

Web Development 1

Deel 1: HTML5

C. De Waele

PBA toegepaste informatica 1

Het digitaal cursusmateriaal is enkel bestemd voor persoonlijk gebruik door de gebruiker. Het is de gebruiker niet toegestaan om de digitale cursus geheel of gedeeltelijk te kopiëren, verkopen, verdelen, doorgeven, vertalen, verspreiden, weergeven, reproduceren, publiceren, verhuren, leasen, onderlicentiëren, vergunnen met licentie of op andere wijze over te dragen.

Web development I – HTML5

auteur: **Christophe De Waele**

studiegebied **HWB**

bachelor in het **toegepaste Informatica**

campus **Kortrijk**

academiejaar **2024-2025**

Legende van de gebruikte iconen

	Denkvraag		Studeeraanwijzingen
	Leerdoelen		Tijdsinschatting
	Formule		Toledo
	Extra informatie		Beeld-/geluidsfragment
	Niet vergeten		Voorbeeld
	Opdracht/Oefening		Tools/Apps
	Presentatie (PowerPoint)		Website
	Rekenblad (Excel)		Zelfstudie
	Samenwerking		(Zelf)toets

Inhoudsopgave

Legende van de gebruikte iconen	2
Inhoudsopgave	3
1 Inleiding.....	6
2 HTML 5	9
2.1 Waarom HTML5?	9
2.2 De structuur van een HTML5-document	11
2.2.1 Opbouw van HTML	12
2.2.2 Doctype en <html> element	12
3 HTML-Elementen	13
3.1 Element-regels	13
3.1.1 Alle tagnamen met kleine letters schrijven?	13
3.1.2 Alle tags afsluiten?	13
3.1.3 Alle waarden van attributen tussen aanhalingstekens plaatsen	14
3.1.4 Alle elementen moeten correct genest zijn	14
3.1.5 Commentaren correct schrijven	15
3.1.6 De tekens & en < correct coderen	15
3.2 Deprecated HTML-elementen	16
3.3 Elementen binnen de <head>-tag	18
3.3.1 <title> element	18
3.3.2 <meta> element	19
3.3.3 <link> element	20
3.4 Elementen binnen de <body>-tag	20
3.4.1 Heading	21
3.4.2 Gestructureerde tekst	21
3.4.3 Lijsten	24
3.4.4 Semantische elementen	26
3.4.5 Voorbeeld van HTML5-pagina	34
3.4.6 Samenvatting	36
3.5 Een expliciete outline met HTML5	37

3.5.1	<i>Document outline en sectioning content</i>	37
3.6	Tabellen	41
3.7	Hyperlinks en anchors.....	45
3.7.1	<i>Tekeningen</i>	46
4	HTML5 valideren	48
4.1	Onlinevalidatie	48
5	Formulieren	49
5.1	Formulierobjecten	50
5.1.1	<i>Tekstvak</i>	50
5.1.2	<i>Paswoordvelden</i>	50
5.1.3	<i>Email Address & Web Address</i>	50
5.1.4	<i>Number</i>	51
5.1.5	<i>Data en tijden invoeren</i>	52
5.1.6	<i>Searchbox</i>	52
5.1.7	<i>Tekstvak met meerdere regels</i>	52
5.1.8	<i>Checkbox</i>	53
5.1.9	<i>Radiobutton</i>	53
5.1.10	<i>Keuzelijst</i>	54
5.1.11	<i>Verborgen velden</i>	54
5.2	Nieuwe attributen voor <input> element	54
5.2.1	<i>Autofocus</i>	54
5.2.2	<i>Required</i>	55
5.2.3	<i>Placeholder</i>	55
5.2.4	<i>List</i>	55
5.2.5	<i>Multiple</i>	56
5.2.6	<i>Pattern</i>	56
5.3	Formulieren verzenden	57
5.4	Voorbeeld van een formulier.....	58
6	HTML5-API's	60
6.1	Video	60
6.2	Audio.....	65

6.3	Canvas	66
6.4	Web Storage	66
6.5	Drag and Drop.....	68
6.6	Geolocation.....	68

1 Inleiding

HTML staat voor HyperText Markup Language en is de taal waarin de inhoud van webpagina's wordt gestructureerd.

Voordat we ons vastpinnen op het heden is het goed om eerst in onze achterrauitkijkspiegel te kijken. Ons onmiddellijk verdiepen in HTML5 zou niet verstandig zijn. Er is namelijk nog geen enkele browser die alle functionaliteiten van HTML5 ondersteunt.

Parsing rules	5	Video	29/33
<!DOCTYPE html> triggers standards mode	Yes ✓	video element	Yes ✓
HTML5 tokenizer	Yes ✓	Subtitles	Yes ✓
HTML5 tree building	Yes ✓	Audio track selection	No ✗
HTML5 defines rules for embedding SVG and MathML inside a regular HTML document. The following tests only check if the browser is following the HTML5 parsing rules for inline SVG and MathML, not if the browser can actually understand and render it.		Video track selection	No ✗
Parsing inline SVG	Yes ✓	Poster images	Yes ✓
Parsing inline MathML	Yes ✓	Codec detection	Yes ✓
Elements	30/33	Video codecs	
Embedding custom non-visible data	Yes ✓	MPEG-4 ASP support	No ✗
MathML support	No ✗	H.264 support	Yes ✓
New or modified elements		H.265 support	Yes ✓
▶ Section elements	Yes ✓	Ogg Theora support	No ✗
▶ Grouping content elements	Yes ✓	WebM with VP8 support	Yes ✓
▶ Text-level semantic elements	Yes ✓	WebM with VP9 support	Yes ✓
▶ Interactive elements	Yes ✓	WebM with AV1 support	Yes ✓
Global attributes or methods		Audio	29/30
hidden attribute	Yes ✓	audio element	Yes ✓
translate attribute	Yes ✓	Loop audio	Yes ✓
accessKey attribute	Yes ✓	Preload in the background	Yes ✓
accessKeyLabel attribute	No ✗	Advanced	
▶ Dynamic markup insertion	Yes ✓	Web Audio API	Yes ✓
Forms	66	Speech Recognition	Prefixed ✓
Field types		Speech Synthesis	Yes ✓
▶ input type=text	Yes ✓	Audio codecs	
▶ input type=search	Yes ✓	PCM audio support	Yes ✓
		MP3 support	Yes ✓
		AAC support	Yes ✓
		Dolby Digital support	No ✗
		Dolby Digital Plus support	No ✗

Figuur 1 Voorbeeld van Chrome¹

De specificaties zoals ze nu staan geschreven kunnen nog gemakkelijk veranderen. Dit wordt ook nadrukkelijk aangegeven door de Web Hypertext Application Technology Working Group (WHATWG). Laten we even terugkeren naar het begin.

Het web bestaat sinds 1991. Tim Berners-Lee (nog steeds president van het W3C²) is de bedenker ervan en daarmee de eerste “webdesigner”. Nadat hij de specificaties voor de webserver, webbrowser en opmaaktaal (HTML) had vrijgegeven, begon zijn idee snel vleugels te krijgen. Bekende namen uit die tijd waren Mosaic, Netscape en Chameleon.

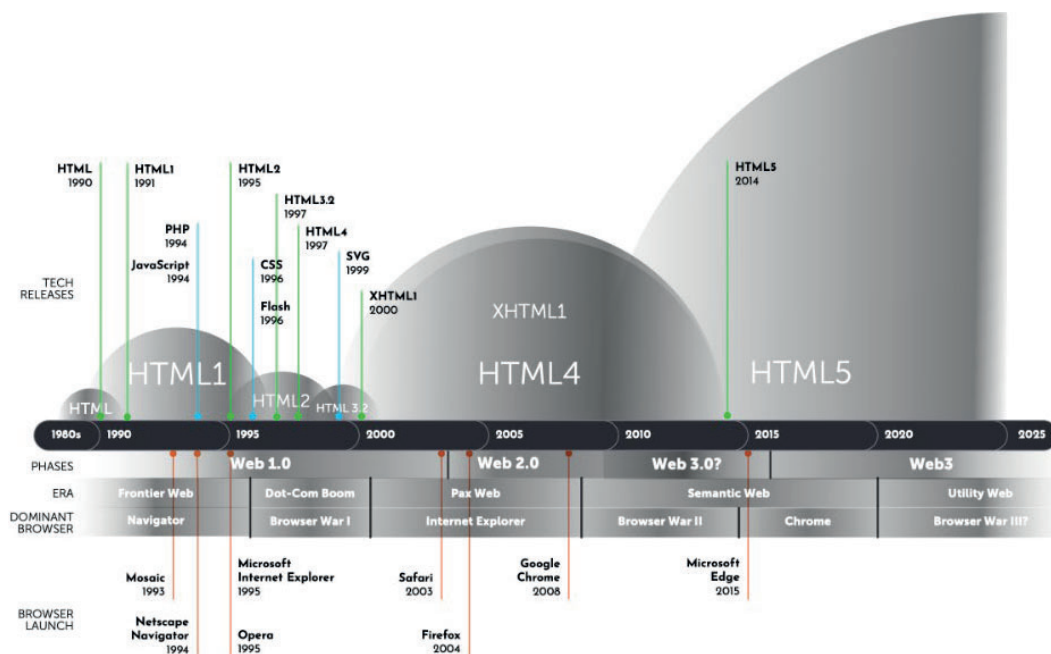
Toen Microsoft in 1995 Internet Explorer (IE) meeleverde met Windows 95 barstte de browser oorlog uit. De technische ontwikkelingen rondom HTML midden de jaren 90 werden voornamelijk gestuurd

¹ <https://html5test.co/>

² World Wide Web Consortium

door browserfabrikanten en community van webdesigners. Als een fabrikant een nieuwtje introduceerde in een nieuwe versie en deze werden omarmd door de webdesigners dan haastte de concurrent om dit ook te ondersteunen. Kortom, er was een wildgroei van tags.

HTML is de meeste verspreide UI-technologie maar de browsers verschillen zeer sterk om mee te werken.



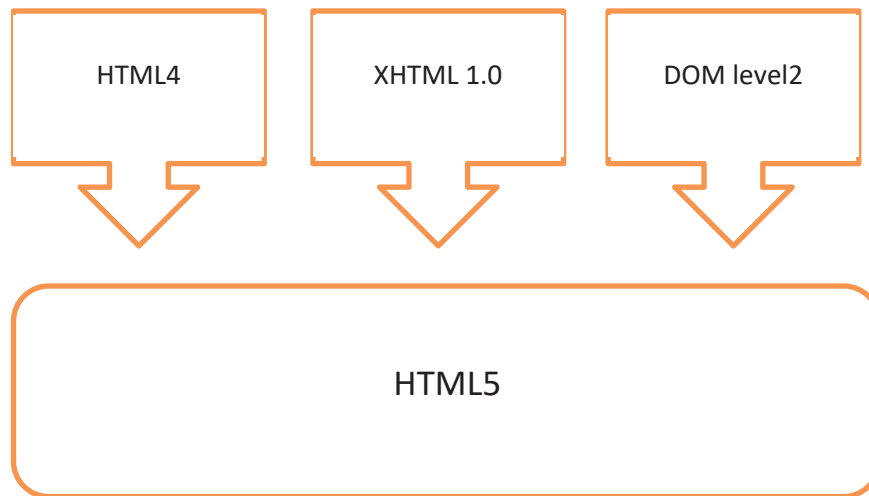
Figuur 2 Overzicht van versies van HTML doorheen de tijd

Op het einde van 1997 werd HTML 4.0 op de markt gebracht en verscheen er nog een update, nl. 4.01 in 1999. Vanaf dat moment kwam de ontwikkeling van HTML tot een halt. De reden voor het stopzetten van de ontwikkeling van HTML kwam door de ontevredenheid van het W3C. Volgens deze organisatie had de ontwikkelaar te veel 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”³.** In 2001 kwam W3C met de opvolger, onder de naam XHTML 1.1. In maart 2007 werd de werkgroep voor XHTML 2.0 officieel geïnstalleerd. Het was de bedoeling dat deze groep een taal zou ontwikkelen die voor het eerst niet achterwaarts compatibel zou zijn met de oude HTML-standaard. XHTML 2.0 moest volledig modulair van opbouw worden. Maar het werk vlotte niet en eind 2009 werd de *XHTML 2.0 Working Group* door W3C opgeheven.

Na de standaardisering van XHTML 1.0 in 2000, werd een verdere ontwikkeling van HTML richting XML wenselijk geacht. Toch was er veel weerstand tegen. Een groot struikelblok was dat XHTML 2.0 niet backwards compatible zou zijn. In 2004 werd daarom WHATWG (Web Hypertext Application Technology Working Group) opgericht. Dit is een groep van ontwikkelaars en architecten die

³ A Brief History of Markup, Jeremy Keith (<https://alistapart.com/article/a-brief-history-of-markup/>)

ontevreden waren met de richting van het W3C. Zij begonnen met een nieuwe standaard die voortborduurt op HTML 4 en uiteindelijk zou uitmonden in wat nu HTML5 is.



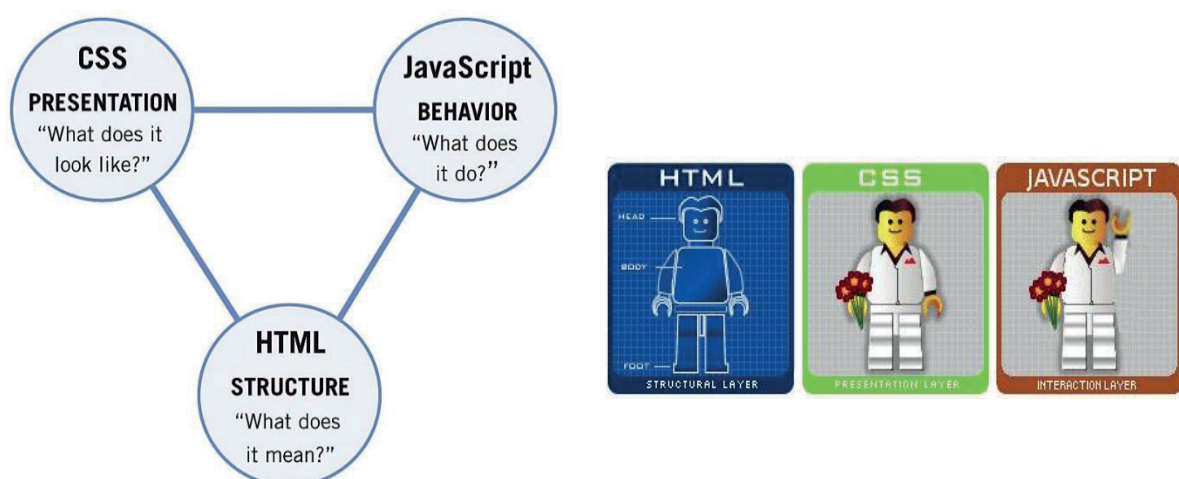
Figuur 3 Prerequisites HTML5

Het behouden van achterwaartse compatibiliteit en tegelijkertijd het introduceren van nieuwe, verbeterde structuurelementen zou HTML weer klaar moeten maken voor de toekomst. Het W3C, dat eerst niets van een opvolger van HTML 4 wilde weten, werkt nu samen met de WHATWG aan HTML 5.

