie in een *HTML* element dat vervangen wordt door de *Flash* applicatie.
- Canvas: informatie plaatsen tussen het `<canvas>` element. Gelijk aan het alt attribuut.

Op dit moment is *HTML5* nog niet de *Flash-Killer*, zoals men had verwacht. Het is waar dat het plug-ins onnodig maakt, maar qua prestaties loopt *HTML5* nog achter. Het is daarom de vraag of gebruikers de kleine ongemakken met plug-ins willen inruilen voor prestatieverlies of dat prestatie boven gemak verkozen wordt. Op dit ogenblik zie ik *Flash* nog niet vervangen worden door *HTML5*. Dit was echter te verwachten aangezien *Flash* al meer dan tien jaar in ontwikkeling. Als *HTML5* daadwerkelijk *Flash* overbodig wil maken, dan zal er de komende jaren nog drastisch aan gesleuteld moeten worden.



Bronnen

- ¹ : Thoughts on Flash | <http://www.apple.com/hotnews/thoughts-on-flash/> | | Steve Jobs | ??-04-2010 | 05-10-2010
- ² : HTML5 Overview | http://www.tutorialspoint.com/html5/html5_overview.htm | | | 05-10-2010
- ³ : Adobe Flash Player | <http://www.adobe.com/products/flashplayer/> | | Adobe | | 05-10-2010
- ⁴ : Adobe CTO Kevin Lynch Defends Flash, Warns HTML5 Will Throw The Web "Back To The Dark Ages Of Video" | <http://techcrunch.com/2010/02/02/adobe-cto-kevin-lynch-defends-flash/> | | Erick Schonfeld | 02-02-2010 | 05-10-2010
- ⁵ : Flash Player: CPU Hog or Hot Tamale? It Depends. | <http://www.streaminglearningcenter.com/articles/flash-player-cpu-hog-or-hot-tamale-it-depends-.html> | | Jan Ozer | 26-02-2010 | 05-10-2010
- ⁶ : Adobe Releases Flash 10.1 Beta with Hardware Acceleration | <http://www.macrumors.com/2010/04/29/adobe-releases-flash-10-1-beta-with-hardware-acceleration/> | | Arnold Kim | 29-05-2010 | 05-10-2010
- ⁷ : GUIMark 2: The rise of HTML5 | <http://www.craftymind.com/quimark2/> | | Sean Christmann | | 05-10-2010
- ⁸ : GUIMark 2: The rise of HTML5 | <http://www.craftymind.com/quimark2/> | Vector Charting Test | Sean Christmann | | 05-10-2010
- ⁹ : GUIMark 2: The rise of HTML5 | <http://www.craftymind.com/quimark2/> | Bitmap Gaming Test | Sean Christmann | | 05-10-2010
- ¹⁰ : GUIMark 2: The rise of HTML5 | <http://www.craftymind.com/quimark2/> | Text Column Test | Sean Christmann | | 05-10-2010
- ¹¹ : GUIMark 2: The rise of HTML5 | <http://www.craftymind.com/quimark2/#comment-22037> | Comment #8 | Om | 05-05-2010 | 15-10-2010
- ¹² : GUIMark 2: The rise of HTML5 | <http://www.craftymind.com/quimark2/#comment-22212> | Comment #73 | David Cabanlit | 11-06-2010 | 15-10-2010
- ¹³ : GUIMark 2: The rise of HTML5 | <http://www.craftymind.com/quimark2/> | GUIMark Mobile | Sean Christmann | | 05-10-2010
- ¹⁴ : Best uses of Flash | <http://googlewebmastercentral.blogspot.com/2007/07/best-uses-of-flash.html> | | Mark Berghausen (Search Quality Team, Google) | 05-07-2007 | 18-10-2010
- ¹⁵ : SEO Optimization and JavaScript | <http://www.webmarketingart.com/search-engine-optimization/JavaScript-and-seo/> | | Dilip Shaw | 17-05-2010 | 18-10-2010
- ¹⁶ : How to SEO Flash | <http://www.hochmanconsultants.com/articles/seo-friendly-flash.shtml> | Example: Making Flash Home Page Spiderable | Jonathan Hochman | | 18-10-2010



Hoofdstuk 6: Nu overstappen op HTML5 of toch nog even wachten?

In het vorige hoofdstuk hebben we kunnen lezen dat *Apple* onder andere op de *iPad* geen *Flash* ondersteuning aanbied. In plaats daarvan nemen andere technieken deze functionaliteiten over: waaronder *HTML5*. Dit is een gewaagde beslissing, aangezien *HTML5* nog steeds in ontwikkeling is. Het geeft echter wel aan dat grote bedrijven er vertrouwen in hebben dat de nieuwe *HTML* versie de webstandaard zal worden.

Maar, het is nog niet zo dat websites van grote bedrijven/organisaties al geheel in *HTML5* worden gemaakt. Zo heeft *Apple* een overzicht gemaakt van websites, die 'iPad ready'¹ zijn. Het valt op dat er voornamelijk gesproken wordt over het `<video>` en/of `<audio>` element. Deze trend is meer te zien: *TED* biedt hun 'TEDTalks' sinds 2010 aan in *HTML5*² en *YouTube* video's kunnen ook in *HTML5* worden weergegeven³. Er is alleen sprake van het gebruik van een enkele nieuwe functionaliteit: `<video>`.

Eén van de uitzonderingen op deze regel is *Google*; zij waren sinds 2007 aan het werk aan een open-source plug-in project genaamd *Google Gears*. Hierin zaten een aantal functionaliteiten, die nu standaard in *HTML5* zijn opgenomen; "[...] *that natively supports a Database API similar to the Gears database API, workers (both local and shared, equivalent to workers and cross-origin workers in Gears), and also new APIs like Local Storage and Web Sockets. Other facets of Gears, such as the LocalServer API and Geolocation, are also represented by similar APIs in new standards and will be included in Google Chrome shortly.*"⁴ Het gaat hier met name om functionaliteiten die oorspronkelijk deel uitmaakten van de *Web Apps 1.0* specificaties, namelijk web workers, web- en local storage en de geolocation API. Als gevolg hiervan zal *Google* niet langer meer verder ontwikkelen aan *Google Gears*. We kunnen hieruit ook ophalen in hoeverre *Google* betrokken is bij de ontwikkeling van *HTML5*: ze stoppen namelijk functionaliteiten van eigen producten in *HTML5*.

Het is haast onmogelijk dat *HTML5* niet de nieuwe webstandaard zal worden; het zou zonde zijn om geen gebruik te maken van de nieuwe functionaliteiten en het gemak dat kan opleveren. Als zelfs toonaangevende bedrijven uit de webwereld gebruik maken/voorkeur geven aan *HTML5*, kan de beweging niet meer stopgezet worden. Maar zoals zojuist is aangegeven gebruiken de meeste diensten maar enkele van de nieuwe functionaliteiten (voornamelijk video). In hoeverre is het daarom verstandig om volledig over te stappen op *HTML5*?

6.1 - Backwards compatibility

Oftewel achterwaartse compatibiliteit is één van de hoofddoelen van de *Web Hypertext Application Technology Working Group (WHATWG)* (zie hoofdstuk 1.2.1). Wat dit inhoudt komt op het volgende neer: er wordt een product ontwikkeld dat een bestaand product zal gaan vervangen. De software voor het nieuwe product is nog niet beschikbaar en daarom zal de oude software in eerste instantie gebruikt worden. Als deze oude software goed functioneert op het nieuwe product (dat niet met/voor deze software is ontwikkeld) spreken we van achterwaartse compatibiliteit. Om het iets meer binnen het onderwerp te plaatsen: *HTML5* zal de rol van *XHTML* en *HTML4* over nemen, maar moet daarvoor wel deze twee versies volledig ondersteunen. Het mag niet zo zijn dat functionaliteiten uit *XHTML* en/of *HTML4*, die niet in de *HTML5* zijn opgenomen, niet meer werken. Aan de hand van een voorbeeld zullen we zien dat dit inderdaad zo is:

```
#1      <!DOCTYPE html>
        <html lang="en">
          <head>
            <meta charset="UTF-8" />
            <style>
              header, footer, aside, nav, article, section { display: block; }
              ...
            </style>
          </head>
<!-- Zie volgende bladzijde voor vervolg codevoorbeeld -->
```



#18

```

<body>
  <section>
    <header>
      <hgroup>
        <h1>HTML4 in HTML5</h1>
        <h2>Elementen die niet zijn opgenomen in HTML5 specificaties</h2>
      </hgroup>
      <br><br>
      <p>Dit is een normaal stuk tekst om mee te vergelijken</p>
    </header>
    ...
    <article>
      <center>HOPPINGER - Center Element</center>
      <br><br>
      <font face="Arial" size="3">Hoppinger - Font Element</font> [face & size]
      <br><br>
      <u>Hoppinger - Underline Element</u>
      <br><br>
      <big>Hoppinger - Big Element</big>
    </article>
  </section>
</body>
</html>

```

Het resultaat van dit stuk code is hiernaast te zien: alle elementen die zijn gebruikt (<center>, , <u> en <big>) worden juist weergegeven. Terwijl deze niet zijn opgenomen in de *HTML5* specificaties en het document in *HTML5* is gemaakt (zie doctype).

- Regel #1 van het document is al een duidelijke indicatie dat het in *HTML5* goed geregeld is met de achterwaartse compatibiliteit. Er wordt immers niet gerefereerd naar een bepaalde versie, maar simpelweg naar *HTML*. Hieruit kan opgehaald worden dat alle functionaliteiten van (X)HTML gebruikt kan worden en dat deze juist worden weergegeven.
- Elementen zonder inhoud (zoals en
 [zie regel #18]) moesten in *XHTML* met een '/' eindigen; en
. In *HTML5* is dit niet langer verplicht en de ontwikkelaar mag zelf kiezen hoe hij/zij het neerzet.



In het geval van *HTML5* kunnen nieuwe functionaliteiten ook werken in de voorgangers. Zo kan bijvoorbeeld het video element gebruikt worden, terwijl de rest van de pagina in *XHTML* is geschreven en de doctype ook aangeeft dat het *XHTML* is. Het voordeel hiervan is dat bestaande documenten niet geheel omgezet hoeven te worden naar *HTML5* voordat een bepaald element (zoals video) kan werken. Tevens kan de pagina als *XHTML* gevalideerd blijven worden. Validators voor *HTML5* zijn schaars en zoals ze zelf aangeven 'experimental'.^{5/6} Voorbeeld:

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="nl" lang="nl">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <script>
      window.onload = function()
      {
        canvas = document.getElementById('canvas');
        context = canvas.getContext('2d');

        logo = new Image();
        logo.src = 'hoppinger.png';

        context.drawImage(logo, 0, 0);
      }
    </script>
  </head>
<!-- Zie volgende bladzijde voor vervolg codevoorbeeld -->

```