Als je aan de slag gaat met het bouwen van sites is het belangrijk te ontleden uit welke componenten een basispagina bestaat. Deze zijn achtereenvolgens:

- HTML5 en XML voor het structureren van gegevens.
- CSS/XSLT voor het presenteren (opmaken) van gegevens.
- JavaScript (ECMAScript) voor interactieve items en gedrag (behaviour) voor gegevens.

Figuur 4 geeft de relatie tussen deze standaarden weer, wat essentieel is om te onthouden.



Figuur 4 Relatie tussen de Web standaarden

In deze cursus wordt de eerste pijler besproken. We snijden eerst de principes van HTML5 aan. Na deze cursus worden de twee andere pijlers besproken (zie cursussen CSS en JavaScript).

2 HTML 5

2.1 Waarom HTML5?



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



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 websites volledig gemaakt met HTML5 als paddenstoelen uit de grond schieten (voor een showcase kan je online terecht⁴). Met de huidige ontwikkelingen rond HTML5 lijkt het erop een technologie is die zal blijven. De kans dat het de nieuwe web standaard zal worden is groot, de vraag is enkel nog *wanneer*. 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*⁵ zal worden. Veel geïnteresseerden in het onderwerp concludeerden dat het geen nut zou hebben om zich 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. 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. Een belangrijke mijlpaal voor HTML5 werd op 17 december 2012 bereikt. Het World Wide Web Consortium (W3C) publiceerde de volledige definitie van de HTML5 en Canvas 2D specificaties in een candidate recommendation.

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

- Webapplicaties moeten gemaakt kunnen worden met technieken, die bekend zijn bij ontwikkelaars (*HTML*, *CSS*, *DOM* en *JavaScript*). Het gebruik van externe plugins (denk aan de oude Flash player) om applicaties te laten werken moet voorkomen worden.
- Error afhandeling moet gedetailleerd gedefinieerd worden om te voorkomen dat browsers zelf mechanismen moeten bedenken.
- Gebruikers van webapplicaties mogen niks merken van fouten die zijn gemaakt door ontwikkelaars.

⁴ <http://html5gallery.com>

⁵ Wanneer de techniek is zijn volledigheid geïmplementeerd is door twee verschillende browsers.

- Alle functies die opgenomen worden in de specificaties van webapplicaties moeten een praktisch en functioneel doel dienen.
- Webapplicaties moeten op elk apparaat en in elke vorm van presentatie identiek zijn.
- Webapplicaties moeten dezelfde functionaliteiten hebben in zowel de desktop als mobiele versie.
- Het ontwikkeltraject van webapplicaties 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. Het W3C zag meer brood in versie 2.0 van *XHTML*. De partijen die de presentatie hadden gegeven verenigden zich en zo ontstond WHATWG die uiteindelijk de ontwikkeling van HTML5 op de schouders heeft genomen.

Globaal kunnen de vernieuwingen rond HTML5 in de volgende categorieën worden verdeeld:

- **Betekenisvolle paginastructuur (semantische elementen):** nieuwe tags moedigen aan om meer betekenis in de pagina aan te brengen. Waar voorheen secties in de pagina m.b.v. div-tags werden gescheiden, zijn er nu nieuwe tags om bijvoorbeeld headers, footers, artikelen, navigatieblokken en afbeeldingen in op te nemen.
- **Multimedia:** voor het weergeven van beeld en geluid hebben browsers traditioneel plug-ins nodig. Flash Player bijvoorbeeld of de Windows Media Player-plug-in. In HTML5 zijn tags opgenomen waarmee direct video kan worden afgespeeld.
- **Animatie:** er is een nieuw element met de naam <canvas> opgenomen, waarmee je een tekengebied in de browser definieert. In de meest eenvoudige vorm teken je hierin lijnen, vormen en vullingen maar ook complete spelletjes zijn mogelijk. Je kan het vergelijken met Flash of Silverlight. Het verschil is dat deze aanvullende technieken niet nodig zijn en de browser het helemaal alleen af kan. Je moet wel thuis zijn in JavaScript, want dat is de motor achter <canvas>. Daarom wordt dit element nog niet besproken in deze cursus.
- **Gegevensopslag:** je kent ongetwijfeld cookies, de kleine tekstbestandjes die op jouw computer worden opgeslagen waarmee websites gegevens registreren. Cookies hebben allerlei nadelen en HTML5 introduceerde nieuwe, betere manieren om gegevens te bewaren. Web Storage en Web SQL Database. Wil je zelf meer weten, bekijk zeker de sites van w3⁶ en w3schools⁷.
- **Offline webapplicaties:** in essentie komt het erop neer dat je in de broncode aangeeft welke bestanden nodig zijn om de site of de applicatie zonder Internettoegang te laten werken. Terwijl je online bent, worden alle noodzakelijke bestanden lokaal opgeslagen, zodat deze ook beschikbaar zijn als er geen internetverbinding is. Deze topic valt buiten het bereik van deze cursus.

In feite is HTML5 geen taal voor het opmaken van webpagina's, maar een taal waarmee documenten gestructureerd worden. De bedoeling is dat je na het doorlopen van deze cursus inzicht hebt in de

⁶ <http://dev.w3.org/html5/webstorage>

⁷ http://www.w3schools.com/html5/html5_webstorage.asp

basissyntax en structuur van HTML5-documenten. HTML5 lijkt erg veel op HTML 4.0. Op het gebied van syntax is er weinig verschil, er komen voornamelijk extra componenten bij. Maar er zijn nog andere verschillen:

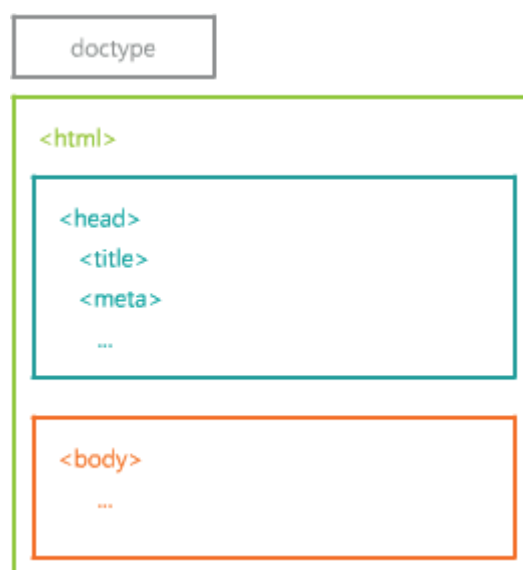
- Mogelijkheid om tekeningen te laten genereren door de browser zelf in plaats van dure serveroplossingen.
- Intuïtieve websites door standaard *drag-and-drop* functionaliteit.
- Foutafhandeling wordt geregeld in de standaard zelf en niet overgelaten aan de browser. Dit bevordert de consistentie in het gedrag van de browser onderling.
- Er zijn tal van API's (Application Programming Interfaces). Bijvoorbeeld de <video> tag om video's af te spelen.

2.2 De structuur van een HTML5-document

De algemene opbouw van een html5-document zullen we uitleggen a.d.h.v. het voorbeeld in Code 1, visueel voorgesteld in Figuur 5.

```
<!DOCTYPE html>
<html lang="nl">
  <head>
    <meta charset="UTF-8" />
    <title>Onze eerste webpagina</title>
  </head>
  <body>
  </body>
</html>
```

Code 1 Voorbeeld HTML code



Figuur 5 Visualisatie HTML voorbeeld

Een HTML-pagina is niets anders dan een tekstdocument dat een andere extensie gekregen heeft en waarin de tekst in een bepaalde codering staat. Dat verklaart ook waarom HTML-bestanden vrij klein zijn.

We bewerken HTML-code bij voorkeur vanuit een tekst-editor. Dit is een programma waarmee we met tekst zonder opmaak kunnen werken, e.g., Emacs, Notepad en BBedit.

Om de webpagina beschreven in Code 1, te bekijken slaan we de code op in een tekstbestand met extensie .html. We kunnen dit bestand vervolgens openen in een browservenster.

2.2.1 Opbouw van HTML

Het root-element van onze webpagina is zoals we zien `<html>`. Verder wordt de code gesplitst in twee grote delen: een `<head>`-gedeelte en een `<body>`-gedeelte.

In het `<head>`-gedeelte komen alle gegevens die niet rechtstreeks de inhoud van onze pagina vormen. We zullen verder bijvoorbeeld zien hoe we hier trefwoorden voor zoekmachines kunnen opgeven. Wanneer we toch inhoudelijke gegevens binnen de `<head>`-tag opnemen is de browser niet verplicht deze weer te geven.

De `<title>`-tag is een verplicht element in het `<head>`-gedeelte. Met deze tag geven we aan de pagina een korte maar informatieve titel. Meestal zal deze in de titelbalk van het browservenster verschijnen. Bezoekers kunnen zo onmiddellijk zien of een pagina voor hun relevante informatie bevat.

Binnen de `<body>`-tag komt de weer te geven content van onze webpagina. Wanneer we hier tekst invoegen verschijnt deze effectief in het browser-venster. Regelovergangen, tabs en meervoudige spaties worden dus genegeerd.

2.2.2 Doctype en `<html>` element

2.2.2.1 Doctype

Elk *HTML-document* begint met een DOCTYPE en daarom zal deze als eerste worden toegelicht. De declaratie is geen HTML-element. Het is informatie naar de browser toe, welk document er mag verwacht worden.

```
<!DOCTYPE html>
```

2.2.2.2 `<html>` element

Het element `<html>` duidt op de start van de HTML-code. Het volgt op de declaratie van het doctype. Het representeert de 'root' (het element op het hoogste niveau) van een HTML-document. Alle andere elementen moeten nakomelingen zijn van dit element. Je kan ook de taal van de website opnemen binnen het `<html>` element met het *attribuut* lang

Voorbeeld

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

3 HTML-Elementen

In dit hoofdstuk zullen we de meeste gebruikte HTML-elementen overlopen. We kunnen hiermee onze basis-webpagina uit Code 1 uitbreiden.

In HTML worden speciale controletekens en codewoorden gebruikt. Deze worden elementen genoemd. Een element staat altijd tussen < en > tekens.

Syntax

```
<element attribute="value"></element>
```

Voorbeeld

```
<a href="https://www.vives.be">Visit VIVES</a>
```

3.1 Element-regels

3.1.1 Alle tagnamen met kleine letters schrijven?

HTML5 is niet case sensitive. Je mag hoofdletters, kleine letters of een mix gebruiken.

De volgende schrijfwijzen zijn correct:

```
<Table>
<TABLE>
<Table>
<a HREF="pagina.html">visit</a>
```

Voor de leesbaarheid en consistentie zullen we altijd kleine letters gebruiken.

```
<table>
<a href="pagina.html">visit</a>
```

3.1.2 Alle tags afsluiten?

In HTML5 is het niet verplicht om de elementen af te sluiten. De volgende code is correct:

```
<p>Deze cursus is een beschrijving van HTML5
<p>Dit is een lege regel

```

Wij zullen toch opteren om alle elementen af te sluiten, zoals:

```
<p>Deze cursus is een beschrijving van HTML5 </p>
<p>Dit is een lege regel </p>

```

```
<a></a>
<table></table>
```

Maar er waren ook een aantal tags die *niet* gesloten hoefden te worden. De meest voorkomende tags van dit type waren:

```
<br>
<hr>
<img>
```

Je kunt deze tags op **twee manieren** sluiten. De *eerste* manier zal je bekend voorkomen. Je voegt gewoon een sluitingstag toe.

```
<br></br>
<hr></hr>
<img></img>
```

Bij de *tweede* manier voeg je de sluitingstag toe aan de al bestaande tag. Dit doe je als volgt:

```
<br />
<hr />
<img />
```

De methode wordt ook aanbevolen door het W3C⁸.

3.1.3 Alle waarden van attributen tussen aanhalingstekens plaatsen

Alle waarden van attributen moeten niet tussen quotes staan. In HTML5 kan je dit bijvoorbeeld gebruiken:

```
<img src=plaatje.gif width="100" height=50px
  alt="Alternatieve tekst" />
```

Alleen als een attribuutwaarde uit meerdere woorden bestaat, is het noodzakelijk aanhalingstekens te gebruiken

Voor de leesbaarheid en consistentie zullen we altijd aanhalingstekens gebruiken.

```

```

3.1.4 Alle elementen moeten correct genest zijn

Een andere belangrijke regel is dat je de elementen **correct** moet **nesten**. Dit betekent dat je de tags in de juiste volgorde moet openen en sluiten.

Fout is dus:

```
<p><b><em>Tekst van de pagina.</b></em></p>
```

⁸ <http://www.w3c.org/>

Waarom is dit nu fout? Omdat de tags niet in de juiste volgorde gesloten zijn. De sluitingstag `` had voor de `` tag moeten staan.

De *correcte* code is dus als volgt:

```
<p><b><em>Tekst van de pagina.</em></b></p>
```

3.1.5 Commentaren correct schrijven

We kunnen in ons bronbestand op elke plek commentaar invoegen. Commentaar begint met de code `<!--` en eindigt met de code `-->`. Alle tekst hiertussen wordt door de parser genegeerd.

Commentaar kan zonder problemen over meerdere regels gespreid worden. We kunnen dus heel eenvoudig een fragment van ons bronbestand buiten beschouwing laten door het binnen commentaar te plaatsen.

3.1.6 De tekens & en < correct coderen

Er zijn enkele “gereserveerde” tekens in HTML5 zoals `>`, `<` en `&`. We moeten ervoor zorgen dat deze tekens niet in de gewone tekst voorkomen of in de tekst van attributen. Ze geven namelijk het begin van een entiteitsnaam aan (zoals `´`; `©`) of ze geven het begin van een element aan (zoals `<body>`). Als je daarom letterlijk de tekens `&`, `>` en `<` in het document wil tonen, moet je dit altijd aangeven met hun entiteitsnamen (entity reference) `&`; `>`; en `<`;

Voor speciale symbolen en tekens maar ook voor een hele hoop wiskunde en andere, zijn er extra entity references gedefinieerd. We zullen later zien dat dit soort definities voorkomt in de zogenaamde Document Type Definition (DTD). Een paar voorbeelden van zulke entities zijn weergegeven in Tabel 1. Een entity reference begint altijd met een ampersand (`&`) en eindigt met een puntkomma (`;`). De volledige lijst kunnen we online⁹ terugvinden.

Entity Reference	Voorgesteld karakter
<code>&copy;</code>	©
<code>&deg;</code>	°
<code>&plusmn;</code>	±
<code>&grave;</code>	È
<code>&acute;</code>	É
<code>&ecirc;</code>	Ê
<code>&euml;</code>	Ë
<code>&egrave;</code>	è

⁹ <https://dev.w3.org/html5/html-author/charref>

´	é
ê	ê
ë	ë
˜	~
€	€

Tabel 1 Een paar veelgebruikte XHTML entity references

3.2 Deprecated HTML-elementen

HTML5 wil 100% achterwaarts compatible zijn. En dus moeten ook verouderde elementen ondersteund blijven. Ze mogen daarom niet verboden worden maar tegelijkertijd wil het W3C graag dat in alle webpagina's de componenten structuur en opmaak zoveel mogelijk gescheiden zijn. Er mogen geen elementen meer gebruikt worden als ``, `<base-font>`, `<big>` enz. Welke richtlijnen werden er voorgeschreven? Er werd gekozen voor een oplossing in twee richtingen:

- Aan webontwikkelaars wordt afgeraden om visuele elementen te gebruiken. Men wordt sterk aangemoedigd om alle presentatie-elementen te realiseren met CSS (zie cursus CSS)
- Browsers/ User Agents zijn verplicht om oude elementen wel nog te ondersteunen.

Soms lukt het niet om elementen te ondersteunen in nieuwe versies, of is het simpelweg niet wenselijk. Deze worden dan als *deprecated*¹⁰ gemarkeerd in de huidige versie (zie voorbeelden van zulke elementen in Tabel 2) zodat ontwikkelaars ruim de tijd krijgen met deze verandering om te gaan. Pas nadat er een aantal versies zijn gepasseerd worden deze deprecated elementen effectief uit de standaard gehaald. Deze manier van werken wordt op alles toegepast, voorbeeld ook op de *attributen* (zie Tabel 3).

Tags / Elementen	Beschrijving
<code><acronym></code>	Definieert een acronym
<code><applet></code>	Definieert een applet.
<code><basefont></code>	Definieert een basis lettertype voor de pagina.
<code><big></code>	Definieert <i>grote</i> tekst.
<code><center></code>	Centreert tekst.
<code><dir></code>	Definieert een folder inhoud.
<code></code>	Definieert het lettertype, inclusief grootte en kleur.
<code><frame></code>	Definieert een <i>frame</i> .

¹⁰ Iets als 'deprecated' markeren betekent dat het in de nabije toekomst kan uitgeschakeld of verwijderd worden. Vaak gebruikt in programmeertalen.

<frameset>	Definieert een set van frames.
<isindex>	Definieert een inputveld (enkele lijn).
<noframes>	Definieert een <i>noframe</i> sectie.
<s>	Definieert doorstreepte tekst.
<strike>	Definieert doorstreepte tekst.
<tt>	Definieert teletype tekst.
<u>	Definieert onderstreepte tekst.

Tabel 2 Deprecated elementen

Verwijderd attribuut	Element waaruit het verwijderd werd
rev	link, a
charset	link, a
shape	a
coords	a
target	link
nohref	area
profile	head
version	html
name	img
scheme	meta
axis	td, t
abbr	td, t
scope	td
align	caption, iframe, img, input, object, legend, table, hr, div, h1, h2, h3, h4, h5, h6, p, col, colgroup, tbody, td, tfoot, th, thead, tr.
alink	body
link	body
vlink	body
text	body
background	body
bgcolor	table, tr, td, th, body.
border	table, object.

cellpadding	table
cellspacing	table
char	col, colgroup, tbody, td, tfoot, th, thead, tr.
charoff	col, colgroup, tbody, td, tfoot, th, thead, tr.
clear	br
compact	dl, menu, ol, ul.
frame	table
compact	dl, menu, ol, ul.
frame	table
frameborder	iframe
hspace	img, object.
vspace	img, object.
noshade	hr
nowrap	td and th
rules	table
size	hr
type	li, ol and ul.
valign	col, colgroup, tbody, td, tfoot, th, thead, tr
width	hr, table, td, th, col, colgroup, pre.

Tabel 3 Deprecated attributen

We weten dat een HTML-document uit twee grote delen bestaat, nl. de head en de body. We zullen eerst het <head>-element bespreken en vervolgens het <body>-element.

3.3 Elementen binnen de <head>-tag

Gezien we in het <head>-gedeelte van een pagina geen inhoudelijke gegevens plaatsen zal hier meestal niet zoveel instaan.

3.3.1 <title> element

We hebben reeds gezien dat we altijd een titel aan onze webpagina's geven met de <title>-tag, als volgt:

```
<title>Een titel</title>
```

We herhalen nogmaals dat de titel kort en informatief hoort te zijn. Gebruikers kunnen dan in één oogopslag zien of onze pagina voor hen relevant is. Zo'n titel is eigenlijk meta-data. Dit wil zeggen dat

het eigenlijk geen inhoudelijk gegeven is, maar eerder een gegeven óver de inhoudelijke gegevens. We kunnen op een meer algemene manier meta-data aangeven met behulp van <meta>-tags.

3.3.2 <meta> element

Sinds *HTML5* is het vereist om minimaal één van de volgende attributen op te nemen: name, http-equiv, charset en itemprop attributen. Dit ligt er uiteraard aan in hoeverre het <meta> element is uitgewerkt. Maar in de simpelste vorm zou het er als volgt uit kunnen zien:

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

We maken hier duidelijk welke karakterset er moet gebruikt worden met het document. Wereldwijd bestaan er verschillende character encodings. UTF-8 is een vaakgebruikte en representeert unicode tekens, wat een breed gamma aan karakters ondersteunt.

Het <meta> element kan een aantal attributen bevatten¹¹, te zien in Tabel 4.

Attribuut	Waarde	Beschrijving
charset	<i>character_set</i>	Stelt de character encoding in voor het HTML document.
content	<i>tekst</i>	Stelt de waarde in die geassocieerd wordt met het attribuut <i>http-equiv</i> of <i>name</i> .
http-equiv	content-type default-style refresh	Geeft een HTTP header voor de waarde van het content attribuut.
name	application-name author description generator keywords	Geeft een naam aan de metadata.

Tabel 4 Attributen van <meta> element

We kunnen op een meer algemene manier meta-data aangeven met behulp van <meta>-tags. Dit ziet er als volgt uit:

```
<meta name ="naam" content ="data"/>
```

Eender welke applicatie die onze pagina inleest is vrij om iets met deze informatie te doen. De specificatie gaat niet verder dan aangeven hoe we documenten van meta-data voorzien. We kunnen er dan ook niet van uitgaan dat browsers met deze tags rekening zullen houden. Trefwoorden zijn een veel voorkomend voorbeeld van meta-data. De meeste zoekmachines zullen deze gebruiken voor het correcter indexeren van onze site. In een webpagina over deze cursus zouden we bijvoorbeeld het volgende kunnen opnemen:

¹¹ https://www.w3schools.com/tags/tag_meta.asp

```
<meta name =" keywords " content =" HTML5 , CSS , cursus "/>
```



Definieer een beschrijving van je web pagina op deze manier:

```
<meta name="description" content="Free Web tutorials on HTML and CSS" />
```



Definieer de auteur van een pagina op deze manier:

```
<meta name="author" content="Hege Refsnes" />
```



Refresh het document elke 30 seconden op deze manier:

```
<meta http-equiv="refresh" content="30" />
```

Het content attribuut moet altijd gedefinieerd worden als het name- of http-equiv attribuut gebruikt wordt.

3.3.3 <link> element

Het <link> element definieert de relatie tussen het document en externe bronnen. Het zal besproken worden in de cursus CSS. Het is in HTML5 niet meer nodig om hier een type aan toe te voegen. Meestal is het duidelijk om welk soort bestanden het gaat. Voorbeeld:

```
<link rel="stylesheet" href="/stylesheets/sifr.css" />
```

3.4 Elementen binnen de <body>-tag

Nadat het volledige document is omsloten door de tags <html> en </html> en de titel een plaatsje heeft gekregen binnen het element <head>, kan je verder alle overige tekst, afbeeldingen en codes kwijt in het body-gedeelte. Er bestaan tientallen elementen om uw document structuur te geven. Een element bestaat altijd uit een openingstag (<html>) en een sluitingstag (</html>) en het beïnvloedt de tekst tussen die tags. Dit wordt in HTML5 een *containertag* of *element* genoemd.



HTML5 gaat over de structuur van de elementen. Niet over de visuele weergave. De opsplitsing van elementen in block en inline level is verdwenen¹². Zie verder in de cursus.

¹² Het belangrijkste verschil is dat een block-level element altijd de volledige breedte gebruikt en geen andere elementen naast zich duldt. Een inline element staat samen met andere elementen.

3.4.1 Heading

Binnen een tekst brengen we meestal structuur aan met titels op verschillende niveaus. In HTML5 zijn zes zulke niveaus voorzien, die we – in dalende volgorde van significantie – voorstellen met de tags <h1>, <h2>, <h3>, <h4>, <h5> en <h6>.



```
<h1>Een heel significante titel</h1>
```

Default worden titels in het algemeen vetgedrukt weergegeven. De tekstgrootte varieert meestal van heel groot, voor <h1>, naar heel klein, voor <h6>.



De volgende HTML code:

```
<h1>Heading 1</h1>
<h2>Heading 2</h2>
<h3>Heading 3</h3>
<h4>Heading 4</h4>
<h5>Heading 5</h5>
<h6>Heading 6</h6>
```

Resulteert in de volgende visualisatie:

Heading 1

Heading 2

Heading 3

Heading 4

Heading 5

Heading 6

Na elke header-tag begint de browser automatisch een nieuwe regel.

Voor de verdere indeling van ons document in blokken gebruiken we de <div>-tag. Hiermee geven we de logische samenhang van een groep elementen aan. Default hebben <div>-tags geen uiterlijke kenmerken, maar we zullen dit vaak met CSS beïnvloeden. Dit wordt verder in de cursus uitgebreid behandeld.

3.4.2 Gestructureerde tekst

Ook binnen platte tekst kunnen we logische onderdelen onderscheiden. Het meest voor de hand liggend is de onderverdeling van tekst in paragrafen. Dit doen we in HTML met de <p>-tag, als volgt:

```
<p>Tekst van een paragraaf.</p>
```

De meeste browsers plaatsen standaard een marge boven en onder paragrafen. Veel mensen misbruiken daarom kennelijk graag de <p>-tag als algemene methode om witruimte tussen elementen te verkrijgen. De specificatie verzoekt browsers dan ook om lege paragrafen te negeren.

We kunnen echter verder gaan en aan individuele zinsdelen een logische betekenis geven. De specificatie vermeldt hiervoor de volgende tags:

	<p>Geeft een benadrukking (emphasize) aan. De meeste browsers geven tekst binnen een -tag cursief weer.</p> <p>Voorbeeld</p> <pre>tekst</pre> <p>Resultaat</p> <p><i>tekst</i></p>
	<p>Geeft een sterkere benadrukking aan. De meeste browsers geven default tekst binnen een -tag vetgedrukt weer.</p> <p>Voorbeeld</p> <pre>tekst</pre> <p>Resultaat Tekst</p>
<cite>	Geeft een citaat uit of referentie naar een andere bron aan. De meeste browsers geven default tekst binnen een <cite>-tag <i>cursief</i> weer.
<dfn>	Geeft een definitie aan. Tekst niet-proportioneel maken (dan neemt elk teken dezelfde hoeveelheid ruimte in). Meestal wordt dan de tekst in het lettertype Courier gezet.
<code>	Geeft computer-code aan. Tekst niet-proportioneel maken (dan neemt elk teken dezelfde hoeveelheid ruimte in). Meestal wordt dan de tekst in het lettertype Courier gezet.
<samp>	Geeft uitvoer aan van, e.g., programma's en scripts.
<kbd>	<p>Geeft tekst die de gebruiker dient in te voeren aan. Deze tekst wordt meestal in een niet-proportioneel lettertype weergegeven.</p> <p>Voorbeeld</p> <pre>Gebruik <kbd>Ctrl</kbd> + <kdb>c</kdb> om tekst te kopiëren.</pre> <p>Resultaat</p> <p>Gebruik Ctrl + c om tekst te kopiëren.</p>
<var>	Geeft een variabele of een programma-argument aan.
<abbr>	Geeft een afkorting aan.

Het is uiteraard niet de bedoeling deze tags te misbruiken louter om een zekere opmaak te bekomen. De meeste van deze tags vloeien duidelijk voort uit de wereld van de technische documentatie. Om meer algemene logische zinsdelen te onderscheiden biedt XHTML de ``-tag. Deze is vergelijkbaar met een `<div>`, maar dan voor het verdelen van zinnen in componenten. Ook de ``-tag heeft default geen uiterlijke kenmerken en dient louter om met CSS beïnvloed te worden.