```

<body>
    <div id="section">
        <div id="header">
            <h1>HTML5 in XHTML</h1>
            <h2>Elementen die niet zijn opgenomen in XHTML</h2>
        </div>
        ...
        <div id="article">
            <canvas id='canvas' width='375' height='175'></canvas>
        </div>
    </div>
</body>
</html>

```



Er is duidelijk aangegeven dat het hier om een *XHTML strict* document gaat. Desalniettemin wordt er gebruik gemaakt van het `<canvas>` element, dat in de tijd van *XHTML* nog niet bestond. Het resultaat is hetzelfde als wanneer het in een *HTML5* document zou staan. Deze ondersteuning heeft als resultaat dat een aantal bedrijven/organisaties hun video's ook in *HTML5* zijn gaan aanbieden: de website hoeft immers niet compleet herbouwd te worden, terwijl de functionaliteit wel werkt.

6.2 - Browser compatibility

Een andere reden voor het gebruik van het `<video>` element door *YouTube* en *TED* is dat alle browsers de functionaliteit ondersteunen. *Internet Explorer* pas vanaf versie 9 en deze is op dit moment in de beta fase. In het geval van *YouTube* en *Internet Explorer* 8 gebruikers zal er teruggevallen worden op de *Flash* versie van de video. Functionaliteiten die nog weinig ondersteuning genieten, zullen dan ook weinig op websites gebruikt worden. Ze zullen hoogstens op experimentele websites terug te vinden zijn. De voorkeur ligt nu nog bij bijvoorbeeld de *JavaScript* alternatieven, die wel in alle browsers werken.

In bijlage één is een overzicht te vinden met de mate van ondersteuning per browser voor de functionaliteiten besproken in hoofdstuk twee. Als alle browsers met elkaar vergeleken worden valt er te concluderen dat er maar een handjevol functionaliteiten zijn die in alle browsers werken. Hoe kan een ontwikkelaar hier het beste mee omgaan? Eén advies zou zijn om te kijken naar de browser met de minste ondersteuning voor *HTML5*. Op dit ogenblik is dat *Internet Explorer* 8, maar in IE9 zal de ondersteuning een stuk beter zijn. Het is al mogelijk om met deze versie te werken.

In de nieuwe versie van IE zullen een vijftal functionaliteiten worden ondersteund: web storage, geolocation, canvas(text), video en audio. Als we dit vergelijken met de overige browsers zullen we zien dat deze functionaliteiten ook daarin worden ondersteund (Geolocation zal pas in versie vijf van *Safari* worden ondersteund, maar technisch gezien is Geolocation geen *HTML5*). Hieruit kunnen we opmaken dat deze gebruikt kunnen worden, zonder dat er een alternatief voor niet ondersteunende browsers hoeft worden aangeboden. Dit is waarschijnlijk de veiligste manier om te beslissen welke mogelijkheden een webontwikkelaar kan/wil gebruiken. Andere manieren zijn: browser met de beste ondersteuning (*Opera*) of de meest gebruikte browser (*Firefox*).

6.3 - Conclusie

Een aantal grote bedrijven/organisaties implementeren steeds meer *HTML5* in hun producten. Zo zijn *YouTube* video's niet alleen meer in *Flash* te aanschouwen en biedt *TED* hun 'TEDtalks' aan in een *HTML5* format. Op grote schaal blijft het vooralsnog gelimiteerd aan het `<video>` element. Het zijn vooral experimentele *HTML5* georiënteerde websites waar een breder scala aan functionaliteiten worden gebruikt.



Deze trend is te onderbouwen met het feit dat alle browsers het <video> element ondersteunen. Bovendien is het niet nodig om een gehele website om te bouwen naar *HTML5* om het te laten werken. In het simpelste geval zal het blijven bij het toevoegen van één enkele regel code (zie hoofdstuk 2.4.1). Met een aantal eenvoudige veranderingen kan de website beter functioneren op mobiele platformen en kunnen de video's gemakkelijker worden geïndexeerd door zoekmachines (zie hoofdstuk 4.3).

Nu overstappen op *HTML5* of toch nog even wachten? Op het moment van schrijven geen van beide; een tussenweg is in dit geval het beste. Dit houdt in dat *HTML5* functionaliteiten geleidelijk worden geïmplementeerd. De achterwaartse compatibiliteit is goed gedocumenteerd waardoor nieuwe mogelijkheden zonder probleem in oude code gebruikt kunnen worden. De beslissing om te kiezen wanneer men een bepaalde functionaliteit implementeert ligt aan de mate van ondersteuning. Het advies luidt; implementeer alleen functionaliteiten die in elke browser werken of die voor bepaalde browsers met een eenvoudig alternatief kunnen worden vervangen

Echter, het advies zal hoogstwaarschijnlijk veranderen naargelang we dichterbij het jaar 2012 komen. In 2012 is namelijk de afgeronde versie van de *HTML5* specificaties ingepland. De specificaties zijn zichtbaar voor iedereen en het is onwaarschijnlijk dat er op dit ogenblik nog nieuwe specificaties bij komen. Daarom hoeven browsers niet te wachten tot 2012 om de functionaliteiten in te bouwen. Mijn verwachting is dat rond 2012 minimaal drie browsers meer dan 90% van de functionaliteiten heeft geïmplementeerd. Uiteraard zullen gebruikers dan wel deze browsers moeten gaan gebruiken. Dit kan een wellicht een probleem zijn, aangezien nog steeds veel mensen met *Internet Explorer 6* werken. En dit terwijl *IE8* al sinds maart 2009 beschikbaar is en versie 9 in de betafase zit. Desalniettemin lijkt het erop dat men steeds meer geneigd is om direct over de stappen naar een nieuwe versie.^{8&9} Een goed voorbeeld hiervan, dat te zien is in de grafiek van bron negen, is Chrome: binnen een maand is versie 3.0 compleet verdwenen en is versie 4.0 ervoor in de plaats gekomen.

Bronnen

¹ : iPad ready | <http://www.apple.com/ipad/ready-for-ipad/> | | | 07-10-2010

² : TED.com now available in HTML5, serving many mobile platforms, including iPhone, iPad | http://blog.ted.com/2010/03/31/tedcom_now_avai/ | | Emily McManus | 31-03-2010 | 07-10-2010

³ : YouTube Launches New HTML5 Mobile Site | <http://www.wired.com/epicenter/2010/07/youtube-launches-new-html5-mobile-site/> | | Michael Calore | 08-07-2010 | 07-10-2010

⁴ : Hello HTML5 | http://gearsblog.blogspot.com/2010/02/hello-html5.html?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed%3A+GearsApiBlog+%28Gears+API+Blog%29 | | Ian Fette, Gears Team | 19-02-2009 | 07-10-2010

⁵ : W3C Markup Validation Service | http://validator.w3.org/#validate_by_uri+with_options | Document Type | | | 07-10-2010

⁶ : Validator.nu (X)HTML5 Validator (Highly Experimental) | <http://html5.validator.nu/> | | | 07-10-2010

⁷ : Browser Statistics | http://www.w3schools.com/browsers/browsers_stats.asp | Browser Statistics Month by Month – September 2010 | | ??-10-2010 | 07-20-2010

⁸ : 2010 Browser Stats and Trends | <http://nicholasbailey.blogspot.com/2010/06/2010-browser-stats-and-trends.html> | IE users are upgrading | | 01-06-2010 | 03-11-2010

⁹ : 2010 Browser Stats and Trends | <http://nicholasbailey.blogspot.com/2010/06/2010-browser-stats-and-trends.html> | The hidden chaos: StatCounter Global Stats | | 01-06-2010 | 03-11-2010



Hoofdstuk 7: Wat heeft HTML5 te bieden?

In dit onderzoek zijn een zestal onderwerpen met betrekking tot de techniek *HTML5* besproken: achtergronden, vernieuwingen, compatibiliteit, semantiek en SEO, *Flash* vs. *HTML5* en toekomstbeeld. In dit hoofdstuk zal er gekeken worden naar hoe *HTML5* een uitkomst/voordeel kan bieden voor webbedrijven. Dit wordt gebaseerd op de informatie dat vergaard is in de loop van dit onderzoek. Elk hoofdstuk wordt voorzien van een samenvatting en een conclusie. Uitzondering hierop is hoofdstuk één waar een conclusie niet van toepassing is.

7.1 - Wat is de essentie van HTML5 en wat wil men ermee bereiken?

7.1.1 - Samenvatting

De *Web Hypertext Application Technology Working Group (WHATWG)* is opgericht door een groep browserontwikkelaars en geïnteresseerden in web technologie, in 2004. De algemene mening luidde als volgt: de huidige generatie markup languages is niet gebouwd voor de manier waarop het web hedendaags wordt gebruikt. Er vindt tegenwoordig een stuk meer interactiviteit plaats dan in de beginjaren van het internet. Op dit moment wordt dit door andere technieken afgehandeld. Een voorbeeld hiervan is *Flash*, dat veel wordt gebruikt voor het maken en/of weergeven van video's en animaties.

Het *WHATWG* wil daarom *HTML* uitbreiden met functionaliteiten, die het gebruik van deze technieken en de plugins die hier vaak voor nodig zijn, onnodig maakt. Het motto hierbij is dat men bekende technieken (zoals *HTML*, *CSS*, *DOM* en *JavaScript*) kan gebruiken om web applicaties te maken. Naast de specificaties voor web applicaties zijn er nog een aantal speerpunten in de ontwikkeling van *HTML5*: error afhandeling moet precies gedocumenteerd worden, zodat browsers niet zelf oplossingen hoeven te bedenken en dat gebruikers uiteindelijk niks zullen merken van fouten in de achterliggende code. Dit is wederom een stap voorwaarts in de queeste om de uniformiteit op het web te vergroten. Daarnaast is het bedoeling dat *HTML5* producten in elke browser en op elk platform dezelfde functionaliteiten aanbieden. Als laatste wil men ervoor zorgen dat *HTML5* achterwaarts compatibel is met zijn voorgangers. Het gebruik van *HTML5* en *XHTML* en vice versa zou geen probleem moeten opleveren.

7.2 - Wat zijn de nieuwe mogelijkheden van HTML5 in vergelijking met XHTML?

7.2.1 - Samenvatting

De nieuwe mogelijkheden van *HTML5* kunnen worden onderverdeeld in een drietal categorieën: elementen, formulieren en web applicaties. De element categorie kan vervolgens worden opgedeeld in twee categorieën: veranderd en toegevoegd. De veranderingen hebben voornamelijk plaats gevonden op de elementen, die worden gedefinieerd boven de daadwerkelijk inhoud (het `<body>` element). De algemene constatering is dat de elementen zijn ingekort, door het verdwijnen van *XHTML* specifieke attributen en/of onnodige attributen. Daarnaast zijn ruim een tiental nieuwe elementen toegevoegd waarmee specifieke rollen binnen een pagina kunnen worden aangegeven. Hierdoor zal het meningsloze `<div>` element in een groot aantal situaties vervangen worden en wordt de algehele semantiek van de pagina verbeterd.



De vernieuwingen die zijn doorgevoerd aan web formulieren hebben een drietal hoofddoelen voor ogen: het formulier uitbreiden met inhoudspecifieke invoervelden, de kracht van het mobiele platform benutten en het makkelijker maken om een uitgebreid formulier te creëren.

- Formulier uitbreiden met inhoudspecifieke invoervelden: er zijn nieuwe invoervelden toegevoegd, die specifiek bedoeld zijn voor email- en web adressen, tijden/datums en nummers. Dit maakt het ook een stuk makkelijker om de inhoud te valideren aangezien de ontwikkelaar/browser weet wat voor waarden er verwacht wordt.
- De kracht van het mobiele platform benutten: populaire mobiele platformen (smartphones en tablets) zijn tegenwoordig vaak uitgerust met een touchscreen. Dit maakt het mogelijk om de inhoud van het toetsenbord aan te passen aan de verwachte waarde. Een `<input type="number">` zou bijvoorbeeld een toetsenbord openen met alleen getallen.
- Het makkelijker maken om een uitgebreid formulier te creëren: nieuwe attributen, zoals autofocus, placeholder, pattern en required bieden met één á twee woorden dezelfde functionaliteiten als een stuk *JavaScript*. Voor autofocus bijvoorbeeld wordt nu een stuk *JavaScript* van minimaal vier regels gebruikt.

Als laatste zijn er dan nog de specificaties met betrekking tot web applicaties. Een web applicatie kan het beste gedefinieerd worden als een stuk software op het internet. Binnen de term web applicaties vallen de volgende specificaties:

- Video, audio en canvas voor het weergeven van verschillende mediavormen. Canvas kan onder andere gebruikt worden om animaties mee te maken.
- Microdata kan gebruikt worden om data leesbaar te maken voor machines, zodat deze bijvoorbeeld gebruikt kunnen worden in samenhang met andere applicaties zoals agenda's (*Google agenda*) en/of verlanglijsten bij een webwinkel.
- Web- en offline storage om data op te slaan en om de applicatie/website identiek te laten werken/weergeven zelfs als er geen internetverbinding is.
- Drag and drop om het selectieproces een stuk interactiever te maken. De gebruiker is niet langer gebonden aan klikken, maar kan ook slepen. Deze specificatie is haast identiek aan eenzelfde functionaliteit, die in *IE5* werd geïntroduceerd.
- Geolocation kan gebruikt worden om de locatie van de gebruiker op te vragen. Dit maakt eigenlijk geen deel uit van *HTML5*, maar wordt door velen wel als zodanig benoemd.

7.2.2 - Conclusie

7.2.2.1 - Semantische elementen

Naast voordelen voor machines (betere semantiek en bijkomende SEO voordelen) hebben de nieuwe semantische elementen ook voor ontwikkelaars voordelen: zo wordt de code een stuk kleiner, leesbaarder en ontstaat er meer uniformiteit.

- Kleiner: vooral elementen, die boven het `<body>` element worden aangegeven, zijn gekortwiekt om onnodige informatie weg te laten. Hierdoor hoeft de ontwikkelaar minder te onthouden, te kopiëren en plakken, code te tikken en zal de foutmarge kleiner worden.
- Leesbaarder: op dit ogenblik worden onderdelen vaak aangegeven met een `<div>`. Ontwikkelaars moeten via de id en/of class attributen achter de context komen. Het ligt uiteraard aan de ontwikkelaar zelf in hoeverre deze attributen representatief benoemd zijn met betrekking tot de inhoud/functie van het element. De nieuwe semantische elementen geven de context aan en het zal daarom voor ontwikkelaars gemakkelijker worden om snel het juiste element te vinden. Voorbeeld: men zal sneller `<footer id="articleInfo">` (tussen andere nieuwe semantische elementen) vinden, dan bijvoorbeeld `<div id="articleInfoFooter">` (tussen andere `<div>`'s).