We hebben reeds vermeld dat alle witruimte in ons bronbestand wordt overgeslagen. Zo kunnen we bijvoorbeeld netjes de zinnen van een paragraaf over verschillende regels spreiden. De browser zal deze regelovergangen negeren.

Willen we toch ergens een nieuwe lijn beginnen dan voegen we een `
`-tag in. Op die plek wordt dan een regelovergang geforceerd. Dit element is leeg en kan als volgt gebruikt worden:

```
<p>
Lab Multimedia<br />
HTML 5...
</p>
```

Met meerdere `
`-tags achter elkaar krijgen we lege regels.

Af en toe is het wenselijk dat de browser zich niet te veel moeite met de formattering van onze tekst. Het typevoorbeeld is het publiceren van gedichten. Het is duidelijk dat we hierbij willen bekomen dat regelovergangen en meerdere spaties achter elkaar behouden blijven. In dit geval gebruiken we de `<pre>`-tag. Er wordt een niet-proportioneel lettertype gebruikt. Hieronder vind je een voorbeeld van tabulaire informatie in tekst formaat.

```
<pre>Test
a      10      20
b       8      16
c       6      12</pre>
```

Het resultaat ziet er exact als volgt uit:

```
Test
a      10      20
b       8      16
c       6      12
```

De meeste browsers gebruiken default een monospaced font om tekst binnen een `<pre>`-tag weer te geven. Alle whitespace blijft behouden. Op een paar bijzondere gevallen na geven we er sterk de voorkeur aan dat de browser de formattering van onze tekst verzorgt. Deze kan immers rekening houden met de situatie waarin de pagina wordt afgebeeld, en bijvoorbeeld op een nieuwe lijn verdergaan wanneer een regel te lang wordt.

3.4.3 Lijsten

Bepaalde gegevens komen het best naar voor in de vorm van een lijst. Op een webpagina kom je ook vaak opsommingen tegen. In dit onderdeel worden de verschillende types besproken. Default komen de termen tegen de linkerkant en worden de verklaringen geïndenteerd.

3.4.3.1 Genummerde lijst

Een genummerde lijst bestaat uit een opsomming met letters of getallen ervoor. Bovendien springen de items van de lijst in. Hieronder vind je een eenvoudig voorbeeld van een genummerde lijst.

```
<ol>
  <li>Eerste jaar</li>
  <li>Tweede jaar</li>
  <li>Derde jaar</li>
  <li>Vierde jaar</li>
</ol>
```

Het resultaat ziet er als volgt uit:

1. Eerste jaar
2. Tweede jaar
3. Derde jaar
4. Vierde jaar

```
<ol>lijst</ol>
```

De `ol` element-tag geeft het begin en het einde van de genummerde lijst aan.

```
<li>lijstitem</li>
```

Hiermee wordt een optie in de lijst aangeduid.

Je kan lijsten ook *nesten* in elkaar, een voorbeeld wordt hieronder gegeven.

```
<ol>
  <li>Dit is optie 1</li>
  <li>Dit is een optie met extra opties eronder:
    <ol>
      <li>Nummer 1</li>
      <li>Nummer 2</li>
    </ol>
  </li>
  <li>En hier is optie 3.</li>
</ol>
```

Het resultaat ziet er als volgt uit:

1. Dit is optie 1

2. Dit is een optie met extra opties eronder:

1. Nummer 1
2. Nummer 2

3. En hier is optie 3.

3.4.3.2 Onge Nummerde lijst

Onge Nummerde lijsten zijn hetzelfde als ge Nummerde lijsten, alleen staat hier geen cijfer of letter voor, maar een ander symbool.

```
<ul>lijst</ul>
```

Deze element-tag omvat een onge Nummerde lijst. Het gaat dus om een lijst met een bepaald symbool bij elk item.

```
<li>lijstitem</li>
```

Hiermee wordt een optie in de lijst aangeduid.

Een voorbeeld van een onge Nummerde lijst.

```
<ul>
<li>Dit is optie 1 </li>
<li>Dit is een optie met extra opties eronder:
<ul>
<li>Nummer 1</li>
<li>Nummer 2</li>
</ul>
</li>
<li>En hier is optie 3.</li>
</ul>
```

Het resultaat ziet er als volgt uit:

- Dit is optie 1
- Dit is een optie met extra opties eronder:
 - Nummer 1
 - Nummer 2
- En hier is optie 3.

3.4.3.3 Definitie lijst

Een definitie lijst is een lijst met koppen en inspringende alinea's.

```
<dl>tekst</dl>
```

Deze element-tag geeft het begin en het einde van een definitielijst aan.

```
<dt>tekst</dt>
```

Dit is de element-tag voor de kop met de tekst of term die uitgelegd gaat worden (Definition Term).

```
<dd>tekst</dd>
```

Deze element-tag omsluit de alinea onder de kop met de uitleg (Definition Description). Hieronder vind je een voorbeeld van een definitielijst:

```
<dl>
<dt>HTML</dt>
<dd>HyperText Markup Language</dd>
<dt>CSS</dt>
<dd>Cascade Stylesheets</dd>
<dt>DOM</dt>
<dd>Data Object Model</dd>
</dl>
```

Dit resulteert in het volgende:

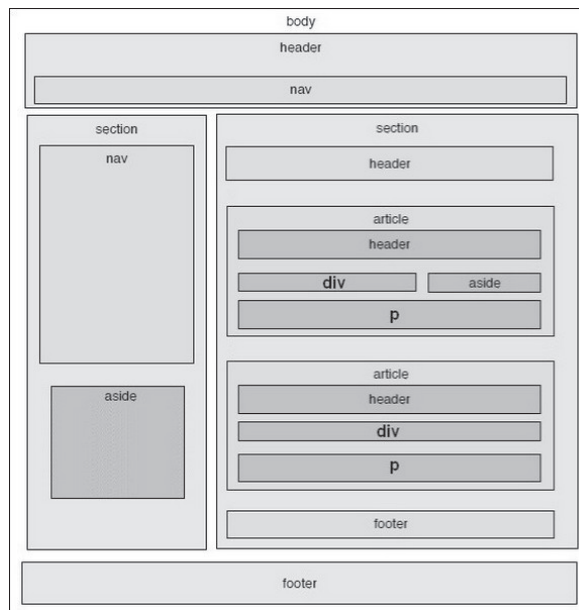
HTML	HyperText Markup Language
CSS	Cascade Stylesheets
DOM	Data Object Model

3.4.4 Semantische elementen

Wanneer je in het verleden reeds webpagina's hebt gemaakt dan ben je bekend met het opdelen van de pagina's in secties met behulp van het element `<div>`. Dit element betekent op zichzelf niets en dient enkel om een blok te omsluiten. Je kan hiermee dus structuur aanbrengen. Het positioneren en opmaken van inhoud op een website m.b.v. het element `<div>` is in de plaats gekomen van een pagina-indeling met tabellen. Complete "oorlogen" zijn er over dit onderwerp uitgevochten en je vindt de sporen ervan gemakkelijk terug op Internet. A.d.h.v. CSS kregen deze div-blokken een betekenis. Bijvoorbeeld `<div id="header">....</div>`. Alle elementen tussen het `<div>`-element kwamen bovenaan de pagina te staan.

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. Om de structurele betekenis van een codeblok op de pagina aan te duiden zijn een aantal nieuwe elementen in HTML5 geïntroduceerd.

In dit onderdeel worden de volgende nieuwe (semantische) elementen besproken (visueel voorgesteld in Figuur 6): `<section>`, `<article>`, `<aside>`, `<hgroup>`, `<header>`, `<footer>`, `<nav>`.



Figuur 6 Visuele voorstelling semantische elementen

Voorbeeld in code:

```
<body>
  <header>
    <hgroup>
      <h1>Page title</h1>
      <h2>Page subtitle</h2>
    </hgroup>
    <figure>
      
      <figcaption>logo
    </figcaption>
    </figure>
    <nav></nav>
  </header>
  <section>
    <h1></h1>
    <nav>
      <ul>Navigation</ul>
    </nav>
    <aside>
      <ul>
        <li>Top links</li>
      </ul>
    </aside>
  </section>
```

```
<section>
  <header></header>
  <article>
    <header></header>
    <div></div>
    <aside></aside>
    <p></p>
  </article>
  <article>
    <header></header>
    <div></div>
    <p></p>
  </article>
  <footer></footer>
</section>
<footer>
  <p>Copyright ©</p>
  <time datetime = "15/11/20">
    2020
  </time>
</footer>
</body>
```

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 door middel van het element 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 ze daarom allemaal als gelijkwaardig gezien. Het was zoeken naar een naald in een hooiberg.

In de volgende subonderdelen bespreken we deze nieuwe elementen.

3.4.4.1 `<header>` element

Het header element moet niet verward worden met het `<head>` element. Header geeft de content-header weer. In een typisch design zal dit het blok kunnen zijn dat bijvoorbeeld de banner, logo's en zoekformulier weergeeft.

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.

```
<div id="topcontainer">
  
  <ul>
    <li>home</li>
    <li>contact</li>
  </ul>
</div>
```

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).

```
<header>
  
  <ul>
    <li>home</li>
    <li>contact</li>
  </ul>
</header>
```

Maar het `<header>` element hoeft niet alleen gebruikt te worden voor de header van de gehele webpagina. Het 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 van toepassing op 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.

Wat kan en mag je onder andere gebruiken in de Header tag:

- figure tag
- h1, h2 etc. elementen
- nav element
- p element

3.4.4.2 <nav> element

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 motorisch gehandicapten laag. Er bestaan plugins en screenreaders die het toelaten om direct naar de navigatie te springen of juist over te slaan. 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 in *HTML5* het <nav> element geïntroduceerd. Voorbeeld XHTML:

```
<div class="menu">
  <ul>
    <li>home</li>
    <li>weblog</li>
  </ul>
</div>
```

In HTML5:

```
<nav>
  <ul>
    <li>home</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 staan. 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.

3.4.4.3 <footer> element

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.

```
<div id="footer">
  <div id="footerContainer">
    <ul>
      <li>home</li>
      ...
      <li>weblog</li>
    </ul>
  </div>
  <div id="footerAddress">...</div>
</div>
```

Daarom is met *HTML5* het <footer> element geïntroduceerd. Voorbeeld:

```
<footer>
  <nav>
    <ul>
      <li>home</li>
      ...
      <li>weblog</li>
    </ul>
  </nav>
  <div id="footerAddress">
    ...
  </div>
</ footer >
```

3.4.4.4 <section> element

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 XHTML:

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


In HTML5 zou bovenstaande code er als volgt uit kunnen zien:

```
<section>
  <article>
    <header>
      <h1>Title</h1>
    </header>
    <section></section>
  </article>
  <article>
    <header>
      <h1>Title</h1>
    </header>
    <section>...</section>
  </article>
</section>
```

3.4.4.5 <article> element

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> een stuk informatie geplaatst dat onafhankelijk is van anderen. Belangrijk aspect hierbij is de nadruk op de onafhankelijkheid van de inhoud van een <article>. Het helpt om je het volgende voor te stellen: als je de inhoud binnen het element <article>...</article> in zijn geheel uit de website kan halen, zou deze dan nog steeds betekenisvol en op zichzelfstaand gebruikt kunnen worden?

Voorbeeld:

```
<section>
  <article>
    <header><h1>Bedrijfsinformatie</h1></header>
    <p>Het bedrijf groeit!</p>
    <footer>Geplaatst door: Rory Servaas</footer>
  </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.

Verschil tussen <article> en <section>

Article wordt vaak verward met <section> en uiteindelijk ook met de vertrouwde <div>. Daarom de volgende regels:

1. Houdt het blok steek als je het leest buiten zijn context? Bijvoorbeeld in een RSS-lezer? Gebruik dan <article>.
2. Is het blok gerelateerd aan de inhoud van je pagina? Gebruik dan <section>.
3. Voor al de rest gebruik je <div>.

In het onderstaande stuk code komen er meerdere articles in een section voor. Het overkoepelende onderdeel is section, omdat het wel content gerelateerd is, maar als blok niet 1 geheel artikel vormt. Daarin zitten artikelen, die los van elkaar op zichzelf staand bruikbaar zijn.

```
<section>
  <h1>Artikelen over: Fruit</h1>
  <article>
    <h2>Appel</h2>
    <p>Een appel komt van een appelboom...</p>
  </article>
  <article>
    <h2>Sinaasappel</h2>
    <p>De sinaasappel is een oranje ronde vrucht met...</p>
  </article>
  <article>
    <h2>Banaan</h2>
    <p>Waarom zijn bananen eigenlijk krom, dat komt omdat...</p>
  </article>
</section>
```

3.4.4.6 <aside> element

In kranten zie je wel eens een kader met achtergrondinformatie of extra uitleg. Hier is een *aside* voor bedoeld.



. At the current rate, 70% of the world's reefs will be destroyed over the next 40 years.

do to help. By developing greener habits, we can all do our part in reducing global warming. For And here are some simple steps you can take to [live sustainably](#).

70% of the world's reefs will be destroyed over the next 40 years.

<aside> element

Het aside element representeert een sectie van een pagina dat bestaat uit inhoud dat zijdelings gerelateerd is aan de inhoud die rond het aside element staat. Het kan dus perfect apart bekeken worden, los van de rest van de pagina. Zulke sectie worden vaak voorgesteld als marges of sidebars in geprinte tekst.

Het element kan gebruikt worden voor typografische effecten, e.g., pull quotes of sidebars. Het is nuttig voor reclame elementen, groepen van nav elementen en voor andere inhoud die apart van de rest van de pagina moet beschouwd worden.

3.4.4.7 <hgroup> element

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 samengevoegd. 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>
```

In HTML5 wordt dit:

```
<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>
```

3.4.4.8 <time> element

Het komt er op neer dat het <time> element vooral bedoeld 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 publicatedatum van een nieuwsbericht in op te slaan. De ontwikkelaar moet zelf de datum toevoegen; bijvoorbeeld hardcoded of door middel van een *servertaal* (vb. *ASP.NET CORE*).

Voorbeeld:

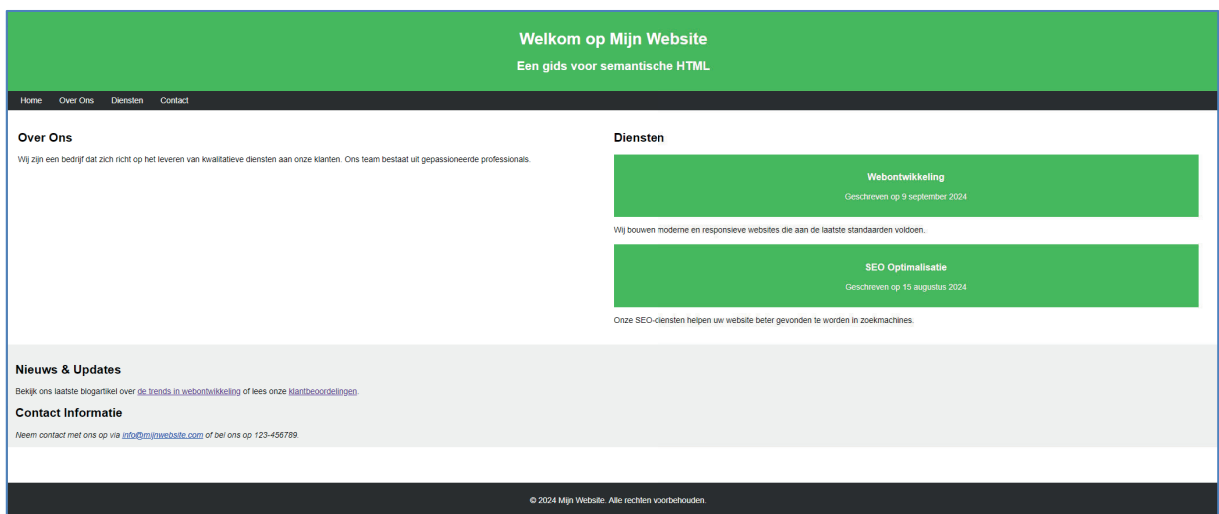
```
<article>
  <header>...</header>
  <h4>HTML5 workshop op 15 november</h4>
  <time datetime="2024-11-15" pubdate="pubdate">
    15 november 24
```

```
</time>
</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 hierboven) is het de publicatiedatum van het artikel. Als het buiten een <article> staat is het de publicatiedatum van het gehele document.

3.4.5 Voorbeeld van HTML5-pagina

In het onderstaand voorbeeld, kan je de verschillende semantische elementen terugvinden in de broncode.



```
<!DOCTYPE html>
<html lang="nl">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta name="description" content="Een voorbeeld van een goed gestructureerde
webpagina met semantische HTML-elementen">
  <meta name="author" content="CDW">
  <title>Voorbeeld Semantische HTML</title>
  <style>
    <!--style elementen, zie cursus CSS -->
  </style>
</head>
<body>
  <header>
    <hgroup>
      <h1>Welkom op Mijn Website</h1>
      <h2>Een gids voor semantische HTML</h2>
```

```

        </hgroup>
    </header>
    <nav>
        <ul>
            <li><a href="#">Home</a></li>
            <li><a href="#">Over Ons</a></li>
            <li><a href="#">Diensten</a></li>
            <li><a href="#">Contact</a></li>
        </ul>
    </nav>
    <main>
        <section>
            <h2>Over Ons</h2>
            <p>Wij zijn een bedrijf dat zich richt op het leveren van kwalitatieve diensten aan onze klanten. Ons team bestaat uit gepassioneerde professionals.</p>
        </section>
        <section>
            <h2>Diensten</h2>
            <article>
                <header>
                    <h3>Webontwikkeling</h3>
                    <p>Geschreven op <time datetime="2024-09-09">
                        9 september 2024</time>
                    </p>
                </header>
                <p>Wij bouwen moderne en responsieve websites die aan de laatste standaarden voldoen.</p>
            </article>
            <article>
                <header>
                    <h3>SEO Optimalisatie</h3>
                    <p>Geschreven op <time datetime="2024-08-15">
                        15 augustus 2024</time>
                    </p>
                </header>
                <p>Onze SEO-diensten helpen uw website beter gevonden te worden in zoekmachines.</p>
            </article>
        </section>
    </main>
    <aside>
        <h2>Nieuws & Updates</h2>
        <p>Bekijk ons laatste blogartikel over <a href="#">de trends in webontwikkeling</a> of lees onze <a href="#">klantbeoordelingen</a>.</p>
        <h2>Contact Informatie</h2>

```

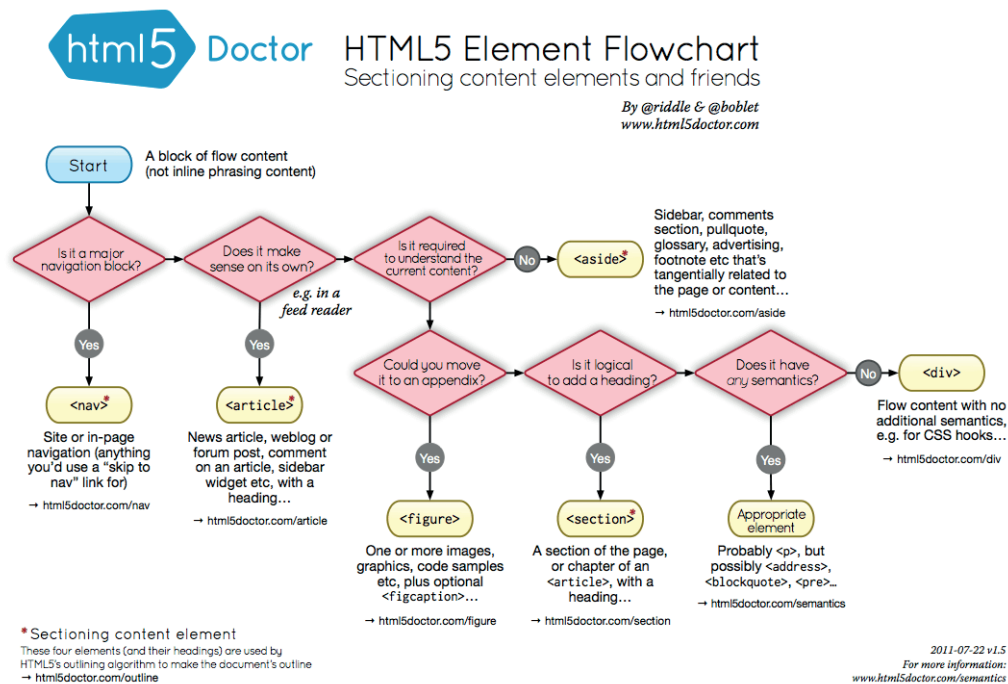
```

<address>
    Neem contact met ons op via <a
href="mailto:info@mijnwebsite.com">info@mijnwebsite.com</a> of bel ons op 123-
456789.
</address>
</aside>
<footer>
    <p>&copy; 2024 Mijn Website. Alle rechten voorbehouden.</p>
</footer>
</body>
</html>

```

3.4.6 Samenvatting

In Figuur 7 wordt weergegeven wanneer je welk HTML5 element gebruikt. Het illustreert de prioriteit die gegeven wordt door de verschillende elementen, e.g., als je een <nav> element kan gebruiken dan



moet je dat doen, ten nadele van een <div> element. Extra informatie kan je vinden op w3schools¹³.

Figuur 7 Flowchart dat aangeeft wanneer je welk HTML5 element gebruikt

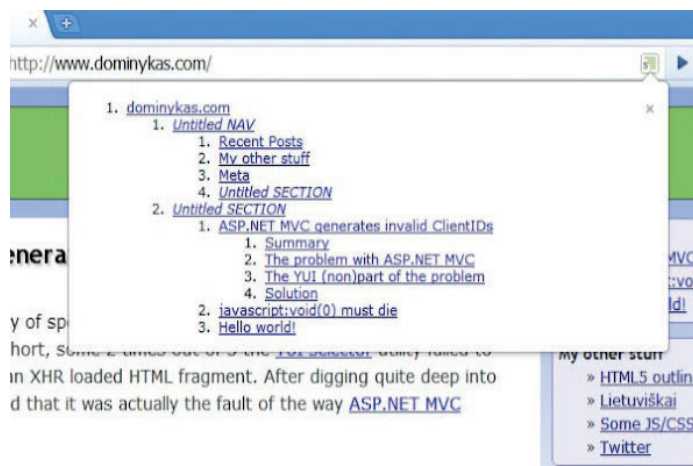
¹³ https://www.w3schools.com/html/html5_semantic_elements.asp

3.5 Een expliciete outline met HTML5

Met de komst van HTML5 verandert de codestructuur van menig website.

3.5.1 Document outline en sectioning content

Alles binnen HTML5 draait om de outline. Een outline is in feite de inhoudsopgave van je pagina. Zorg voor een goeie boomstructuur. Je kunt die eenvoudig bekijken met de Chrome extension *HTML5 outliner*¹⁴, waarvan het resultaat geïllustreerd wordt in Figuur 7.



Figuur 8 Illustratie output Outliner extension

Er bestaan ook online tools die dit kunnen doen voor je, zoals de *HTML 5 Outliner* web pagina¹⁵.

(X)HTML4 pagina's hebben ook een outline. De sections worden impliciet gemaakt door middel van de headingelementen (<h1>...<h6>) die je gebruikt. Iedere keer wanneer je een headingelement plaatst, wordt de section gesloten en wordt er een nieuwe section gestart. Als je een headingelement gebruikt dat lager in rang is dan het vorige headingelement (bijvoorbeeld een *h3* na een *h2*), start er een subsection. Op die manier krijg je een soort boomstructuur, de *outline*.



```
<h1>Forest elephants</h1>
<p>In this section, we discuss the lesser known forest
elephants.
...this section continues...</p>

<h2>Habitat</h2>
<p>Forest elephants do not live in trees but among them.
...this subsection continues...</p>

<h2>Diet</h2>
```

¹⁴ <https://github.com/h5o/>

¹⁵ <https://gsnedders.html5.org/outliner/>

```
<h1>Mongolian gerbils</h1>
```

Outline

1. Forest elephants
 1. Habitat
 2. Diet
2. Mongolian gerbils

Je zal al opgemerkt hebben dat HTML5 een halt wil toeroepen aan ongestructureerde inhoud. HTML5 gaat over structuur maar niet over visuele weergave.

In de vroegere versies van HTML was de documentstructuur (*document-outline*) eenvoudig. Een pagina had een titel (<h1>) en daaronder vertakkingen met <h2>, <h3>, ... Er was maar één boomstructuur mogelijk op een pagina die begon met een <h1> element. In HTML5 kan je gemakkelijk meerdere vertakkingen binnen de hoofdstructuur maken.

De volgende elementen <article>, <aside>, <nav>, <section>, <h1>, <h2>, <h3>, <h4>, <h5>, <h6> introduceren een nieuwe tak in de structuur.



```
<h1>My fantastic site</h1>
<section>
  <h1>About me</h1>
  <p>I am a man who lives a fascinating life. Oh the
  ...</p>
  <section>
    <h1>What I do for a living</h1>
    <p>I sell enterprise-managed ant farms.</p>
  </section>
</section>
<section>
  <h1>Contact</h1>
  <p>Shout my name and I will come to you.</p>
</section>
```

Outline van bovenstaand voorbeeld:

1. My fantastic site
 1. About me
 1. What i do for a living
2. Contact



```
<body>
  <header>
    Site title etc.
    <nav>
      <ul>
        <li><a href="/">Nav item</a></li>
      </ul>
    </nav>
  </header>
  <article id="main">
    <h1>Article title</h1>
    <p>Article content.</p>
    <h2>Article sub-heading</h2>
    <p>More content.</p>
    <h3>Article sub-sub-heading</h3>
    <p>More content.</p>
  </article>
  <aside>
    <h2>Sidebar heading</h2>
    <h3>Sidebar sub-heading</h3>
  </aside>
  <footer>
    <section>
      <h2>Footer heading</h2>
      <p>Footer content.</p>
    </section>
  </footer>
</body>
```

Outline van bovenstaand voorbeeld:

1. Untitled section
 1. Untitled section
 2. Article title
 1. Article sub-heading
 1. Article sub-sub-heading
 3. Sidebar heading
 1. Sidebar sub-heading
2. Footer heading

Het browseralgoritme gaat op de volgende manier te werk om de documentenstructuur (outline) op te bouwen:

- Het element `<body>` is de outline-root. Het is het hoogste element in de structuur.
- Vervolgens zoekt de parser naar sectioning-content- en heading-elementen. Elke keer dergelijk element wordt gevonden, wordt een tak toegevoegd aan de boomstructuur.
- Section wordt behandeld als container. Wanneer binnen een sectie een nieuwe sectie wordt gevonden, wordt er een nieuwe tak binnen de huidige aangemaakt.

Belangrijk: als een sectioning-contentelement een heading (`<h1>`, `<h2>`, ...) bevat, wordt deze gebruikt om de sectie een naam te geven. Als dit niet wordt aangetroffen, blijft de sectie naamloos (unnamed). Dus het eerste heading-element binnen een sectie zal geen nieuwe tak aanmaken, het volgende heading-element wel.



```
<section>
  <h1>Forest elephants</h1>
  <p>In this section, we discuss the lesser known forest
elephants.
  ...this section continues...
</p>
  <h3>Habitat</h3>
  <p>Forest elephants do not live in trees but among
them.
  ...this subsection continues...
</p>
</section>
```

Outline

1. Forest elephants
 - 1.1 Habitat (*impliciet gedefinieerd door het h3 element*)

Uitleg

De HTML heading elementen, i.e., `<h1>` tot en met `<h6>`, definiëren een nieuwe sectie wanneer ze *niet de eerste heading zijn van hun parent sectie*. The manier waarop deze impliciete sectie wordt gepositioneerd in de outline is gedefinieerd door zijn relatieve rank ten opzichte van de vorige heading in zijn parent sectie. Als het een lagere rank is dan de vorige heading, wordt er een sub sectie geopend.



```
<section>
  <h1>Forest elephants</h1>
  <p>In this section, we discuss the lesser known forest
elephants.
  ...this section continues...
  <h1> Mongolian gerbils</h1>
  <p>Mongolian gerbils are cute little mammals.
```

```
...this section continues...
</section>
```

Outline

1. Forest elephants
2. Mongolian gerbils (*impliciet gedefinieerd door het <h1> element, wat terzelfdertijd de vorige sectie sluit*)

Uitleg

Als het dezelfde rank heeft als de vorige heading, sluit het de vorige sectie en opent het een nieuwe op hetzelfde niveau.