- Uniformiteit: Elke ontwikkelaar benoemt zijn elementen anders (bv. naar de inhoud `class="newsarticle"` of hiërarchie `class="maincontent-text"`) en het kan daardoor lastig zijn om snel de functie van het element te achterhalen. De nieuwe semantische elementen tonen de ontwikkelaar in één opslag wat voor soort informatie het bevat. Bovenstaand voorbeeld kan bijvoorbeeld worden opgelost met `<article class="news">`.



De laatste twee voordelen maakt het ook makkelijker om een document over te dragen aan een andere ontwikkelaar. Deze zal een document krijgen waarin elk onderdeel is afgebakend en is benoemd naar de inhoud. Als er iets in de header van de webpagina veranderd moet worden, hoeft er alleen maar op '<header>' of 'header' gezocht te worden. De ontwikkelaar hoeft geen weet te hebben van de id of class naam, die er door de vorige ontwikkelaar aan is gegeven (bijvoorbeeld id="paginaHoofd").

7.2.2.2 - Formulieren

Het zal een stuk makkelijker worden om een uitgebreid formulier te maken, zonder dat hier lastige *JavaScript* codes aan te pas hoeven te komen. Bijvoorbeeld het valideren van ingevulde waarden; nu kan er met *PHP* en/of *JavaScript* gekeken worden of hetgeen er ingevuld is ook daadwerkelijk overeenkomt met wat er verwacht wordt. In HTML5 zal dit automatisch gebeuren, zoals op dit ogenblik al gebeurd op email, url en number input types in Opera. Dit neemt een hoop werk uit handen van ontwikkelaars en de algemene kwaliteit van formulieren zal stijgen. De algemene tendens van de formulier specificaties is het standaardiseren van functionaliteiten, die tot nu toe met *JavaScript* werden opgelost. Voorbeelden hiervan zijn; de autofocus en placeholder attributen en het output element.

Daarnaast zijn er een aantal input typen en elementen, die de visuele kant van de webpagina uitbreiden. Zo zijn er de progress en meter element, die harde waarden omzetten naar een grafische weergave (bv. een geanimeerde laadbalk). Ook de date input types doen iets soortgelijks; deze zullen de gebruiker een kalender aanbieden om een datum of tijd te kiezen. Ook deze input types werden tot dit moment met *JavaScript* aangeboden. Wederom komt het erop neer dat het makkelijker wordt om met enkele simpele stappen een webpagina rijker te maken qua functionaliteiten.

7.2.2.3 - Web applicaties

Vooral het <video> (en <audio>) element zal aanzienlijke voordelen opleveren; het is niet meer nodig om de video ergens anders te hosten, zoals *YouTube* of *Vimeo*. Video's die embed worden zullen daardoor niet meer voorzien worden van watermerken en/of advertenties. De professionaliteit van de website zal hierdoor stijgen. Daarnaast zal het niet nodig zijn om een plugin te installeren om de inhoud te zien. Dit is ook het geval met <canvas> waarmee animaties gemaakt kunnen worden.

Een andere functionaliteit dat de web ervaring positief kan uitbreiden is de mogelijkheid om een website ook offline aan te bieden. Hierdoor kan een gebruiker de website altijd zien, ook al is zijn/haar internetverbinding weggefallen. Dit wordt gerealiseerd door de gebruikte bestanden lokaal op te slaan en te gebruiken zodra er geen internetverbinding is. Een ander opslagmedium is web storage waarmee bijvoorbeeld gebruikersgegevens kan worden opgeslagen (identiek aan een cookie alleen uitgebreider). Het gedrag van een bezoeker kan bijvoorbeeld hiermee worden opgeslagen om later gebruikt te worden door de website om een betere dienst te verlenen. Bijvoorbeeld door artikelen aan te bieden die passen bij de interesses van de gebruiker. Een andere specificatie die hierbij kan helpen is de Geolocation API waarmee de locatie van de gebruiker kan worden opgevraagd. Hierdoor kunnen er bijvoorbeeld producten en diensten worden aangeboden, die relevant van zijn voor de locatie van de gebruiker.

7.3 - Ligt het succes van HTML5 bij het aanpassingsvermogen van de grote browsers?

7.3.1 - Samenvatting

Eén van de grote verschillen in de manier van ontwikkelen tussen de *W3C* en *WHATWG* is dat laatstgenoemde het belangrijk vindt dat er volledige openheid is. Iedereen moet toegang kunnen hebben tot de specificaties. Dit heeft ervoor gezorgd dat ontwikkelaars met de techniek aan de slag konden gaan terwijl het nog in ontwikkeling is: in 2012 wordt pas de voltooide specificaties verwacht. Aan deze aanpak kleef één groot voordeel (op lang termijn) en één groot nadeel (op kort termijn).



Het nadeel is dat alle functionaliteiten gebruikt kunnen worden, maar nog niet alle functionaliteiten ondersteund worden door de browsers. Figuurlijk gezien wordt men blij gemaakt met een dode mus. Letterlijk gezien krijgt men functionaliteit voorgeschoteld die veel voordeel op kan leveren, maar die echter nog niet gebruikt kan worden. Dit is een groot nadeel op korte termijn en wordt gecreëerd doordat men kan ontwikkelen met een techniek die nog in ontwikkeling is. Maar de openheid die de WHATWG verkiest kan er wel voor zorgen dat de browsers volledige/zeer grote mate van ondersteuning bieden zodra de specificaties voltooid zijn. Dit is het grote voordeel op langer termijn, aangezien browsers voorheen pas kon beginnen met implementeren zodra de voltooid specificaties werden vrijgegeven.

7.3.2 - Conclusie

Een obstakel in het werken met *HTML5*, op dit moment, ligt in de mate van ondersteuning door browsers. Zoals in bijlage één te zien is verschilt dit nogal. Zo is Opera de trendsetter met haast 75% ondersteuning en *Internet Explorer 8* de hekkensluiter met minder dan 10%. Er zijn maar een aantal functionaliteiten, die door alle moderne browsers worden ondersteund. Oudere browsers zoals *Internet Explorer 6* en *7* bieden weinig/geen ondersteuning. Er zijn echter wel scripts/libraries geschreven om *HTML5* functionaliteiten te laten werken in oudere browsers: zoals *ExplorerCanvas* om de `<canvas>` functionaliteit te laten werken in *Internet Explorer 8* en lager. Maar niet alle functionaliteiten zijn voorzien van dit soort oplossingen.

7.4 - Zal *HTML5* voordelen opleveren op het gebied van *SEO* in vergelijking met *XHTML*?

7.4.1 - Samenvatting

In dit hoofdstuk zijn een drietal onderwerpen besproken; page segmentation, het beïnvloeden van de browser en *HTML5* video.

- Page segmentation: De nieuwe semantische elementen kunnen gebruikt worden om het zoekmachines gemakkelijker te maken om de context en relevantie van een webpagina te indexeren. In tegenstelling tot *HTML5* moeten er voor het segmenteren (het uit elkaar halen) van pagina's lastige algoritmen gebruikt worden om context en relevantie te berekenen. In *HTML5* zal de ontwikkelaar zelf de context aangeven met de nieuwe elementen. Zo zal de header van de webpagina of een artikel in het `<header>` element staan en de hoofdinhoud in een `<article>` element. Zoekmachines zullen begrijpen dat het `<article>` element het meest relevant is en gebruiken dit in hun zoekresultaat.
- Het beïnvloeden van de browser: met behulp van het `<time>` element kan de ontwikkelaar aangeven wanneer iets is gepubliceerd of wanneer er iets plaatsvindt (bv. een evenement). Hierdoor geeft hij/zij tevens de relevantie aan van de webpagina of artikel. Hoe nieuwer het immers is, hoe groter de kans dat het relevant is voor de gebruiker. Daarnaast is er nog microdata waarmee ontwikkelaars belangrijke informatie opslaan in een virtueel 'dossier'. Deze informatie is leesbaar voor machines en kan bijvoorbeeld door zoekmachines gebruikt worden om de zoekresultaten specifiekere mee te maken. De ontwikkelaar geeft kortom een hint aan de zoekmachine welke informatie belangrijk is.
- *HTML5* video: als er gezocht wordt naar afbeeldingen in zoekmachines zullen de resultaten overal vandaan gehaald worden (zelfs van de website van ome Joop en zijn tandenstoker verzameling). Als er gezocht zou worden naar video's zou het aanbod gelimiteerd blijven tot websites, die gespecialiseerd zijn in het aanbieden van dit medium (bv. *YouTube* en *Dailymotion*). Dit komt doordat tot nu toe video wordt weergegeven met *Flash*, dat een onzichtbare muur rond het bronbestand plaatst. Het is voor zoekmachines daardoor moeilijk tot onmogelijk om de video te kunnen indexeren. *HTML5* video wordt daarentegen op dezelfde manier aangegeven als een afbeelding (bronbestand via het `src` attribuut) en kan daardoor wel geïndexeerd worden.



7.4.2 - Conclusie

Een aantal van de functionaliteiten in *HTML5* leveren voordelen, op het gebied van Search Engine Optimization (SEO) op. Het grootste voordeel zal behaald kunnen worden via de semantische elementen. In de wereld van zoekmachines draait het namelijk om de content en de nieuwe elementen zorgen dat dit beter te bereiken is voor machines. Doordat het voor zoekmachines makkelijker zal zijn om de relevantie, context en inhoud te verkrijgen van een webpagina zal deze beter geïndexeerd worden. Resultaat is dat de website of webpagina hoger in de zoekresultaten komt te staan en daardoor sneller bezocht zal worden. Hier zal de klant blij mee zijn en de ontwikkelaars zullen hier ook voldoening uit krijgen. Een website waar weken werk in zit, maar die geen bezoekers krijgt resulteert in het tegenovergestelde.

Om semantische elementen in alle browsers (zowel oud als nieuw) te kunnen gebruiken zijn er twee kleine 'hacks' nodig. Eén daarvan is het importeren van een stuk *JavaScript* code, waarmee Internet Explorer de elementen zal herkennen. De tweede 'hack' is in CSS en wordt gebruikt om van alle elementen een block element te maken. Hierdoor hebben ze dezelfde opmaak als een standaard `<div>` en kan er CSS op uitgevoerd worden. Het kost kortom weinig moeite om de semantische elementen in elke browser te laten werken en er de figuurlijke vruchten van te plukken. In de conclusie van hoofdstuk 7.6 zal ik hier verder op in gaan.

7.5 - Is HTML5 een goed alternatief om Flash mee te vervangen?

7.5.1 - Samenvatting

Steve Jobs, directeur bij *Apple*, uitte zijn ongenoegen over Flash in een brief gericht naar *Adobe* in april van 2010. In één van de laatste zinnen gaf hij aan dat ze beter meer tijd in *HTML5* konden steken in plaats van in het achterhaalde *Flash*. Het algemene beeld dat er werd gecreëerd was er één van *HTML5* als 'flash-killer'. De *WHATWG* heeft zijn hoofddoel om plugins onnodig te maken behaald, aangezien video, audio, games en animaties niet langer meer met *Flash* (of *Silverlight*) gemaakt en afgespeeld hoeven te worden. Uit meerdere onderzoeken en benchmark tests is echter wel gebleken dat *HTML5* qua prestaties nog niet op het zelfde niveau staat als *Flash*.

Voor het afspelen van een *YouTube* video (die in zowel *HTML5* als *Flash* worden aangeboden) eiste *HTML5* gemiddeld meer CPU. Ook bij tests naar het FPS van een drietal applicaties kwam Flash als beste naar voren. Flash kwam bovendien ook als beste voor de dag als het gaat om SEO, in vergelijking tot Canvas: Javascript (dat gebruikt wordt voor Canvas) wordt namelijk niet doorlopen, terwijl crawlers uit Flash nog links en tekst kunnen halen. De relevantie en context hiervan kan echter niet worden bepaald.

7.5.2 - Conclusie

Met *HTML5* zouden onder andere de volgende werkzaamheden van Flash overgenomen kunnen worden; video, audio, rijkelijke geanimeerde en interactieve websites en games. De eerste twee functionaliteiten kunnen inderdaad deze rol al overnemen zoals onder andere te zien is met de *HTML5* videospeler op *YouTube*. Een voordeel is dat de `<video>` en `<audio>` elementen in haast iedere browser ondersteund worden. Met minimaal één regel code staat er al een volledig werkende videospeler op de website.

Naast video en audio zou *HTML5* ook gebruikt kunnen worden voor rijkelijke geanimeerde en interactieve websites en games Met `<canvas>` kunnen de eerder genoemde werkzaamheden vervaardigd worden in een al bekende techniek (*JavaScript*). Echter uit eigen experimenten kan ik concluderen dat beide technieken niet dezelfde functionaliteiten bieden. Zo is een simpel klik event op een afbeelding in `<canvas>` niet mogelijk. Met Flash is op dit ogenblik in de ontwikkeling van *HTML5* meer te bereiken en tevens een stuk makkelijker (bv. een tijdlijn animatie).



7.6 - Nu overstappen op HTML5 of toch nog even wachten?

7.6.1 - Samenvatting

Doordat *HTML5* achterwaarts compatibel is met zijn voorgangers, zou het geen problemen moeten opleveren om *HTML5* in *XHTML* te implementeren of vice versa. Daardoor is het <video> element zo immens populair en wordt het al zo veel gebruikt. Het komt er op neer dat men gebruik kan maken van de voordelen zonder dat de hele website omgezet hoeft te worden naar *HTML5*. Dit is overigens voor alle nieuwe functionaliteiten het geval. Uiteraard moet er wel gekeken worden of de functionaliteit voldoende of volledig ondersteund wordt.

7.6.2 - Conclusie

Niet overstappen en ook niet wachten; een tussenweg is in dit geval het beste. Dit houdt in dat *HTML5* functionaliteiten geleidelijk worden geïmplementeerd. De beslissing om te kiezen wanneer men een bepaalde functionaliteit implementeert ligt aan de mate van ondersteuning. Het advies luidt; implementeer alleen functionaliteiten die in elke browser werken of die voor bepaalde browsers met een eenvoudig alternatief vervangen kunnen worden. Het moet in ieder geval niet zo zijn dat bepaalde functionaliteiten van een website in bepaalde browsers en/of platformen niet werken.

In 2012 verwacht men dat de voltooide versie van *HTML5* zal worden opgeleverd. Mijn verwachting is dat een tweetal browsers rond dit tijdstip haast volledige ondersteuning bieden. Dit betekent echter niet dat vanaf dat moment alle functionaliteiten gebruikt kunnen gaan worden. Het aantal mogelijke functionaliteiten verschilt met welke browsers men de website compatible wilt hebben. Zo zullen oudere browserversies (bv. IE6) niet meer geüpdate worden en ondersteuning voor deze versies vanuit de ontwikkelaar zal het aantal mogelijke *HTML5* functionaliteiten limiteren. In de komende twee jaar zal echter het aantal gebruikers per browser veranderen (meer *Chrome* gebruikers en minder *Internet Explorer*), zoals nu al gaande is. Hier kunnen ontwikkelaars zich aan aanpassen en dit zal zorgen voor een bredere set aan functionaliteiten die gebruikt kunnen worden.