Online staat er voldoende uitleg om het HTML5 content en outline model in detail te bekijken. Bekijk zeker de artikels die gaan over het content model in de gehele context¹⁶, outline voorbeelden¹⁷, het gebruik van div elementen in HTML5¹⁸ en secties en outlines in HTML5^{19, 20}.

3.6 Tabellen

Het komt vrij regelmatig voor dat we gegevens willen voorstellen in tabelvorm. Bij het opmaken van HTML-pagina's worden tabellen vaak gebruikt. Laten we ze echter niet meer gebruiken voor opmaak (om bijvoorbeeld elementen op de pagina visueel te schikken), maar enkel nog om een lijst van data weer te geven.

```
<table>kolommen, rijen en cellen</table>
```

Deze <table> tag geeft het begin en einde van een tabel aan. In Tabel 5 kan je de attributen vinden die aan <table> kunnen gegeven worden.

Attribuut	Betekenis
border=n	Een rand van n beeldpunten rond de tabel en een standaardlijn tussen de cellen.
cellspacing=n	Het aantal beeldpunten tussen twee cellen. Standaard is dit 2.
cellpadding=n	Het aantal beeldpunten tussen inhoud en rand van een cel. Standaard is dit 1.

¹⁶ <http://www.vanseodesign.com/web-design/html5-content-models/>

¹⁷ <http://html5doctor.com/outlines/>

¹⁸ <http://html5doctor.com/you-can-still-use-div/>

¹⁹ http://www.456bereastreet.com/archive/201103/html5_sectioning_elements_headings_and_document_outlines/

²⁰ https://developer.mozilla.org/en-US/docs/Sections_and_Outlines_of_an_HTML5_document

Attribuut	Betekenis
height=n height=n%	De hoogte van de tabel in beeldpunten of de relatieve hoogte ten opzichte van de beschikbare ruimte (in %)
width=n width=n%	De breedte van de tabel in beeldpunten of de relatieve hoogte ten opzichte van de beschikbare ruimte (in %)

Tabel 5 Attributen van <table>

Sommige <table> attributen zoals border, cellspacing, cellpadding, height en width werden gebruikt om het uitzicht van de tabel te veranderen maar hiervoor gebruikt men beter CSS. Idem voor het valign en baseline attribuut van <td> elementen. Toch zie je ze soms nog gebruikt worden, wellicht omdat tabellen stylen met CSS niet altijd even eenvoudig is

```
<caption>tekst</caption>
```

Met deze element-tag kun je een titel boven de tabel zetten.

```
<tr>...</tr>
```

Deze element-tag duidt het begin en einde van een (horizontale) rij aan.

```
<td>...</td>
```

Deze element-tag duidt het begin en einde van een cel aan.

Het is ook mogelijk om attributen aan de verschillende rijen en kolommen te hangen. Deze attributen pas je dan toe op, e.g., <tr>, <th> en <td>, wanneer van toepassing.

Attribuut	Beschrijving
colspan=n	Het aantal kolommen waar de cel overheen komt te staan ('omspannen')
rowspan=n	Het aantal rijen waar de cel overheen komt te staan ('omspannen')
valign=	top: de tekst aan de bovenkant van de cel middle: de tekst in het midden van de cel bottom: de tekst aan de onderkant van de cel baseline: de onderkant van de tekst gelijk met de onderkant van het grootste element

Tabel 6 Attributen voor kolommen en rijen

```
<th>...</th>
```

Deze element-tag wordt gebruikt om de eerste rij of kolom van een tabel als header aan te duiden. Het resultaat is dat de inhoud van de cel in het vet getoond wordt.

Laten we een voorbeeld bekijken. De code wordt hieronder gegeven en de visualisatie ervan kan je zien in Figuur 9.

```
<table border="1" width="95%">
  <caption>Een uitgebreide voorbeeld -tabel</caption>
  <colgroup>
    <col width="20%" />
```

```

    <col width="*" />
    <col width="20%" />
</colgroup >
<thead>
  <tr>
    <th colspan="2" >Heading<br />rij 1<br />kolom 1 en 2</th>
    <th>Heading<br />rij 1<br/>kolom 3</th>
  </tr>
  <tr>
    <td>Heading<br/>rij 2<br/>kolom 1</td>
    <td>Heading<br/>rij 2<br/>kolom 2</td>
    <td>Heading<br/>rij 2<br/>kolom 3</td>
  </tr>
</thead>
<tfoot>
  <tr>
    <td>Footer<br/>kolom 1</td>
    <td>Footer<br/>kolom 2</td>
    <td>Footer<br/>kolom 3</td>
  </tr>
</tfoot>
<tbody>
  <tr>
    <td>Body <br/>rij 1<br/>kolom 1</td>
    <td>Body <br/>rij 1<br/>kolom 2</td>
    <td rowspan="2">body<br/>rij 1 en 2<br/>kolom 3</td>
  </tr>
  <tr>
    <td>Body <br/>rij 2<br/>kolom 1</td>
    <td>Body <br/>rij 2<br/>kolom 2</td>
  </tr>
</tbody>
</table>

```

Een uitgebreide voorbeeld -tabel

Heading rij 1 kolom 1 en 2		Heading rij 1 kolom 3
Heading rij 2 kolom 1	Heading rij 2 kolom 2	Heading rij 2 kolom 3
Body rij 1 kolom 1	Body rij 1 kolom 2	body rij 1 en 2 kolom 3
Body rij 2 kolom 1	Body rij 2 kolom 2	
Footer kolom 1	Footer kolom 2	Footer kolom 3

Figuur 9 Visualisatie voorbeeld tabel

We plaatsen eerst en vooral de hele structuur binnen een `<table>`-tag. Desgewenst geven we met een optioneel `width`-attribute een gewenste breedte aan. In het voorbeeld neemt de tabel 95% van de beschikbare breedte van de omliggende container in beslag. Een gewoon getal zou hier geïnterpreteerd worden als een aantal beeldpunten.

We dienen er overigens op te letten om tables nooit aan de gehele breedte van het browser-venster toe te kennen. Internet Explorer houdt namelijk bij het berekenen van breedtes geen rekening met de zichtbare scroll-balken. Als er bij het weergeven van onze pagina een verticale scroll-balk zichtbaar is zou hier een stuk van onze tabel onder vallen. Er zou zodoende ook een horizontale scroll-balk getoond worden.

Met het `<caption>`-element geven we een korte omschrijving aan onze tabel. Dit zal default boven de eigenlijke tabel worden afgebeeld. Daarna zien we in het voorbeeld een `<colgroup>`-element met hierin een aantal `<col>`-elementen. Deze geven aan hoeveel kolommen onze tabel zal bevatten en hoe breed deze zullen zijn. De breedte wordt hier opnieuw procentueel of in beeldpunten uitgedrukt. Let wel dat de percentuele uitdrukking hier ten opzichte van de breedte van de tabel zelf is. Met een `*` geven we aan dat die kolom de rest van de beschikbare ruimte dient in te nemen.

In principe geven we voor elke kolom een `<col>`-element op. Vervolgens geven we de effectieve gegevens binnen onze tabel op. Deze worden verdeeld in drie delen: een header-gedeelte binnen een `<thead>`-tag, een footer-gedeelte binnen een `<tfoot>`-tag en een body-gedeelte binnen een `<tbody>`-tag.

Binnen deze delen geven we telkens individuele rijen aan met `<tr>`-tags. Daarbinnen geven we dan de individuele cellen aan met `<th>`- en `<td>`-tags, naargelang ze respectievelijk headers of gegevens bevatten. Met de `rowspan`- en `colspan`-attributes geven we aan dat een cel zich respectievelijk over meerdere rijen of kolommen moet spreiden. We laten de `<td>`- of `<th>`-elementen die hierdoor bedekt worden gewoon weg. We moeten hierbij wel goed opletten dat we nooit overlappende cellen definiëren!

Verder moeten we er ook op letten dat een cel nooit leeg mag zijn. Wanneer we in een zekere cel geen gegevens willen plaatsen, voegen we hier in de plaats een no-breaking space in. Dit kan met de ` `-entity. In praktijk zullen we vaak geen `<caption>`- of `<th>`-elementen gebruiken. We zullen meestal ook geen `<thead>`- of `<tfoot>`-gedeeltes opgeven. We mogen in dat geval de `<tbody>`-tag overigens ook weglaten. Wanneer we deze elementen wel opgeven, moeten we de correcte volgorde aanhouden: dus de `<caption>`-tag vlak na de `<table>`-tag en de `<thead>`-, `<tfoot>`- en `<tbody>`-tags in die volgorde.

Wanneer we dit voorbeeld in een browser bekijken zien we dat onze gegevens netjes in tabelvorm worden weergegeven. Wanneer we de grootte van het browser-venster aanpassen zal de tabel bovendien worden herberekend om opnieuw maar 95% van de beschikbare breedte in beslag te nemen.

Iets wat ons misschien – en terecht – niet zo lekker zit is dat we breedtes opgeven in onze HTML-code. Dit is immers duidelijk een layout-eigenschap en behoort dus eigenlijk in een stylesheet. We zullen later echter zien dat het in CSS wat omslachtig is om verschillende layouteigenschappen aan verschillende individuele tags toe te kennen. Vaak is het opgeven van een breedte in HTML daarom toch wat duidelijker. De `width`-attributes en het `<colgroup>`-element zijn dus niet verplicht. Wanneer de breedte van de tabel of de kolommen ook niet uit een stylesheet blijkt zal de browser waarden kiezen aan de hand van de tabelinhoud.

3.7 Hyperlinks en anchors

Eén van de bijzondere structurele mogelijkheden van XHTML is het verbinden van gegevens met hyperlinks. We kunnen zo een component van de ene pagina laten verwijzen naar gegevens op een andere pagina. Om dit te bereiken omvatten we deze component in een `<a>`-tag, als volgt:

```
<a href="url">tekst</a>
```

De component tussen de `<a>`-tags wordt meestal op een duidelijke visuele manier als hyperlink gekenmerkt. De waarde van het `href`-attribute wordt niet getoond en is het internetadres dat de browser moet openen wanneer de link wordt geactiveerd. Het volgende voorbeeld toont drie types van zulke adressen.



```
<a href="http://www.vives.be/vhti/informatica">De website
van informatica</a>
<a href="leden/lijst.html">Naar de ledenlijst</a>
<a href="mailto:christophe.dewaele@vives.be">Verstuur een
email</a>
```

1. De eerste voorbeeld-link verwijst naar een volledig adres, zoals we gewend zijn dat in de adresbalk van onze browser in te typen. We noemen dit een absoluut adres.
2. Het tweede voorbeeld verwijst naar het bestand `lijst.html` in de directory `leden`, gezien vanuit de bestandslocatie van de huidige pagina. Dit is een

relatief adres. Voor verwijzingen tussen verschillende pagina's op dezelfde website gebruiken we bij voorkeur relatieve adressen. We moeten dan bij het verplaatsen van onze site naar een andere server niets aanpassen.

3. De bedoeling van de laatste voorbeeld-link is dat de gebruiker een mailclient te zien krijgt na het klikken op de link, met hierin een nieuw, aan ons geadresseerd, bericht. Dit is een mailto adres.

Let op: met het vermelden van email-adressen op webpagina's moeten we voorzichtig zijn. Spambots doen immers niet liever dan het Internet afspeuren op zoek naar nieuwe "klanten".

We kunnen hyperlinks ook laten verwijzen naar een heel specifiek gedeelte binnen een bepaalde webpagina. We moeten dan wel van de component waarnaar we willen verwijzen een *destination anchor* maken, hetgeen we ook met de <a>-tag (waarvan de attributen in Tabel 7 Attributen van <a>Tabel 7 opgelijst zijn) doen. Meestal maken we destination anchors naar titels, bijvoorbeeld als volgt:

```
<h1><a id="hoofdstuk_HTML">Nodige en voldoende HTML 5  
elementen</a></h1>
```

Attribuut	Betekenis
id="JouwNaam" href="#JouwNaam"	Als je het attribuut id gebruikt, krijgt een positie in een webpagina een naam. De tekst is dan niet aanklikbaar, maar er kan vanuit een andere plaats op de pagina een sprongopdracht naar uitgevoerd worden via het attribuut href met als verwijzing een #, gevolgd door de naam van de locatie.

Tabel 7 Attributen van <a>

Een destination anchor heeft standaard geen speciale opmaak. Om nu specifiek naar dit gedeelte te verwijzen plaatsen we achter het internetadres van de betreffende pagina een #-teken gevolgd door de naam van het anchor.



```
<a href="#hoofdstuk_HTML5">HTML 5</a>  
<a href="xhtml-css.html #hoofdstuk_HTML5">HTML5</a>  
<a href="http://www.domainname.be/xhtmll-  
css.html#hoofdstuk_HTML5">HTML5</a>
```

3.7.1 Tekeningen

Het Internet zou niet zijn wat het is als we informatie niet met figuren zouden kunnen ondersteunen. Hier zie je hoe je de verwijzing, de opmaak en de positie van een figuur kunt instellen.

```
<img />
```

Met deze tag kun je een afbeelding invoegen. De mogelijke attributen kan je in Tabel 8 terugvinden.

Attribuut	Betekenis
<code>src="foto.gif"</code>	De locatie en de naam van het afbeeldingbestand. Denk ook aan hoofd- en kleine letters en vermijd spaties in de naam. De figuur moet van het type GIF of JPG zijn.
<code>alt="tekst"</code>	Alternatieve tekst die verschijnt als de afbeelding nog niet geladen is of niet geladen kan worden
<code>width=n</code>	De breedte van de afbeelding in pixels
<code>height=n</code>	De hoogte van de afbeelding in pixels

Tabel 8 Attributen van

In het `src`-attribuut geven we het absoluut of relatief adres van de figuur op. In het `alt`-attribuut – dat eveneens verplicht is – plaatsen we een korte tekstuele beschrijving van de figuur. Dit laatste wordt bijvoorbeeld gebruikt door voorlees-software voor slechtzienden en text-based browsers – of natuurlijk omdat de tekening om één of andere reden niet beschikbaar is.

Ook aan figuren kunnen we buiten CSS om een breedte en een hoogte meegeven en wel met respectievelijk de `width`- en `height`-attributen. We kunnen de afmetingen opgeven in zowel pixels als percentages. Onthoud hierbij wel dat percentages berekend worden ten opzichte van de dimensies van de omliggende container, en dus niet ten opzichte van de figuur zelf!



```
<p>
  <img
    src ="http
://www.vives.be/hwb/informatica/images/vives-logo.jpg"
    alt = "Een grote figuur" width ="100" height ="40" />
</p>
```

We moeten goed beseffen dat het de browser is die de tekeningen herschaalt. We kunnen dus onmogelijk bandbreedte besparen of onze pagina's sneller laten laden door op deze manier figuren te verkleinen.

Een horizontale lijn invoegen:

```
<hr />
```

4 HTML5 valideren

Vooraf wanneer we informatie gaan uitwisselen zijn duidelijke afspraken omtrent het formaat en de structuur zeer belangrijk. Voor een HTML markup-taal betekent dit dat we duidelijk moeten vastleggen welke tags, attributes en entities er bestaan en waar ze mogen voorkomen.

Wanneer we een pagina eenmaal structuur hebben gegeven, is het interessant om te controleren of er geen kleine foutjes in de tags zijn. Hiervoor zijn verschillende validatietools beschikbaar. In deze paragraaf bespreken we enkel de online validatie.

4.1 Onlinevalidatie

De bekendste online validatieservice is waarschijnlijk die van W3C zelf (W3C Markup Validation Service²¹). Deze service valideert pagina's die reeds online staan en pagina's die je zelf uploadt. Dit laatste is uiteraard handig als controlemiddel voordat je de site definitief publiceert. Het valideren van een pagina op je computer is eenvoudig:

- Open in je browser de W3C Markup Validation Service.
- Klik in het vak *Local File* en klik op *Browse* om het gewenste bestand op je computer te selecteren.
- Klik daarna op *Check* om het bestand te uploaden en te laten valideren.

Als alle HTML-regels zijn toegepast, wordt het document als HTML5 aangemerkt.

Een andere handige online-dienst om pagina's te valideren vind je op validator.nu²².

²¹ <https://validator.w3.org>

²² <http://html5.validator.nu/>

5 Formulieren

We kunnen pas van een dynamische website spreken als verschillende invoeringen door de client tot verschillende resultaten en reacties leiden. Omdat een muisklik alleen niet veel toepassingsmogelijkheden biedt, moeten er formulieren gebruikt worden. HTML5 biedt de webprogrammeur een hele waaier aan formulierelementen.

Formulieren kunnen in elk normaal HTML-document worden opgenomen. Je hoeft hiervoor geen speciale voorzorgsmaatregelen te treffen of acties te ondernemen. Formulieren kunnen prima gecombineerd worden met gewone tekst., opmaakcode, tabellen, enz. Op de plek waar je in het webdocument een formulier wilt opnemen, plaats je het element `<form>...</form>`.

Alle formulierobjecten staan binnen dit element. Binnen het formulierelement kunt u desgewenst allerlei andere opmaakcodes opnemen (alineas, lijsten, tabellen, ...). De basisopbouw van een formulier ziet er als volgt uit:

```
<form action="filename" method="post/get">
  formulierobjecten
</form>
```

Het element `<form>` bevat altijd een aantal attributen die de webserver vertellen hoe het formulier moet worden verwerkt. Die attributen beschrijven de methode waarop de gegevens verstuurd worden en de actie die de server moet ondernemen om de gegevens verder te verwerken. De attributen heten **method** en **action**.

Het method attribuut geeft aan hoe de form-data verstuurd moeten worden. De data kan verzonden worden als URL-variabelen (zoals bij Google zoekopdrachten) of via een HTTP POST request.

Je kan met de methode GET maximaal 8192 karakters meezenden. Alle gegevens uit het formulier worden als een lange sliert gegevens achter de URL gepakt (de querystring). Op het moment dat je veel gegevens wilt versturen (zoals textarea's of paswoorden) is het verstandiger om de POST methode te gebruiken.

Het attribuut action geeft letterlijk aan welke action de server met de gegevens die je hebt gepost moet uitvoeren. De URL geeft aan waar het script zich bevindt dat de gegevens verwerkt. Met dit attribuut kan je twee kanten op:

- Je stuurt de data naar een script dat de gegevens verwerkt, bijvoorbeeld `action="http://www.vives.be/verwerk"`
- Je stuurt de data naar een e-mailadres, waardoor het attribuut wordt: `action=mailto:gegevens@vives.be?subject=gegevensformulier.`

Je hoeft bij het verzenden via e-mail niet over een script te beschikken en krijgt zelf de resultaten van het formulier per e-mail geleverd. Deze vorm is gemakkelijk te gebruiken maar wordt uit veiligheidsoverwegingen niet meer door alle browsers ondersteund.

Wij zullen voorlopig action niet benoemen, we laten het dus leeg: `action=""`.

5.1 Formulierobjecten

5.1.1 Tekstvak

Een gebruiker kan een tekstvak gebruiken om tekst in te voeren. Deze tekst kan vervolgens door de webapplicatie verwerkt worden.



Figuur 10 Een tekstvak, standaard styling

```
<input type="text"
  name="invoerveld" size="aantal karakters zichtbaar"
  maxlength="max lengte" value="startwaarde" />
```



```
<input type="text" name="postcode" size="4"
  maxlength="4"/>
```

5.1.2 Paswoordvelden

```
<input type="password"
  name="data-id" size="aantal karakters zichtbaar"
  maxlength="max lengte" value="startwaarde" />
```



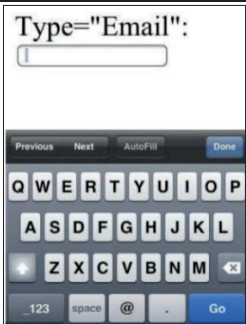

```
<input type="password" name="paswoord" size="10"
  maxlength="10" />
```

5.1.3 Email Address & Web Address

Deze twee input types zijn ontwikkeld met mobiele browsers in het achterhoofd, en voornamelijk voor smartphones waar tekstinvoer 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:

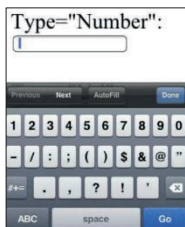
- emailadres: elk e-mailadres bestaat uit minimaal een punt en @ teken en spaties zijn niet toegelaten. Het toetsenbord zal daarom de twee tekens weergeven en maakt de spatiebalk kleiner dan in de standaard weergave.
- webadres: de spatiebalk is weggelaten, de punt en slash zijn weergegeven en tevens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

In desktopbrowsers 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.

<code><input type="email" name="txtEmail"/></code>	<code><input type="url" name="txtUrl"/></code>
	

Niet alle attributen werden opgenomen, bekijk **de volledige** lijst op w3schools²³.

5.1.4 Number



Voor het ingeven van nummers zijn er een tweetal input types in HTML5 toegevoegd:

```
<input type="number" name="txtNumber" />
<input type="range" name="txtRange" />
```

Voorheen werd hier een standaard tekstveld voor gebruikt. Wanneer een mobiele browser dit input type tegenkomt wordt het toetsenbord aangepast: cijfers in de plaats van letters. Desktop browsers geven het op twee manieren weer: *Opera* 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.



Figuur 11 Illustratie spinbox

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



Figuur 12 Illustratie schuifbalk

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.

²³ https://www.w3schools.com/html/html_form_input_types.asp

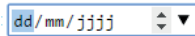
- Value: de beginwaarde die wordt weergegeven.

5.1.5 Data en tijden invoeren

Er zijn totaal een zestal input types waarmee data gekozen kan worden:

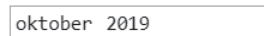
- `<input type="date" name="" />`

De dropdown button geeft een kalender weer waarin een maand, jaar en dag gekozen kan worden.



- `<input type="month" name="txtMaand" />`

Dezelfde kalender wordt weergegeven als bij date, maar in het invoerveld wordt alleen het jaar en maand weergegeven.




- `<input type="week" name="txtWeek" />`

Dezelfde kalender wordt weergegeven als bij date, maar in het invoerveld wordt alleen het jaar en week weergegeven

- `<input type="time" name="txtTime" />`

Met een spinbox of door het zelf in te voeren kan de tijd worden aangegeven.



5.1.6 Searchbox

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.

```
<input type="search" name="txtSearch" />
```



5.1.7 Tekstvak met meerdere regels

Een tekstvak waarin je meerdere regels in kan typen noemen we een *textarea*.



Figuur 13 Illustratie textarea

```
<textarea rows="aantal rijen"
  cols="aantal karakters breed" name="date-id" wrap="virtual">
  eventueel begintekst
</textarea>
```



```
<textarea rows="5" cols="50" name="Opmerkingen"
  wrap="virtual">
  Bij opmerkingen gelieve dit invoerveld te gebruiken
</textarea>
```

5.1.8 Checkbox

Een checkbox gebruik je om een binaire keuze te vragen of geven aan de gebruiker.



Figuur 14 Een checkbox met standaard styling

```
<input type="checkbox"
  name="data-id"
  value="waarde die wordt overgedragen"
  checked = "checked" />
```



```
<p>Kent u deze sporten ?</p>
<input type="checkbox" name="voetbal" value="v"
  checked="checked" />
Voetbal<br/>
<input type="checkbox" name="tennis" value="t" />
Tennis
```

5.1.9 Radiobutton

Een radiobutton gebruik je om input van de gebruiker te vragen, en wanneer er maar één antwoord mag geselecteerd worden.



Figuur 15 Illustratie radiobutton

```
<input type="radio"
  name="data-id"
  value="waarde die wordt overgedragen"
  checked = "checked" />
```



```
Hoe zal u betalen ? <br>
<input type="radio" name="betaalmiddel" value="c" />
Cash
<input type="radio" name="betaalmiddel" value="v" />
Visa
<input type="radio" name="betaalmiddel" value="b"
  checked="checked" />
Bancontact
```

Opmerking: het name-attribuut heeft voor alle drie de radiobuttons dezelfde waarde.

5.1.10 Keuzelijst

Een keuzelijst laat toe om (potentieel) meerdere opties te selecteren.

```
<select name="data-id" size="aantal menuopties zichtbaar"
multiple="multiple">
  <option value="waarde1" selected ="selected">keuzel</option>
  <option value="waarde2">keuze2</option>
</select>
```



```
<p>Welke programmeertalen kent u ?</p>
<select name="taal" size="3">
  <option value="java">JAVA </option>
  <option value="delphi">Delphi</option>
  <option value="c">C#.NET</option>
  <option value="asp">ASP.NET CORE</option>
  <option value="vb">VB.NET</option>
</select>
```

5.1.11 Verborgen velden

Onder een verborgen veld verstaan we een formulierelement dat niet op het beeldscherm is te zien, maar wel wordt overgedragen.



```
<input type="hidden"
name="data-id"
value="waarde" />
```

5.2 Nieuwe attributen voor <input> element

5.2.1 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.

```
<input type="text" name="txtNaam" autofocus="autofocus" />
```

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 is belangrijk om te weten dat aan het invoerveld id="element" wordt meegegeven):


```
<script>
  window.addEventListener('load', (event) => {
    document.getElementById('element').focus();
  });
</script>
```

5.2.2 Required

Voor het registreren op een website zijn er altijd een aantal waarden, die verplicht zijn; o.a. naam, 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. Voorlopig ondersteunen enkel Opera en Chrome dit: als het veld niet is ingevuld en de gebruiker drukt op de *submit* knop, zal de focus op het niet ingevulde invoerveld worden gezet.

```
<input type="text" name="txtEmail" required="required" />
```

Waarden in andere invoervelden blijven ingevuld. Voorheen moest de ontwikkelaar zelf een stuk code in *JavaScript* of *ASP.NET* schrijven om te kijken of de verplichte velden ingevuld waren. In *HTML5* wordt het als volgt aangegeven:



Figuur 16 Verplicht input veld in HTML5

5.2.3 Placeholder

Met het *placeholder* attribuut kan standaard inhoud meegegeven worden aan een invoerveld. Zodra er in het invoerveld geklikt wordt zal de tekst verdwijnen.

```
<input type="text" name="txtNaam"
  placeholder="gelieve naam in te geven" />
<input type="submit" />
```



Figuur 17 Placeholder waarde in <input >

5.2.4 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. Werkt voorlopig enkel in Opera.

Figuur 18 List voorbeeld

Code voor bovenstaand voorbeeld is als volgt:

```
<input list="browsers" name="browser" />
<datalist id="browsers">
  <option value="Internet Explorer">
  <option value="Firefox">
  <option value="Chrome">
  <option value="Opera">
  <option value="Safari">
</datalist>
<input type="submit" />
```

5.2.5 Multiple

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 kunnen geselecteerd worden. Om de functionaliteit van dit attribuut te gebruiken moet het in het input type element worden geplaatst:

```
<input type="text" multiple="multiple" />
```

5.2.6 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 Belgische postcode bestaat altijd uit vier cijfers. Aan het invoerveld zou dan meegegeven kunnen worden dat er totaal vier cijfers verwacht worden of exact twee letters. Validatie van de waarden wordt hierdoor een stuk makkelijker.

Figuur 19 Illustratie pattern op <input>



```
<input type="text" pattern="[0-9]{4}"
  title="Vier cijfers" />
```

Hoe je dergelijke patronen moet opstellen, is geen materie binnen deze cursus. Het komt wel aan bod binnen het vak webprogrammeren (andere voorbeelden: <http://html5pattern.com/>).

5.3 Formulieren verzenden

In de voorgaande paragrafen heb je de belangrijkste HTML-opdrachten voor het maken van een formulier gezien. Eén aspect is tot nu toe nog niet behandeld: hoe verzend je een formulier?