Kort samengevat; De enige functionaliteit, die door alle browsers ondersteund worden, waarvan ik adviseer om ze te gaan gebruiken zijn de semantische elementen. Deze hebben immers geen eigen functionaliteit (bv. een video toevoegen of data opslaan) en worden door browsers gezien als onbekende elementen. Er zijn enkel twee kleine 'hacks' nodig om de elementen goed weer te geven in de DOM (*Internet Explorer*) en de nieuwe elementen te kunnen opmaken met CSS (alle browsers).



het **hoppinger**▲ html5 handboek

Bijlagen



B1 - Compatibiliteit browsers

B1.1 - Opera 10.62

COMPATIBILITY

Deze browser ondersteund **33/45 (73%)** van de nieuwe HTML5 functionaliteiten.

Web Applications

✓ Local Storage (a.k.a. Web Storage)	✓ Session Storage (a.k.a. Web Storage)	✓ Application Cache (a.k.a. Offline Storage)	✗ Microdata
✗ Drag and Drop	✓ Geolocation API		

Embedded Content

✓ Canvas	✓ CanvasText	✓ Video	✓ Audio
VIDEO CODECS			
✓ ogg	✗ H.264	✓ WebM	
AUDIO CODECS			
✓ ogg	✗ mp3	✓ wav	✗ ACC

Web Forms

FORM INPUT TYPES

✓ Email	✓ URL	✗ Telephone	✓ Number
✓ Number (Range)	✗ Search	✗ Colour	✓ Date
✓ Month	✓ Week	✓ Time	✓ Datetime
✓ Datetime-Local			

FORM ATTRIBUTES

✗ Placeholder	✓ Autofocus	✓ Autocomplete ***	✓ Required
✓ List	✓ Pattern	✗ Mutiple	✓ Min
✓ Max	✓ Step		

FORM ELEMENTS

✓ Output	✗ Meter *	✗ Progress *	✓ Datalist
✓ Keygen */***			

LEGENDA

- ✗ Functionaliteit wordt **NIET** ondersteund door de browser.
- ✓ Functionaliteit wordt **WEL** ondersteund door de browser.

OPMERKINGEN

- * Functionaliteit is op het ogenblik niet te testen. Ondersteuning is gebaseerd op eigen bevindingen.
- ** Testuitslag van Modernizr komt niet overeen met werkelijkheid. Ondersteuning is gebaseerd op eigen bevindingen.
- *** Functionaliteit is niet besproken in het onderzoek.



B1.2 - Chrome 6.0.472.63

COMPATIBILITY

Deze browser ondersteund **32/45 (71%)** van de nieuwe HTML5 functionaliteiten.

Web Applications

✓ Local Storage (a.k.a. Web Storage)	✓ Session Storage (a.k.a. Web Storage)	✓ Application Cache (a.k.a. Offline Storage)	✗ Microdata
✓ Drag and Drop	✓ Geolocation API		

Embedded Content

✓ Canvas	✓ CanvasText	✓ Video	✓ Audio
----------	--------------	---------	---------

VIDEO CODECS

✓ ogg	✓ H.264	✓ WebM
-------	---------	--------

AUDIO CODECS

✓ ogg	✓ mp3	✗ wav	✓ ACC
-------	-------	-------	-------

Web Forms

FORM INPUT TYPES

✓ Email	✓ URL	✓ Telephone	✗ Number
✓ Number (Range)	✓ Search	✗ Colour	✗ Date
✗ Month	✗ Week	✗ Time	✗ Datetime
✓ Datetime-Local			

FORM ATTRIBUTES

✓ Placeholder	✓ Autofocus	✗ Autocomplete ***	✓ Required
✗ List	✓ Pattern	✓ Mutiple	✓ Min
✓ Max	✓ Step		

FORM ELEMENTS

✗ Output	✓ Meter *	✓ Progress *	✗ Datalist
✓ Keygen */***			

LEGENDA

- ✗ Functionaliteit wordt **NIET** ondersteund door de browser.
- ✓ Functionaliteit wordt **WEL** ondersteund door de browser.

OPMERKINGEN

* Functionaliteit is op het ogenblik niet te testen. Ondersteuning is gebaseerd op eigen bevindingen.

** Testuitslag van Modernizr komt niet overeen met werkelijkheid. Ondersteuning is gebaseerd op eigen bevindingen.

*** Functionaliteit is niet besproken in het onderzoek.



B1.3 - Safari 4.0.5 (531.22.7)

COMPATIBILITY

Deze browser ondersteund **17/45 (38%)** van de nieuwe HTML5 functionaliteiten.

Web Applications

<input checked="" type="checkbox"/> Local Storage <small>(a.k.a. Web Storage)</small>	<input checked="" type="checkbox"/> Session Storage <small>(a.k.a. Web Storage)</small>	<input checked="" type="checkbox"/> Application Cache <small>(a.k.a. Offline Storage)</small>	<input type="checkbox"/> Microdata
<input checked="" type="checkbox"/> Drag and Drop	<input type="checkbox"/> Geolocation API		

Embedded Content

<input checked="" type="checkbox"/> Canvas	<input checked="" type="checkbox"/> CanvasText	<input checked="" type="checkbox"/> Video	<input checked="" type="checkbox"/> Audio
--	--	---	---

VIDEO CODECS

<input type="checkbox"/> ogg	<input checked="" type="checkbox"/> H.264	<input type="checkbox"/> WebM
------------------------------	---	-------------------------------

AUDIO CODECS

<input type="checkbox"/> ogg	<input checked="" type="checkbox"/> mp3	<input checked="" type="checkbox"/> wav	<input checked="" type="checkbox"/> ACC
------------------------------	---	---	---

Web Forms

FORM INPUT TYPES

<input type="checkbox"/> Email	<input type="checkbox"/> URL	<input checked="" type="checkbox"/> Telephone	<input type="checkbox"/> Number
<input type="checkbox"/> Number (Range)	<input checked="" type="checkbox"/> Search	<input type="checkbox"/> Colour	<input type="checkbox"/> Date
<input type="checkbox"/> Month	<input type="checkbox"/> Week	<input type="checkbox"/> Time	<input type="checkbox"/> Datetime
<input type="checkbox"/> Datetime-Local			

FORM ATTRIBUTES

<input checked="" type="checkbox"/> Placeholder	<input checked="" type="checkbox"/> Autofocus	<input type="checkbox"/> Autocomplete ***	<input type="checkbox"/> Required
<input type="checkbox"/> List	<input type="checkbox"/> Pattern	<input checked="" type="checkbox"/> Mutiple	<input type="checkbox"/> Min
<input type="checkbox"/> Max	<input type="checkbox"/> Step		

FORM ELEMENTS

<input type="checkbox"/> Output	<input type="checkbox"/> Meter *	<input type="checkbox"/> Progress *	<input type="checkbox"/> Datalist
<input type="checkbox"/> Keygen */***			

LEGENDA

- Functionaliteit wordt **NIET** ondersteund door de browser.
- Functionaliteit wordt **WEL** ondersteund door de browser.

OPMERKINGEN

- * Functionaliteit is op het ogenblik niet te testen. Ondersteuning is gebaseerd op eigen bevindingen.
- ** Testuitslag van Modernizr komt niet overeen met werkelijkheid. Ondersteuning is gebaseerd op eigen bevindingen.
- *** Functionaliteit is niet besproken in het onderzoek.



B1.4 - Firefox 3.6.10

COMPATIBILITY

Deze browser ondersteund **14/45 (31%)** van de nieuwe HTML5 functionaliteiten.

Web Applications

✓ Local Storage (a.k.a. Web Storage)	✓ Session Storage (a.k.a. Web Storage)	✓ Application Cache (a.k.a. Offline Storage)	✗ Microdata
✓ Drag and Drop	✓ Geolocation API		

Embedded Content

✓ Canvas	✓ CanvasText	✓ Video	✓ Audio
----------	--------------	---------	---------

VIDEO CODECS

✓ ogg	✗ H.264	✗ WebM
-------	---------	--------

AUDIO CODECS

✓ ogg	✗ mp3	✓ wav	✗ ACC
-------	-------	-------	-------

Web Forms

FORM INPUT TYPES

✗ Email	✗ URL	✗ Telephone	✗ Number
✗ Number (Range)	✗ Search	✗ Colour	✗ Date
✗ Month	✗ Week	✗ Time	✗ Datetime
✗ Datetime-Local			

FORM ATTRIBUTES

✗ Placeholder	✗ Autofocus	✗ Autocomplete ***	✗ Required
✗ List	✗ Pattern	✓ Mutiple	✗ Min
✗ Max	✗ Step		

FORM ELEMENTS

✗ Output	✗ Meter *	✗ Progress *	✗ Datalist
✓ Keygen */***			

LEGENDA

- ✗ Functionaliteit wordt **NIET** ondersteund door de browser.
- ✓ Functionaliteit wordt **WEL** ondersteund door de browser.

OPMERKINGEN

- * Functionaliteit is op het ogenblik niet te testen. Ondersteuning is gebaseerd op eigen bevindingen.
- ** Testuitslag van Modernizr komt niet overeen met werkelijkheid. Ondersteuning is gebaseerd op eigen bevindingen.
- *** Functionaliteit is niet besproken in het onderzoek.



B1.5 - Internet Explorer 9

COMPATIBILITY

Deze browser ondersteund **8/45 (18%)** van de nieuwe HTML5 functionaliteiten.

Web Applications

<input checked="" type="checkbox"/> Local Storage <small>(s.k.s. Web Storage)</small>	<input checked="" type="checkbox"/> Session Storage <small>(s.k.s. Web Storage)</small>	<input type="checkbox"/> Application Cache <small>(s.k.s. Offline Storage)</small>	<input type="checkbox"/> Microdata
<input checked="" type="checkbox"/> Drag and Drop	<input type="checkbox"/> Geolocation API **		

Embedded Content

<input checked="" type="checkbox"/> Canvas	<input checked="" type="checkbox"/> CanvasText	<input checked="" type="checkbox"/> Video	<input checked="" type="checkbox"/> Audio
--	--	---	---

VIDEO CONTAINERS

<input type="checkbox"/> ogg	<input type="checkbox"/> H.264	<input type="checkbox"/> WebM
------------------------------	--------------------------------	-------------------------------

AUDIO CONTAINERS

<input type="checkbox"/> ogg	<input checked="" type="checkbox"/> mp3	<input type="checkbox"/> wav	<input type="checkbox"/> acc *
------------------------------	---	------------------------------	--------------------------------

Web Forms

FORM INPUT TYPES

<input type="checkbox"/> Email	<input type="checkbox"/> URL	<input type="checkbox"/> Telephone	<input type="checkbox"/> Number
<input type="checkbox"/> Number (Range)	<input type="checkbox"/> Search	<input type="checkbox"/> Colour	<input type="checkbox"/> Date
<input type="checkbox"/> Month	<input type="checkbox"/> Week	<input type="checkbox"/> Time	<input type="checkbox"/> Datetime
<input type="checkbox"/> Datetime-Local			

FORM ATTRIBUTES

<input type="checkbox"/> Placeholder	<input type="checkbox"/> Autofocus	<input type="checkbox"/> Autocomplete ***	<input type="checkbox"/> Required
<input type="checkbox"/> List	<input type="checkbox"/> Pattern	<input type="checkbox"/> Mutiple	<input type="checkbox"/> Min
<input type="checkbox"/> Max	<input type="checkbox"/> Step		

FORM ELEMENTS

<input type="checkbox"/> Output	<input type="checkbox"/> Meter *	<input type="checkbox"/> Progress *	<input type="checkbox"/> Datalist
<input type="checkbox"/> Keygen */***			

LEGENDA

Functionaliteit wordt **NIET** ondersteund door de browser.

Functionaliteit wordt **WEL** ondersteund door de browser.

OPMERKINGEN

* Functionaliteit is op het ogenblik niet te testen. Ondersteuning is gebaseerd op eigen bevindingen.

** Testuitslag van Modernizr komt niet overeen met werkelijkheid. Ondersteuning is gebaseerd op eigen bevindingen.

*** Functionaliteit is niet besproken in het onderzoek.



B1.6 - Internet Explorer 8.0.6001

COMPATIBILITY

Deze browser ondersteund **3/45 (7%)** van de nieuwe HTML5 functionaliteiten.

Web Applications

<input checked="" type="checkbox"/> Local Storage (s.k.a. Web Storage)	<input checked="" type="checkbox"/> Session Storage (s.k.a. Web Storage)	<input type="checkbox"/> Application Cache (s.k.a. Offline Storage)	<input type="checkbox"/> Microdata
<input checked="" type="checkbox"/> Drag and Drop	<input type="checkbox"/> Geolocation API **		

Embedded Content

<input type="checkbox"/> Canvas	<input type="checkbox"/> CanvasText	<input type="checkbox"/> Video	<input type="checkbox"/> Audio
---------------------------------	-------------------------------------	--------------------------------	--------------------------------

VIDEO CODECS

<input type="checkbox"/> ogg	<input type="checkbox"/> H.264	<input type="checkbox"/> WebM
------------------------------	--------------------------------	-------------------------------

AUDIO CODECS

<input type="checkbox"/> ogg	<input type="checkbox"/> mp3	<input type="checkbox"/> wav	<input type="checkbox"/> ACC**
------------------------------	------------------------------	------------------------------	--------------------------------

Web Forms

FORM INPUT TYPES

<input type="checkbox"/> Email	<input type="checkbox"/> URL	<input type="checkbox"/> Telephone	<input type="checkbox"/> Number
<input type="checkbox"/> Number (Range)	<input type="checkbox"/> Search	<input type="checkbox"/> Colour	<input type="checkbox"/> Date
<input type="checkbox"/> Month	<input type="checkbox"/> Week	<input type="checkbox"/> Time	<input type="checkbox"/> Datetime
<input type="checkbox"/> Datetime-Local			

FORM ATTRIBUTES

<input type="checkbox"/> Placeholder	<input type="checkbox"/> Autofocus	<input type="checkbox"/> Autocomplete ***	<input type="checkbox"/> Required
<input type="checkbox"/> List	<input type="checkbox"/> Pattern	<input type="checkbox"/> Mutiple	<input type="checkbox"/> Min
<input type="checkbox"/> Max	<input type="checkbox"/> Step		

FORM ELEMENTS

<input type="checkbox"/> Output	<input type="checkbox"/> Meter *	<input type="checkbox"/> Progress *	<input type="checkbox"/> Datalist
<input type="checkbox"/> Keygen */***			

LEGENDA

Functionaliteit wordt **NIET** ondersteund door de browser.

Functionaliteit wordt **WEL** ondersteund door de browser.

OPMERKINGEN

* Functionaliteit is op het ogenblik niet te testen. Ondersteuning is gebaseerd op eigen bevindingen.

** Testuitslag van Modernizr komt niet overeen met werkelijkheid. Ondersteuning is gebaseerd op eigen bevindingen.

*** Functionaliteit is niet besproken in het onderzoek.



B1.7 - iPhone Browser & iPad Browser

Toestel: iPhone 3GS – iPad (Model MC349NF)

OS-versie: 4.1 – 3.2.2

Browserversie: Safari v.[?]

COMPATIBILITY			
Deze browser ondersteund (42%) van de nieuwe HTML5 functionaliteiten.			
Web Applications			
<input checked="" type="checkbox"/> Local Storage <small>(s.k.a. Web Storage)</small>	<input checked="" type="checkbox"/> Session Storage <small>(s.k.a. Web Storage)</small>	<input checked="" type="checkbox"/> Application Cache <small>(s.k.a. Offline Storage)</small>	<input checked="" type="checkbox"/> Microdata
<input checked="" type="checkbox"/> Drag and Drop **	<input checked="" type="checkbox"/> Geolocation API		
Embedded Content			
<input checked="" type="checkbox"/> Canvas	<input checked="" type="checkbox"/> CanvasText	<input checked="" type="checkbox"/> Video	<input checked="" type="checkbox"/> Audio
VIDEO CODECS			
<input checked="" type="checkbox"/> ogg	<input checked="" type="checkbox"/> H.264	<input checked="" type="checkbox"/> WebM	
AUDIO CODECS			
<input checked="" type="checkbox"/> ogg	<input checked="" type="checkbox"/> mp3	<input checked="" type="checkbox"/> wav	<input checked="" type="checkbox"/> ACC
Web Forms			
FORM INPUT TYPES			
<input checked="" type="checkbox"/> Email	<input checked="" type="checkbox"/> URL	<input checked="" type="checkbox"/> Telephone	<input checked="" type="checkbox"/> Number **
<input checked="" type="checkbox"/> Number (Range) **	<input checked="" type="checkbox"/> Search	<input checked="" type="checkbox"/> Colour	<input checked="" type="checkbox"/> Date
<input checked="" type="checkbox"/> Month	<input checked="" type="checkbox"/> Week	<input checked="" type="checkbox"/> Time	<input checked="" type="checkbox"/> Datetime
<input checked="" type="checkbox"/> Datetime-Local **			
FORM ATTRIBUTES			
<input checked="" type="checkbox"/> Placeholder	<input checked="" type="checkbox"/> Autofocus **	<input checked="" type="checkbox"/> Autocomplete ***	<input checked="" type="checkbox"/> Required
<input checked="" type="checkbox"/> List	<input checked="" type="checkbox"/> Pattern	<input checked="" type="checkbox"/> Mutiple **	<input checked="" type="checkbox"/> Min
<input checked="" type="checkbox"/> Max	<input checked="" type="checkbox"/> Step		
FORM ELEMENTS			
<input checked="" type="checkbox"/> Output	<input checked="" type="checkbox"/> Meter *	<input checked="" type="checkbox"/> Progress *	<input checked="" type="checkbox"/> Datalist
<input checked="" type="checkbox"/> Keygen */***			
LEGENDA			
<input checked="" type="checkbox"/> Functionaliteit wordt NIET ondersteund door de browser.			
<input checked="" type="checkbox"/> Functionaliteit wordt WEL ondersteund door de browser.			
OPMERKINGEN			
* Functionaliteit is op het ogenblik niet te testen. Ondersteuning is gebaseerd op eigen bevindingen.			
** Testuitslag van Modernizr komt niet overeen met werkelijkheid. Ondersteuning is gebaseerd op eigen bevindingen.			
*** Functionaliteit is niet besproken in het onderzoek.			



B1.8 - Android Browser

Toestel: HTC Desire

Android-versie: 2.2 (Froyo)

Browsersversie: WebKit 3.1

COMPATIBILITY

Deze browser ondersteund (1%) van de nieuwe HTML5 functionaliteiten.

Web Applications

<input checked="" type="checkbox"/> Local Storage (s.k.s. Web Storage)	<input checked="" type="checkbox"/> Session Storage (s.k.s. Web Storage)	<input checked="" type="checkbox"/> Application Cache (s.k.s. Offline Storage)	<input type="checkbox"/> Microdata
<input type="checkbox"/> Drag and Drop **	<input checked="" type="checkbox"/> Geolocation API		

Embedded Content

<input checked="" type="checkbox"/> Canvas	<input checked="" type="checkbox"/> CanvasText	<input checked="" type="checkbox"/> Video	<input checked="" type="checkbox"/> Audio
--	--	---	---

VIDEO CODECS

<input type="checkbox"/> ogg	<input type="checkbox"/> H.264	<input type="checkbox"/> WebM
------------------------------	--------------------------------	-------------------------------

AUDIO CODECS

<input type="checkbox"/> ogg	<input type="checkbox"/> mp3	<input type="checkbox"/> wav	<input type="checkbox"/> ACC
------------------------------	------------------------------	------------------------------	------------------------------

Web Forms

FORM INPUT TYPES

<input type="checkbox"/> Email	<input type="checkbox"/> URL	<input checked="" type="checkbox"/> Telephone	<input type="checkbox"/> Number
<input type="checkbox"/> Number (Range)	<input checked="" type="checkbox"/> Search	<input type="checkbox"/> Colour	<input type="checkbox"/> Date
<input type="checkbox"/> Month	<input type="checkbox"/> Week	<input type="checkbox"/> Time	<input type="checkbox"/> Datetime
<input type="checkbox"/> Datetime-Local **			

FORM ATTRIBUTES

<input checked="" type="checkbox"/> Placeholder	<input type="checkbox"/> Autofocus **	<input type="checkbox"/> Autocomplete ***	<input checked="" type="checkbox"/> Required
<input type="checkbox"/> List	<input checked="" type="checkbox"/> Pattern	<input type="checkbox"/> Mutiple **	<input type="checkbox"/> Min **
<input type="checkbox"/> Max **	<input type="checkbox"/> Step **		

FORM ELEMENTS

<input type="checkbox"/> Output	<input type="checkbox"/> Meter *	<input type="checkbox"/> Progress *	<input type="checkbox"/> Datalist
<input checked="" type="checkbox"/> Keygen */***			

LEGENDA

- Functionaliteit wordt **NIET** ondersteund door de browser.
- Functionaliteit wordt **WEL** ondersteund door de browser.

OPMERKINGEN

* Functionaliteit is op het ogenblik niet te testen. Ondersteuning is gebaseerd op eigen bevindingen.

** Testuitslag van Modernizr komt niet overeen met werkelijkheid. Ondersteuning is gebaseerd op eigen bevindingen.

*** Functionaliteit is niet besproken in het onderzoek.



B1.9 – Opera Mini

Toestel: HTC Desire

Android-versie: 2.2 (Froyo)

Browsersversie: 5.1.2.1126

COMPATIBILITY

0) van de nieuwe HTML5 functionaliteiten.

Web Applications

<input type="checkbox"/> Local Storage (s.k.s. Web Storage)	<input type="checkbox"/> Session Storage (s.k.s. Web Storage)	<input type="checkbox"/> Application Cache (s.k.s. Offline Storage)	<input type="checkbox"/> Microdata
<input type="checkbox"/> Drag and Drop **	<input type="checkbox"/> Geolocation API		

Embedded Content

<input checked="" type="checkbox"/> Canvas	<input checked="" type="checkbox"/> CanvasText	<input type="checkbox"/> Video	<input type="checkbox"/> Audio
--	--	--------------------------------	--------------------------------

VIDEO CODECS

<input type="checkbox"/> ogg	<input type="checkbox"/> H.264	<input type="checkbox"/> WebM
------------------------------	--------------------------------	-------------------------------

AUDIO CODECS

<input type="checkbox"/> ogg	<input type="checkbox"/> mp3	<input type="checkbox"/> wav	<input type="checkbox"/> ACC
------------------------------	------------------------------	------------------------------	------------------------------

Web Forms

FORM INPUT TYPES

<input type="checkbox"/> Email	<input type="checkbox"/> URL	<input type="checkbox"/> Telephone	<input type="checkbox"/> Number
<input type="checkbox"/> Number (Range)	<input type="checkbox"/> Search	<input type="checkbox"/> Colour	<input type="checkbox"/> Date
<input type="checkbox"/> Month	<input type="checkbox"/> Week	<input type="checkbox"/> Time	<input type="checkbox"/> Datetime
<input type="checkbox"/> Datetime-Local			

FORM ATTRIBUTES

<input type="checkbox"/> Placeholder	<input type="checkbox"/> Autofocus	<input type="checkbox"/> Autocomplete ***	<input type="checkbox"/> Required
<input type="checkbox"/> List	<input type="checkbox"/> Pattern	<input type="checkbox"/> Mutiple	<input type="checkbox"/> Min
<input type="checkbox"/> Max	<input type="checkbox"/> Step		

FORM ELEMENTS

<input type="checkbox"/> Output	<input type="checkbox"/> Meter *	<input type="checkbox"/> Progress *	<input type="checkbox"/> Datalist
<input checked="" type="checkbox"/> Keygen */***			

LEGENDA

- Functionaliteit wordt **NIET** ondersteund door de browser.
- Functionaliteit wordt **WEL** ondersteund door de browser.

OPMERKINGEN

* Functionaliteit is op het ogenblik niet te testen. Ondersteuning is gebaseerd op eigen bevindingen.

** Testuitslag van Modernizr komt niet overeen met werkelijkheid. Ondersteuning is gebaseerd op eigen bevindingen.

*** Functionaliteit is niet besproken in het onderzoek.



B2 - XHTML2

Zoals in hoofdstuk één is te lezen werd de *Web Hypertext Application Technology Working Group (WHATWG)* opgericht, nadat bleek dat de *World Wide Web Consortium (W3C)* geen interesse had in wat nu *HTML5* is. Een reden hiervoor was dat de W3C toen al twee jaar bezig was met de ontwikkeling van *XHTML2*: ***“XHTML 2 is a general purpose markup language designed for representing documents for a wide range of purposes across the World Wide Web. To this end it does not attempt to be all things to all people, supplying every possible markup idiom, but to supply a generally useful set of elements [...]”***¹

In de bovenstaande quote staat het grote verschil tussen *XHTML2* en *HTML5* en het komt neer op één woord: documents. De W3C is met hun producten gefocust om documenten weer te geven, zoals men dit sinds de intrede van *HTML* heeft gedaan. Onder de term documenten wordt het volgende bedoelt: teksten en/of afbeeldingen, die op een statische manier worden weergegeven. De informatie staat er en de eindgebruiker kan er verder weinig mee doen. Echter het gebruik van het web is veranderd en de gebruikers eisen steeds meer dynamiek en interactiviteit. Een goed voorbeeld hiervan zijn bijvoorbeeld *Flash* producenten. De WHATWG heeft ingezien dat deze verandering plaatsvindt en willen zich met *HTML5* juist focussen op web applications: ***“XHTML 2 was designed to reform the Web as a medium for publishing documents, but the developers--and the browser makers who listened closely to those developers--instead wanted a platform for interactive applications.”***²

B2.1 - Design Aims

In de working draft van *XHTML2* worden de hoofddoelen aangegeven waar de W3C tijdens de ontwikkeling mee werkt:³

- Gebruik maken van functionaliteiten van *XML* en niet onnodig de werking ervan dupliceren in *XHTML*.
- Met *XHTML* meer richten op de structuur en minder op de weergave.
- Het schrijven van code makkelijker maken en het gebruiken van het resultaat makkelijker maken.
- De toegankelijkheid van de code moet verbeterd worden, zodat toekomstige ontwerpers geen problemen ermee zullen ondervinden.
- Internationalisering van de techniek, aangezien het internet en webontwikkeling op een wereldwijze basis gebeurt.
- Het moet niet langer nodig zijn om voor verschillende platformen (desktop en mobiele) meerdere documenten te schrijven. Het doel is om elk platform te bereiken met één document.
- Functionaliteiten, die alleen behaald konden worden met behulp van scripts (bv. *JavaScript*) en die vaak gebruikt worden, af laten handelen door markup opgenomen in *XHTML2*.
- Functionaliteiten aanbieden, die de semantiek van een webpagina verbeteren en daardoor de leesbaarheid van de webpagina vergroten.

Het is interessant om op te merken dat de laatste drie doelen ook opgenomen zijn in *HTML5*. Zo zijn in hoofdstuk twee de nieuwe semantische elementen besproken en zijn ook een aantal form attributen langsgesproken waarmee huidige *JavaScript* oplossingen onnodig worden. De *iPad* is een goed voorbeeld van het feit dat men met *HTML5* zowel desktop als mobiele platformen wil bereiken. Maar uit deze doelen is wederom op te halen dat *XHTML2* zich vooral richt op het weergeven van documenten. Punt twee en acht getuigen hiervan; het gaat om de structuur van het document en niet om de weergave.



B2.2 - Ontwikkeling XHTML2 stopgezet

Vanaf 2004 kwam er echter een concurrent van XHTML2 om de hoek kijken, met de oprichting van de WHATWG. Vanaf dat moment was XHTML2 gedoemd om te falen. Hoe kan het toevoegen van het href attribuut in elk element opboksen tegen nieuwe HTML5 functionaliteiten zoals canvas en video? Ontwikkelaars stonden te springen voor nieuwe uitbreidingen en dit leverde de W3C met XHTML2 niet.: *"People were so focused on XHTML 2 that they were substantially less interested in modifying the application model and introducing new features to HTML that developers were clamoring for [...] We felt the standards going on at the time...were disconnected from a large majority of developers."*²

De W3C zag uiteindelijk in dat het doorontwikkelen van XHTML2 geen nut had. Geen enkele grote browser had namelijk nog moeite gedaan om ondersteuning voor de nieuwe XHTML versie te bieden. Bovendien was XHTML2 niet achterwaarts compatibel met zijn voorgangers en eiste van ontwerpers dat ze websites compleet moesten herschrijven. In 2006 werd besloten om de ontwikkeling stop te zetten en de krachten met de WHATWG te bundelen, voor de doorontwikkeling van HTML. Het is vanaf dat moment dat de losse specificaties ('Web Apps 1.0' en 'Web Forms 2.0') van de WHATWG, samengevoegd werden tot één; HTML5.

Een interessante opmerking hierbij is dat HTML5 ook gedeeltelijk als XHTML5 kan worden beschouwd. Het is immers zo dat HTML5 achterwaarts compatibel is met zowel HTML als XHTML. Daardoor kan de ontwikkelaar zijn documenten opmaken zoals hij/zij gewend is: zowel in HTML of in XHTML syntax. Slashes aan het eind van inhoudsloze elementen (dit zijn elementen die geen content bevatten en ook geen sluit tag hebben. Voorbeelden: en
) en aanhalingstekens rondom attribuut waarden zijn niet langer verplicht. Maar het maakt voor HTML5 echter niet uit als de ontwikkelaar dit wel zou doen. Er kan geconcludeerd worden dat de ontwikkelaar er zelf voor kiest in hoeverre zijn/haar producten XHTML zijn.

B2.3 - Veranderingen in XHTML2

De meeste veranderingen in XHTML2 zullen hoogstwaarschijnlijk nooit het daglicht zien. Het is toch interessant om een aantal van deze veranderingen toe te lichten. Dit om te vergelijken met identieke functionaliteiten uit HTML5 of om aan te geven welke richting de W3C in wilde gaan.

B2.3.1 - <h> en <section> element

Voorheen werden de <h1> t/m <h6> elementen gebruikt om bijvoorbeeld titels en subtitels aan te geven. Hierdoor werd er een hiërarchie in de pagina gecreëerd: <h1> staat het hoogst in 'aanzien', terwijl <h6> onderaan komt. Volgens de W3C snapte niet elke ontwikkelaar dit en gaf een eigen functionaliteit aan de elementen: "[...] a problem when authors misused the heading elements for visual effect"⁴

De oplossing waarmee de W3C is gekomen zijn de <h> en <section> elementen. Hiermee wordt de inhoud waar een header (<h>) naar refereert op een zelfde manier aangegeven. De hiërarchie is niet langer gelimiteerd tot de headers, maar op de inhoud in zijn compleetheid. In HTML5 is het gebruik van <h1> t/m <h6> onveranderd, maar is het <hgroup> element geïntroduceerd om bijvoorbeeld titel en subtitel samen te voegen. Het verschil tussen het <section> element is dat het in HTML5 gebruikt wordt om onafhankelijke onderdelen af te bakenen. In XHTML2 wordt het gebruikt om een hiërarchische structuur mee aan te geven.

```
<h> Titel van de site </h> <!-- Gelijk aan <h1> -->
<p> Een intro tekst</p>
<section>
  <h> Titel van een nieuwsartikel </h> <!-- Gelijk aan <h2> -->
  <p> Inhoud van het nieuwsartikel</p>
  <section>
    <h> Titel van de comments </h> <!-- Gelijk aan <h3> -->
    <p> Inhoud van de comments </p>
  </section>
</section>
```



B2.3.2 - <l> element

Om een ruimte te creëren tussen een tweetal regels werd tot nu voornamelijk gebruik gemaakt van het
 element. In XHTML2 zou hiervoor het <l> element (letter 'L') worden geïntroduceerd waarmee een zin tekst aangegeven kan worden. Tussen elke zin zou een ruimte, equivalent aan dat van het
 element, worden gecreëerd. Het zou echter ook voor de volgende functionaliteiten gebruikt kunnen worden: *"Amongst other advantages, this gives more presentational opportunities, such as the ability to automatically number lines, or to color alternate lines differently."*⁴

```
<h> Haiku4 </h>
<l> Ach oude vijver</l>
<l> een kikker springt er in</l>
<l> geluid van water. </l>
```

B2.3.3 - <nl> element

Dit is een uitbreiding van de al bestaande lijst functionaliteit en is bedoelt om de navigatie van een webpagina aan te geven: *"One obvious component of very many HTML pages is the 'navigation list', consisting of a collection of links to other parts of the site, presented vertically, horizontally, or as a drop-down menu. To support this type of usage, XHTML 2 introduces the navigation list element nl, which codifies such parts of documents, and allows different presentational idioms to be applied."*⁴

Daarnaast zijn er overeenkomsten met het <nav> element in HTML5 te bekennen: doordat het element een specifieke functie heeft kunnen browsers en andere software hier beter mee omgaan: screenreaders en soortgelijke software kunnen de inhoud van dit element bijvoorbeeld overslaan.

```
<nl>
  <label>Hoofd navigatie:</label> <!-- het <nl> element eist een <label> element -->
  <li>Startpagina</li>
  <li>Producten</li>
  <li>Diensten</li>
  <li>Features</li>
</nl>
```

B2.3.4 - href attribuut

In XHTML2 zou het mogelijk geweest zijn om in elk element een href attribuut toe te voegen. De ontwikkelaar is niet langer gelimiteerd aan het gebruik van het <a> element om te linken naar een andere bron. Het voordeel is duidelijk: in plaats van twee verschillende elementen hoeft er nog maar één gebruikt te worden.

We nemen als voorbeeld een afbeelding:

```
<a href="/images/logobig.png">  </a>
```

```
 Logo </img>
<!-- Het alt attribuut verdwijnt en het img is niet langer meer een void element. De omschrijving van de afbeelding komt tussen de tags te staan. -->
```

B2.3.5 - role attribuut

Het role attribuut zou gebruikt kunnen worden om een bepaalde rol/functionaliiteit van een element aan te geven: *"In order to aid adding semantics to documents, the role attribute has been added, along with an initial set of useful values, in order to classify the use of a particular element."*⁴ Als dit attribuut vergeleken wordt met HTML5, zou microdata het meest in de buurt komen. Daarmee kan immers ook een bepaalde rol van een zinsdeel aangegeven worden. Het role attribuut zou tevens dezelfde voordelen kunnen opleveren als microdata: *"The role attribute can add new semantics and metadata to existing elements, helping search engines and assistive technologies better process Web pages."*⁶



```
<nl role="breadcrumbs">5
  <label>You are here:</label>
  <li>Home</li>
  <li>Products</li>
  <li>Widgit</li>
  <li>Features</li>
</nl>
```

B2.3.6 - Verwijderde elementen

Eén van de doelen van de W3C was om de structuur en opmaak van een document te scheiden. Met deze gedachte zijn er een aantal elementen verwijderd, die specifiek gebruikt konden worden om een document mee op te maken. De elementen met een asterisk zijn ook niet opgenomen in de *HTML5* specificaties.

- ``
- `<i>`
- `<small>`
- `<big>` *
- `<tt>` *
- `` *
- `<basefont>` *

Naast bovenstaande elementen zouden ook de volgende elementen niet langer meer opgenomen worden in *XHTML2*:

- `<acronym>`
- `<iframe>`

Bronnen

¹ : XHTML 2.0 W3C Working Draft | http://www.w3.org/TR/xhtml2/introduction.html#s_intro_whatixhtml2 | 1.1. What is XHTML 2? | 26-07-2006 | 11-10-2010

² : XHTML 2.0 W3C Working Draft | http://www.w3.org/TR/xhtml2/introduction.html#s_intro_whatixhtml2 | 1.1.1. Design Aims | 26-07-2006 | 11-10-2010

² : An epitaph for the Web standard, XHTML 2 | http://news.cnet.com/8301-17939_109-10281477-2.html | Stephen Shankland | 08-07-2009 | 11-10-2010

³ : XHTML 2.0 W3C Working Draft | http://www.w3.org/TR/xhtml2/introduction.html#s_intro_whatixhtml2 | 1.2. Major Differences with XHTML 1 | 26-07-2006 | 11-10-2010

⁵ : Haiku (dichtvorm) | [http://nl.wikipedia.org/wiki/Haiku_\(dichtvorm\)](http://nl.wikipedia.org/wiki/Haiku_(dichtvorm)) | Dichtvorm | 28-09-2010 | 11-10-2010

⁶ : XHTML 5 Versus XHTML 2 | <http://xhtml.com/en/future/x-html-5-versus-xhtml-2/#x2> | XHTML 2 -What's Cool About XHTML 2 - Ability To Add Additional Semantics To Existing Elements | ??-??-2007/2008 | 12-10-2010



B3 - 'Thoughts on Flash'

Apple has a long relationship with Adobe. In fact, we met Adobe's founders when they were in their proverbial garage. Apple was their first big customer, adopting their Postscript language for our new Laserwriter printer. Apple invested in Adobe and owned around 20% of the company for many years. The two companies worked closely together to pioneer desktop publishing and there were many good times. Since that golden era, the companies have grown apart. Apple went through its near death experience, and Adobe was drawn to the corporate market with their Acrobat products. Today the two companies still work together to serve their joint creative customers – Mac users buy around half of Adobe's Creative Suite products – but beyond that there are few joint interests.

I wanted to jot down some of our thoughts on Adobe's Flash products so that customers and critics may better understand why we do not allow Flash on iPhones, iPods and iPads. Adobe has characterized our decision as being primarily business driven – they say we want to protect our App Store – but in reality it is based on technology issues. Adobe claims that we are a closed system, and that Flash is open, but in fact the opposite is true. Let me explain.

First, there's "Open".

Adobe's Flash products are 100% proprietary. They are only available from Adobe, and Adobe has sole authority as to their future enhancement, pricing, etc. While Adobe's Flash products are widely available, this does not mean they are open, since they are controlled entirely by Adobe and available only from Adobe. By almost any definition, Flash is a closed system.

Apple has many proprietary products too. Though the operating system for the iPhone, iPod and iPad is proprietary, we strongly believe that all standards pertaining to the web should be open. Rather than use Flash, Apple has adopted HTML5, CSS and JavaScript – all open standards. Apple's mobile devices all ship with high performance, low power implementations of these open standards. HTML5, the new web standard that has been adopted by Apple, Google and many others, lets web developers create advanced graphics, typography, animations and transitions without relying on third party browser plugins (like Flash). HTML5 is completely open and controlled by a standards committee, of which Apple is a member.

Apple even creates open standards for the web. For example, Apple began with a small open source project and created WebKit, a complete open-source HTML5 rendering engine that is the heart of the Safari web browser used in all our products. WebKit has been widely adopted. Google uses it for Android's browser, Palm uses it, Nokia uses it, and RIM (Blackberry) has announced they will use it too. Almost every smartphone web browser other than Microsoft's uses WebKit. By making its WebKit technology open, Apple has set the standard for mobile web browsers.

Second, there's the "full web".

Adobe has repeatedly said that Apple mobile devices cannot access "the full web" because 75% of video on the web is in Flash. What they don't say is that almost all this video is also available in a more modern format, H.264, and viewable on iPhones, iPods and iPads. YouTube, with an estimated 40% of the web's video, shines in an app bundled on all Apple mobile devices, with the iPad offering perhaps the best YouTube discovery and viewing experience ever. Add to this video from Vimeo, Netflix, Facebook, ABC, CBS, CNN, MSNBC, Fox News, ESPN, NPR, Time, The New York Times, The Wall Street Journal, Sports Illustrated, People, National Geographic, and many, many others. iPhone, iPod and iPad users aren't missing much video.

Another Adobe claim is that Apple devices cannot play Flash games. This is true. Fortunately, there are over 50,000 games and entertainment titles on the App Store, and many of them are free. There are more games and entertainment titles available for iPhone, iPod and iPad than for any other platform in the world.



Third, there's reliability, security and performance.

Symantec recently highlighted Flash for having one of the worst security records in 2009. We also know first hand that Flash is the number one reason Macs crash. We have been working with Adobe to fix these problems, but they have persisted for several years now. We don't want to reduce the reliability and security of our iPhones, iPods and iPads by adding Flash.

In addition, Flash has not performed well on mobile devices. We have routinely asked Adobe to show us Flash performing well on a mobile device, any mobile device, for a few years now. We have never seen it. Adobe publicly said that Flash would ship on a smartphone in early 2009, then the second half of 2009, then the first half of 2010, and now they say the second half of 2010. We think it will eventually ship, but we're glad we didn't hold our breath. Who knows how it will perform?

Fourth, there's battery life.

To achieve long battery life when playing video, mobile devices must decode the video in hardware; decoding it in software uses too much power. Many of the chips used in modern mobile devices contain a decoder called H.264 – an industry standard that is used in every Blu-ray DVD player and has been adopted by Apple, Google (YouTube), Vimeo, Netflix and many other companies.

Although Flash has recently added support for H.264, the video on almost all Flash websites currently requires an older generation decoder that is not implemented in mobile chips and must be run in software. The difference is striking: on an iPhone, for example, H.264 videos play for up to 10 hours, while videos decoded in software play for less than 5 hours before the battery is fully drained.

When websites re-encode their videos using H.264, they can offer them without using Flash at all. They play perfectly in browsers like Apple's Safari and Google's Chrome without any plugins whatsoever, and look great on iPhones, iPods and iPads.

Fifth, there's Touch.

Flash was designed for PCs using mice, not for touch screens using fingers. For example, many Flash websites rely on "rollovers", which pop up menus or other elements when the mouse arrow hovers over a specific spot. Apple's revolutionary multi-touch interface doesn't use a mouse, and there is no concept of a rollover. Most Flash websites will need to be rewritten to support touch-based devices. If developers need to rewrite their Flash websites, why not use modern technologies like HTML5, CSS and JavaScript?

Even if iPhones, iPods and iPads ran Flash, it would not solve the problem that most Flash websites need to be rewritten to support touch-based devices.

Sixth, the most important reason.

Besides the fact that Flash is closed and proprietary, has major technical drawbacks, and doesn't support touch based devices, there is an even more important reason we do not allow Flash on iPhones, iPods and iPads. We have discussed the downsides of using Flash to play video and interactive content from websites, but Adobe also wants developers to adopt Flash to create apps that run on our mobile devices.

We know from painful experience that letting a third party layer of software come between the platform and the developer ultimately results in sub-standard apps and hinders the enhancement and progress of the platform. If developers grow dependent on third party development libraries and tools, they can only take advantage of platform enhancements if and when the third party chooses to adopt the new features. We cannot be at the mercy of a third party deciding if and when they will make our enhancements available to our developers.



This becomes even worse if the third party is supplying a cross platform development tool. The third party may not adopt enhancements from one platform unless they are available on all of their supported platforms. Hence developers only have access to the lowest common denominator set of features. Again, we cannot accept an outcome where developers are blocked from using our innovations and enhancements because they are not available on our competitor's platforms.

Flash is a cross platform development tool. It is not Adobe's goal to help developers write the best iPhone, iPod and iPad apps. It is their goal to help developers write cross platform apps. And Adobe has been painfully slow to adopt enhancements to Apple's platforms. For example, although Mac OS X has been shipping for almost 10 years now, Adobe just adopted it fully (Cocoa) two weeks ago when they shipped CS5. Adobe was the last major third party developer to fully adopt Mac OS X.

Our motivation is simple – we want to provide the most advanced and innovative platform to our developers, and we want them to stand directly on the shoulders of this platform and create the best apps the world has ever seen. We want to continually enhance the platform so developers can create even more amazing, powerful, fun and useful applications. Everyone wins – we sell more devices because we have the best apps, developers reach a wider and wider audience and customer base, and users are continually delighted by the best and broadest selection of apps on any platform.

Conclusions.

Flash was created during the PC era – for PCs and mice. Flash is a successful business for Adobe, and we can understand why they want to push it beyond PCs. But the mobile era is about low power devices, touch interfaces and open web standards – all areas where Flash falls short.

The avalanche of media outlets offering their content for Apple's mobile devices demonstrates that Flash is no longer necessary to watch video or consume any kind of web content. And the 250,000 apps on Apple's App Store proves that Flash isn't necessary for tens of thousands of developers to create graphically rich applications, including games.

New open standards created in the mobile era, such as HTML5, will win on mobile devices (and PCs too). Perhaps Adobe should focus more on creating great HTML5 tools for the future, and less on criticizing Apple for leaving the past behind.

Steve Jobs
April, 2010¹

Bronnen

¹ : Thoughts on Flash | <http://www.apple.com/hotnews/thoughts-on-flash/> | | Steve Jobs | ??-04-2010 | 03-11-2010



Bronnenlijst

Aanleiding

1. HTML5 & CSS3 Support | <http://www.findmebyip.com/litmus#target-selector> | HTML5 Web Applications t/m HTML5 Forms Attributes
2. HTML5 (including next generation additions still in development) Draft Standard | <http://www.whatwg.org/specs/web-apps/current-work/multipage/> | Status of this document

Hoofdstuk 1

1. Hypertext Markup Language (HTML) Internet Draft | <http://www.w3.org/MarkUp/draft-ietf-iiir-html-01.txt>
2. A Brief History of Markup | <http://www.alistapart.com/articles/a-brief-history-of-markup/> | XHTML 1: HTML as XML
3. HTML5: Up and Running | Hfst. 1 : How Did We Get Here? – Everything You Know About XHTML Is Wrong
4. The W3C Workshop on Web Applications and Compound Documents | <http://www.w3.org/2004/04/webapps-cdf-ws/>
5. Position Paper for the W3C Workshop on Web Applications and Compound Documents | <http://www.w3.org/2004/04/webapps-cdf-ws/papers/opera.html> | Design Principles for Web Application Technologies
6. Summary of The W3C Workshop on Web Applications and Compound Documents | <http://www.w3.org/2004/04/webapps-cdf-ws/summary> | Next steps
7. WHAT open mailing list announcement | <http://www.whatwg.org/news/start>
8. Web Applications 1.0 Early Working Draft | <http://www.whatwg.org/specs/web-apps/2005-09-01/>
9. HTML5 (including next generation additions still in development) Draft Standard | <http://www.whatwg.org/specs/web-apps/current-work/>
10. HTML 5 Editor Ian Hickson discusses features, pain points, adoption rate, and more | <http://blogs.techrepublic.com.com/programming-and-development/?p=718>

Hoofdstuk 2

1. The HTML5 Semantics Debate | <http://visitmix.com/opinions/The-HTML5-Semantics-Debate> | Tags are Too Final
2. HTML 5 Reference | <http://html5tutorial.net/html-5-reference/html-5-reference.html>
3. HTML5 (including next generation additions still in development) Draft Standard | <http://www.whatwg.org/specs/web-apps/current-work/multipage/sections.html#the-header-element> | 4.4.8 The header element
4. HTML5: Up and Running | Hfst. 3 : What Does It All mean? – Headers
5. HTML5 (including next generation additions still in development) Draft Standard | <http://www.whatwg.org/specs/web-apps/current-work/multipage/sections.html#the-nav-element> | 4.4.3 The nav element
6. HTML5: Up and Running | Hfst. 3 : What Does It All mean? – Navigation
7. HTML5 (including next generation additions still in development) Draft Standard | <http://www.whatwg.org/specs/web-apps/current-work/multipage/sections.html#the-section-element> | 4.4.2 The section element
8. HTML5 (including next generation additions still in development) Draft Standard | <http://www.whatwg.org/specs/web-apps/current-work/multipage/sections.html#the-article-element> | 4.4.4 The article element
9. HTML5 (including next generation additions still in development) Draft Standard | <http://www.whatwg.org/specs/web-apps/current-work/multipage/sections.html#the-aside-element> | 4.4.5 The aside element
10. HTML5 (including next generation additions still in development) Draft Standard | <http://www.whatwg.org/specs/web-apps/current-work/multipage/sections.html#the-footer-element> | 4.4.9 The footer element
11. HTML5 (including next generation additions still in development) Draft Standard | <http://www.whatwg.org/specs/web-apps/current-work/multipage/sections.html#the-hgroup-element> | 4.4.7 The hgroup element
12. HTML5 (including next generation additions still in development) Draft Standard | <http://www.whatwg.org/specs/web-apps/current-work/multipage/sections.html#the-address-element> | 4.4.10 The address element
13. HTML5 (including next generation additions still in development) Draft Standard | <http://www.whatwg.org/specs/web-apps/current-work/multipage/sections.html#the-address-element> | 4.4.10 The address element
14. 28 HTML5 Features, Tips, and Techniques you Must Know | <http://net.tutsplus.com/tutorials/html-css-techniques/25-html5-features-tips-and-techniques-you-must-know/> | 22. Mark Element



15. HTML5 (including next generation additions still in development) Draft Standard | <http://www.whatwg.org/specs/web-apps/current-work/multipage/grouping-content.html#the-figure-element> | 4.4.11 The figure element
16. 28 HTML5 Features, Tips, and Techniques you Must Know | <http://net.tutsplus.com/tutorials/html-css-techniques/25-html5-features-tips-and-techniques-you-must-know/> | 2. The Figure Element
17. WHAT open mailing list announcement | <http://www.whatwg.org/news/start>
18. HTML5: Up and Running | Hfst. 8 : A Form of Madness? – Email Addresses
19. Default To The Numeric, Email, And URL Keyboards On The iPhone | <http://www.bennadel.com/blog/1721-Default-To-The-Numeric-Email-And-URL-Keyboards-On-The-iPhone.htm>
20. Coordinated Universal Time | http://en.wikipedia.org/wiki/Coordinated_Universal_Time
21. HTML5 (including next generation additions still in development) Draft Standard | <http://www.whatwg.org/specs/web-apps/current-work/multipage/common-input-element-attributes.html#the-multiple-attribute> | 4.10.7.2.6 The multiple attribute
22. HTML5 (including next generation additions still in development) Draft Standard | <http://www.whatwg.org/specs/web-apps/current-work/multipage/the-button-element.html#the-output-element> | 4.10.15 The output element
23. HTML5 (including next generation additions still in development) Draft Standard | <http://www.whatwg.org/specs/web-apps/current-work/multipage/the-button-element.html#the-progress-element> | 4.10.16 The progress element
24. HTML5 (including next generation additions still in development) Draft Standard | <http://www.whatwg.org/specs/web-apps/current-work/multipage/the-button-element.html#the-meter-element> | 4.10.17 The meter element
25. Dive into HTML5 | <http://diveintohtml5.org/forms.html#validation> | №9. A Form of Madness - Form Validation
26. TED.com now available in HTML5, serving many mobile platforms, including iPhone, iPad | <http://www.wired.com/epicenter/2010/07/youtube-launches-new-html5-mobile-site/>
27. YouTube Launches New HTML5 Mobile Site | <http://www.wired.com/epicenter/2010/07/youtube-launches-new-html5-mobile-site/>
28. HTML5 (including next generation additions still in development) Draft Standard | <http://www.whatwg.org/specs/web-apps/current-work/multipage/the-canvas-element.html#the-canvas-element> | 4.8.11 The canvas element
29. HTML5 (including next generation additions still in development) Draft Standard | <http://www.whatwg.org/specs/web-apps/current-work/multipage/links.html#microdata> | 5 Microdata
30. RDFa Primer Bridging the Human and Data Webs | <http://www.w3.org/TR/xhtml-rdfa-primer/>
31. HTML5: Up and Running | Hfst. 10 : “Distributed,” “Extensibility,” and Other Fancy Words – Marking Up People
32. Web Storage Editor's Draft | <http://dev.w3.org/html5/webstorage/>
33. HTTP cookie | http://en.wikipedia.org/wiki/HTTP_cookie
34. HTML5: Up and Running | Hfst. 7 : The Past, Present, and Future of Local Storage for Web Applications
35. Hackers stelen Twitter-cookies via XSS-lek | http://www.security.nl/artikel/34382/1/Hackers_stelen_Twitter-cookies_via_XSS-lek.html
36. Web Storage Editor's Draft | <http://dev.w3.org/html5/webstorage/#implementation-risks> | 7.3 Implementation risks
37. HTML5 (including next generation additions still in development) Draft Standard | <http://www.whatwg.org/specs/web-apps/current-work/#offline> | 6.6.1 Introduction
38. Hixie's Natural Log | <http://ln.hixie.ch/?start=1115899732&count=1> | Beneath The Surface
39. Geolocation API Specification W3C Candidate Recommendation | <http://www.w3.org/TR/geolocation-API/>
40. Geolocation API Specification W3C Candidate Recommendation | <http://www.w3.org/TR/geolocation-API/#usecases> | 6 Use-Cases and Requirements

Hoofdstuk 3

1. HTML5 Cross browser Polyfills | <http://github.com/Modernizr/Modernizr/wiki/HTML5-Cross-browser-Polyfills>
2. HTML5: Up and Running | Hfst. 3 : What Does It All mean? – A Long Digression into How Browsers Handle Unknown Elements
3. HTML5 IE enabling script | <http://code.google.com/p/html5shiv/>
4. HTML5 New Input Types | http://www.w3schools.com/html5/tryit.asp?filename=tryhtml5_form_number | TryIt Editor – Number input type
5. Browser Statistics | http://www.w3schools.com/browsers/browsers_stats.asp
6. The HTML5 Test – How well does your browser support HTML5? | <http://html5test.com/>
7. Microdata support for Rich Snippets | <http://googlewebmastercentral.blogspot.com/2010/03/microdata-support-for-rich-snippets.html>
8. HTML5 & CSS3 Support | <http://www.findmebyip.com/litmus#target-selector> | HTML5 Web Applications



9. HTML5 is not backward compatible | <http://mattwilcox.net/archive/entry/id/957/> | And by way of clarifying why HTML5 is a dead donkey for the foreseeable future

Hoofdstuk 4

1. Top Ten Search Engines - Top 10 SEs | <http://www.seoconsultants.com/search-engines/> | Search Engine Statistics - What Market Share?
2. SEO Starter Guide | <http://www.google.com/webmasters/docs/search-engine-optimization-starter-guide.pdf>
3. SEO implications of Page Segmentation concepts | <http://www.huomah.com/Search-Engines/Search-Engine-Optimization/SEO-implications-of-Page-Segmentation-concepts.html> | How search engines could get granular
4. SEO implications of Page Segmentation concepts | <http://www.huomah.com/Search-Engines/Search-Engine-Optimization/SEO-implications-of-Page-Segmentation-concepts.html> | Segmenting the page
5. Google: meta keywords & description niet gebruikt in ranking | <http://www.edwords.nl/2009/09/21/google-meta-keywords-meta-description-tag-niet-gebruikt-ranking/> | Waarom negeert Google de meta keywords tag?
6. The time element (and microformats) | <http://html5doctor.com/the-time-element/> | Where could time be used?
7. HTML5 Microdata: Welcome to the Machine | <http://net.tutsplus.com/tutorials/html-css-techniques/html5-microdata-welcome-to-the-machine/>
8. SEO Debate: HTML vs. Flash | <http://www.rickydeez.com/web-design/seo-debate-html-vs-flash/> | SEO / Search Engine Hindrances

Hoofdstuk 5

1. Thoughts on Flash | <http://www.apple.com/hotnews/thoughts-on-flash/>
2. HTML5 Overview | http://www.tutorialspoint.com/html5/html5_overview.htm
3. Adobe Flash Player | <http://www.adobe.com/products/flashplayer/>
4. Adobe CTO Kevin Lynch Defends Flash, Warns HTML5 Will Throw The Web "Back To The Dark Ages Of Video" | <http://techcrunch.com/2010/02/02/adobe-cto-kevin-lynch-defends-flash/>
5. Flash Player: CPU Hog or Hot Tamale? It Depends. | <http://www.streaminglearningcenter.com/articles/flash-player-cpu-hog-or-hot-tamale-it-depends-.html>
6. Adobe Releases Flash 10.1 Beta with Hardware Acceleration | <http://www.macrumors.com/2010/04/29/adobe-releases-flash-10-1-beta-with-hardware-acceleration/>
7. GUIMark 2: The rise of HTML5 | <http://www.craftymind.com/guimark2/>
8. GUIMark 2: The rise of HTML5 | <http://www.craftymind.com/guimark2/> | Vector Charting Test
9. GUIMark 2: The rise of HTML5 | <http://www.craftymind.com/guimark2/> | Bitmap Gaming Test
10. GUIMark 2: The rise of HTML5 | <http://www.craftymind.com/guimark2/> | Text Column Test
11. GUIMark 2: The rise of HTML5 | <http://www.craftymind.com/guimark2/#comment-22037> | Comment #8
12. GUIMark 2: The rise of HTML5 | <http://www.craftymind.com/guimark2/#comment-22212> | Comment #73
13. GUIMark 2: The rise of HTML5 | <http://www.craftymind.com/guimark2/> | GUIMark Mobile
13. Best uses of Flash | <http://googlewebmastercentral.blogspot.com/2007/07/best-uses-of-flash.html>
14. SEO Optimization and JavaScript | <http://www.webmarketingart.com/search-engine-optimization/JavaScript-and-seo/>
15. How to SEO Flash | <http://www.hochmanconsultants.com/articles/seo-friendly-flash.shtml> | Example: Making Flash Home Page Spiderable

Hoofdstuk 6

1. iPad ready | <http://www.apple.com/ipad/ready-for-ipad/> | | | 07-10-2010
2. TED.com now available in HTML5, serving many mobile platforms, including iPhone, iPad | http://blog.ted.com/2010/03/31/tedcom_now_avai/
3. YouTube Launches New HTML5 Mobile Site | <http://www.wired.com/epicenter/2010/07/youtube-launches-new-html5-mobile-site/>
4. Hello HTML5 | http://gearsblog.blogspot.com/2010/02/hello-html5.html?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed%3A+GearsApiBlog+%28Gears+API+Blog%29
5. W3C Markup Validation Service | http://validator.w3.org/#validate_by_uri+with_options | Document Type Validator.nu (X)HTML5 Validator (Highly Experimental) | <http://html5.validator.nu/>
6. Browser Statistics | http://www.w3schools.com/browsers/browsers_stats.asp | Browser Statistics Month by Month – September 2010
7. 2010 Browser Stats and Trends | <http://nicholasbailey.blogspot.com/2010/06/2010-browser-stats-and-trends.html> | IE users are upgrading
8. 2010 Browser Stats and Trends | <http://nicholasbailey.blogspot.com/2010/06/2010-browser-stats-and-trends.html> | The hidden chaos: StatCounter Global Stats



Bijlage 2

1. XHTML 2.0 W3C Working Draft | http://www.w3.org/TR/xhtml2/introduction.html#s_intro_whatixhtml2 | 1.1. What is XHTML 2?
2. XHTML 2.0 W3C Working Draft | http://www.w3.org/TR/xhtml2/introduction.html#s_intro_whatixhtml2 | 1.1.1. Design Aims
3. An epitaph for the Web standard, XHTML 2 | http://news.cnet.com/8301-17939_109-10281477-2.html
4. XHTML 2.0 W3C Working Draft | http://www.w3.org/TR/xhtml2/introduction.html#s_intro_whatixhtml2 | 1.2. Major Differences with XHTML 1
5. Haiku (dichtvorm) | [http://nl.wikipedia.org/wiki/Haiku_\(dichtvorm\)](http://nl.wikipedia.org/wiki/Haiku_(dichtvorm)) | Dichtvorm
6. X/HTML 5 Versus XHTML 2 | <http://xhtml.com/en/future/x-html-5-versus-xhtml-2/#x2> | XHTML 2 -What's Cool About XHTML 2 - Ability To Add Additional Semantics To Existing Elements

Bijlage 3

1. Thoughts on Flash | <http://www.apple.com/hotnews/thoughts-on-flash/>