Hiervoor zijn speciale formulierobjecten beschikbaar: *buttons*. Er zijn drie hoofdtypen:

- Een button waarmee de informatie die met het formulier verzameld is, verstuurd wordt; dit heet Submit. De code is: `<input type="submit" />`
- Een andere hoofdtype is een button waarmee alle invoervelden in het formulier gewist worden. Dit type heet Reset. De code is: `<input type="reset" />`
- Een derde type is een neutrale button. Je zal acties voor deze knop zelf moeten programmeren met JavaScript (zie cursus deel: Javascript).

Voor alle typen is het attribuut `value` beschikbaar, waarmee je tekst op een button kunt plaatsen.

```
<form action="filename" method="post/get">  
  formulierobjecten  
  <input type="submit" name="knop" value="Verzenden" />  
</form>
```

Wanneer een formulier verstuurd wordt, moet men aangegeven volgens welke METHOD dit zal gebeuren. Hierbij hebben we de keuze tussen POST of GET. Met POST worden de gegevens naar de server gestuurd als onderdeel van de HTTP-headers, met GET als onderdeel van de URL. Welke actie er moet gebeuren op de verzonden data, wordt vermeld na ACTION. Dit attribuut laten we voorlopig leeg. Het zullen we later gebruiken binnen het domein webprogrammeren (ASP.NET CORE). Met de HTML-code `<input type="submit">` krijgt men een verzendknop. Let op de volgende attributen. `id` is de identifier van het knopvlak. `value` geeft een waarde aan het knopvlak, belangrijk voor de latere identificatie. Als men een aantal van deze knoppen in een formulier gebruikt, kan men met `value` vaststellen welke knop is aangeklikt.

5.4 Voorbeeld van een formulier

Inlichtingen

Name:

Email:

Website:

Number:

Range:

Message:

Figuur 20 Voorbeeldformulier

Opmerking: de formulierelementen staan niet mooi onder elkaar. Hoe we dit correct afhandelen komt aan bod in de cursus CSS.

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Voorbeeld formulier</title>
  </head>
  <body>
    <h2>Inlichtingen</h2>
    <form action="" method="post">
      <label for="name"> Name:</label>
      <input type="text" name="name" required placeholder="Name" />
      <br/>
      <label for="email">Email:</label>
      <input type="email" name="email" required
        placeholder="info@vives.be" />
      <br/>
      <label for="website">Website:</label>
      <input type="url" name="website" required
        placeholder="http://www.example.com" />
      <br/>
      <label for="number">Number:</label>
      <input type="number" name="number" min="0" max="10"
        step="2" required placeholder="Even num < 10"/>
      <br/>
      <label for="range">Range:</label>
      <input type="range" name="range" min="0" max="10" step="2" />
      <br/>
      <label for="message">Message:</label>
```

```
<textarea name="message" required></textarea>
<br/>
<input type="submit" value="Send Message" />
</form>
</body>
</html>
```

6 HTML5-API's

Naast nieuwe elementen en tags in de html5-specificatie zijn er door de W3C ook diverse ondersteunende *application programming interfaces (API)* ontwikkeld. Dat betekent dat er ook programmeerinterfaces beschikbaar zijn om de elementen in de pagina te besturen.

Een aantal API's maken deel uit van de HTML-standaard (geïntegreerde API's) zelf, andere zijn in de loop van het proces losgekoppeld van HTML (gerelateerde API's).

Geïntegreerde API's:

- Video en audio
- Inline Editing
- Offline application
- History application
- Web protocol
- Drag & Drop

Gerelateerde API's:

- Geolocation
- 2D canvas
- Web storage
- Web Worker
- Web Sockets
- Messaging
- Contact
- Indexed Database
- Microdata

We beperken ons in dit hoofdstuk tot een aantal API's, o.a. Video en audio. Canvas, Drag & Drop en Web Storage.

6.1 Video

Eén van de doelen van de WHATWG met *HTML5* was om externe plugins overbodig te maken. Dit wil men onder andere bereiken met het <video> element, dat plugins als *Flash*, *Silverlight* en *QuickTime* overbodig maakt.

In de meeste gevallen (tot 2015) werd een video afgespeeld via de Flash Player plug-in. Sinds 2008 is ook Microsoft Silverlight een plug-in geworden. De videoplug-in die wordt gebruikt, is in de meeste

gevallen vooraf geïnstalleerd op nieuwe computers en werkt naadloos samen met de webpagina waarin de plug-in is geplaatst. Maar in sommige gevallen is een plug-in niet aanwezig of kan het niet geïnstalleerd worden, denk maar aan de iPhone/iPad van Apple. Het element `<video>` is ontwikkeld om dit probleem op te vangen. Dit element wordt ondertussen ondersteund door de meeste browsers.

Het `<video>`-element in HTML5 biedt de mogelijkheid om video's direct in een webpagina te integreren zonder het gebruik van externe plug-ins zoals Flash. Een belangrijk aspect van het werken met video's in HTML5 is het begrijpen van de verschillende videoformaten die ondersteund worden door verschillende browsers.

6.1.1 Basissyntaxis van het `<video>`-element

Hier is de basisstructuur van het `<video>`-element:

```
<video src="video.mp4" controls>
  Your browser does not support the video element.
</video>
```

- **src:** Geeft het pad naar de videobron aan.
- **controls:** Toont de standaardbedieningselementen voor de video zoals afspelen, pauzeren en volume regelen.
- **Fallback:** De tekst tussen het `<video>`-element is een fallback-bericht voor browsers die het `<video>`-element niet ondersteunen.

6.1.2 Belangrijke attributen van het `<video>`-element

Het `<video>`-element biedt verschillende attributen die de functionaliteit van video's op een webpagina verbeteren:

- **controls:** Toont de standaard bedieningsknoppen van de browser.

```
<video src="video.mp4" controls></video>
```

- **autoplay:** De video begint automatisch te spelen zodra deze is geladen. Let op: browsers kunnen autoplay blokkeren tenzij de video is gedempt.

```
<video src="video.mp4" autoplay ></video>
```

- **muted:** De video begint standaard zonder geluid (vaak gebruikt met autoplay om te zorgen dat de video zonder problemen start).

```
<video src="video.mp4" autoplay muted></video>
```

- **loop:** De video wordt automatisch opnieuw afgespeeld zodra deze is afgelopen.

```
<video src="video.mp4" controls loop></video>
```

- **poster:** Een afbeelding die wordt weergegeven voordat de video wordt afgespeeld. Dit is handig als je een representatieve thumbnail wilt tonen.

```
<video src="video.mp4" controls poster="thumbnail.jpg"></video>
```

6.1.3 Waarom verschillende videoformaten?

Er zijn verschillende videoformaten omdat niet elke browser en niet elk platform dezelfde videocodecs ondersteunt. Een codec is een technologie die wordt gebruikt om video's te comprimeren (kleiner te maken) en te decomprimeren (af te spelen). Verschillende browsers en apparaten ondersteunen verschillende codecs, en dat is waarom je meerdere formaten moet aanbieden om ervoor te zorgen dat je video werkt op alle platforms.

De belangrijkste redenen voor verschillende videoformaten zijn:

- **Compatibiliteit:** Browsers hebben hun eigen voorkeuren voor codecs. Sommige browsers ondersteunen bepaalde videoformaten vanwege licentieovereenkomsten of technische redenen.
- **Bestandsgrootte:** Verschillende formaten bieden verschillende niveaus van compressie. Sommige formaten bieden een goede balans tussen kwaliteit en bestandsgrootte, wat belangrijk is voor prestaties en laadsnelheid.
- **Kwaliteit:** Afhankelijk van de toepassing wil je video's in een formaat met een hogere of lagere compressie. Dit heeft invloed op de kwaliteit van de video en de laadsnelheid.

6.1.4 Veelgebruikte videoformaten in HTML5

Hier zijn de drie meest gebruikte videoformaten die je tegenkomt bij het insluiten van video's in HTML5:

1. MP4 (H.264 + AAC)

- **Type:** video/mp4
- **Codec:** H.264 (video) en AAC (audio)
- **Voordelen:**
 - Breed ondersteund door de meeste browsers, zoals Chrome, Safari, Internet Explorer, Edge en mobiele browsers.
 - Goede balans tussen bestandsgrootte en kwaliteit.
- **Nadelen:**
 - H.264 is een gepatenteerde codec, dus er zijn licentiekosten voor het gebruik in bepaalde softwaretoepassingen.
- **Gebruik:** MP4 is het meest aanbevolen formaat voor maximale compatibiliteit.

2. WebM (VP8/VP9 + Vorbis/Opus)

- **Type:** video/webm
- **Codec:** VP8 of VP9 (video) en Vorbis of Opus (audio)
- **Voordelen:**
 - Open-source en vrij van patenten, dus er zijn geen licentiekosten.
 - Ondersteund door moderne browsers zoals Chrome, Firefox en Opera.
 - Zeer efficiënt in compressie, vooral VP9, wat resulteert in kleinere bestanden met goede kwaliteit.
- **Nadelen:**
 - Niet ondersteund door alle browsers, met name oudere versies van Safari en Internet Explorer.
- **Gebruik:** Ideaal voor moderne browsers, vooral voor situaties waarin bestandsgrootte belangrijk is (bijvoorbeeld bij langzame verbindingen).

3. Ogg/Theora (Theora + Vorbis)

- **Type:** video/ogg
- **Codec:** Theora (video) en Vorbis (audio)
- **Voordelen:**
 - Open-source, zonder licentiebeperkingen.
 - Ondersteund door Firefox en Opera.
- **Nadelen:**
 - Niet ondersteund door veel andere browsers, zoals Safari, Internet Explorer en Edge.
 - Grotere bestandsgroottes en lagere compressiekwaliteit vergeleken met H.264 en VP8/VP9.
- **Gebruik:** Minder gebruikelijk vanwege de lagere compressiekwaliteit, maar kan een fallback-optie zijn voor sommige open-sourceprojecten.

6.1.5 Gebruik van meerdere videobronnen

Om ervoor te zorgen dat je video in zoveel mogelijk browsers werkt, kun je meerdere videobronnen aanbieden in verschillende formaten. De browser kiest dan automatisch het formaat dat het beste ondersteund wordt. Dit doe je met het <source>-element:

```
<video controls poster="thumbnail.jpg">
  <source src="video.mp4" type="video/mp4">
  <source src="video.webm" type="video/webm">
  <source src="video.ogv" type="video/ogg">
  Your browser does not support the video tag.
```

</video>

The browser support for the different formats is:

Brows	MP4	WebM	Ogg
IE	YES	NO	NO
Chrom	YES	YES	YES
Firefox	YES	YES	YES
Safari	YES	NO	NO
Opera	YES (from Opera 25)	YES	YES

Er zijn namelijk drie hoofdvarianten (OGG, MP4 en WebM) en geen van deze drie worden door alle browsers ondersteund. Webontwikkelaars 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 Firefox, *Chrome* en *Opera* werkt, kan het <video> element als volgt worden gebruikt:

```
<!DOCTYPE html>
<html>
<body>

<video width="320" height="240" autoplay>

  <source src="movie.ogg" type="video/ogg">

  Your browser does not support the video tag.
</video>

</body>
</html>
```



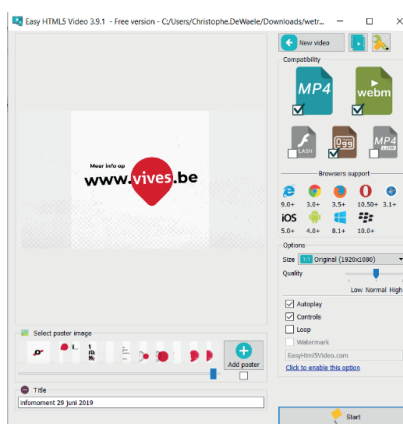
```
<video src="video /movie.ogg" width="320" height="240" controls
></video>
```

Hieronder wordt een woordje uitleg gegeven over het concept van een container en een codec. Video en audio worden zelden of nooit ongecomprimeerd verdeeld. De reden is heel eenvoudig: opslagruimte en transfertijd. Om je een idee te geven: een bestand met ongecomprimeerde HD-video in een resolutie van 720p heeft een fikse 330 GB per uur aan opslagruimte nodig. Comprimeren is de oplossing. Het gebruikte algoritme en de implementatie om een mediabestand te verkleinen, noemen we een codec (**co**deren / **de**coderen of **com**primeren / **de**comprimeren). Speelt je mediaspeler een film of een nummer niet afgespeeld, dan heb je dus waarschijnlijk de juiste codec niet geïnstalleerd. De uiteindelijke verpakking van het gecomprimeerde bestand, af te lezen aan de extensie, is de container.

Containers kunnen een heleboel data bevatten. Wav bevat bijvoorbeeld enkel audiodata. Een mkv-bestand daarentegen kan verschillende audio- en videostromen bevatten, maar ook ondertitels, hoofdstukindelingen en allerlei metadata. Een avi-bestand kan net zo goed een MPEG4-video als een ongecomprimeerde video bevatten, of zelfs alleen een audiobestand. Codecproblemen kom je voornamelijk tegen bij video en minder bij audio.

Om te weten welke (video-)codec er gebruikt werd, zal je speciale software moeten gebruiken, zoals [GSpot](#). Sleep je daar een videobestand naartoe, dan krijg je alle informatie die je nodig hebt.

Je zal meerdere formaten video moeten aanbieden en hiernaar verwijzen in jouw code. Je moet dus hetzelfde filmpje op meerdere verschillende manieren coderen en dubbel of driedubbel op jouw webserver plaatsen. In de praktijk zal je de video moeten exporteren voor verschillende codecs. Je kunt eventueel Adobe Premiere Elements gebruiken maar er bestaan ook goedkopere manieren vb. “Easy HTML5 Video”.



Software: easy HTML5 Video

Je opent het gewenste video-fragment en selecteert vervolgens naar welke codecs je wilt exporteren.

Extra info: <http://www.adobe.com/devnet/html5/articles/html5-multimedia-pt1.html>

6.2 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 is:

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

Attributen die voor het <audio>-element van toepassing zijn:

- **Controls** Hiermee worden een standaardset van controls aangeboden: play/pause, volume en progressiebalk.
- **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 het volledig is afgespeeld.

Extra info: <http://www.adobe.com/devnet/html5/articles/html5-multimedia-pt2.html>

