hogeschool vives

Web Development 1 : Labo 3 HTML - 2

studiegebied **HWB**

bachelor in het **toegepaste Informatica**

campus **Kortrijk**

academiejaar **2023-2024**

Inhoudsopgave

Inhoudsopgave **2**

1 Inleiding **3**

1.1 Verslag 3

2 Labo **4**

2.1 De DOM-tree 4

2.2 Opdracht 1 4

2.3 Opdracht 2 5

2.4 Opdracht 3 5

2.5 Opdracht : Personal homepage assignment 6

2.6 HTML tables 6

2.7 Opdracht: Contact information assignment 7

2.8 Semantische elementen 7

2.9 Opdracht Opleidingsaanbod 8

2.10 Outline 8

2.11 Opdracht Uitbreiding personal homepage 8

1 Inleiding

Deze les behandelt enkele praktische zaken rond werken met afbeeldingen, HTML validering en online hosting.

1.1 Verslag

In dit deel van de cursus staan verschillende vragen die je moet beantwoorden en opdrachten om iets te maken of uit te proberen. Het is belangrijk dat je alle opdrachten zorgvuldig uitvoert!

Documenteer je werk in een verslag document "**verslag HTML deel 2.pdf**" waarin je:

- Voor elke uitprobeeropdracht een entry maakt met screenshots ter staving van wat je deed.
- Je antwoorden op de gestelde vragen neerschrijft.

Oplossingen van 'grotere' opdrachten (met veel code) bewaar je in een Webstorm project "**HTML deel 02**". Per opdracht maak je in dit project een aparte folder waarin je de bestanden (en subfolders) plaatst.

2 Labo

2.1 De DOM-tree

De structuur van een HTML document wordt door de browser intern voorgesteld in een boomstructuur die men de DOM-tree van dit document noemt (DOM = Document Object Model).

De DOM-tree is belangrijk voor ons omdat we deze programmatorisch (met een javascript programma) kunnen aanpassen om zo de pagina inhoud te wijzigen. We zullen ook zien dat de opmaakregels die we met CSS zullen definieren, gebaseerd zijn op de structuur van de DOM-tree.

Je kan een voorstelling van de DOM-tree bekijken in de Chrome Developer Tools, op het Elements tabblad. Als je met de muis over een element in de DOM-tree gaat, zal Chrome de visualisatie van dit element highlighten in de webpagina.

Je kan trouwens ook rechtsklikken op een onderdeel van een webpagina en "Inspect element" kiezen om rechtstreeks naar het corresponderende element in de DOM-tree te gaan, zelfs als de Chrome Developer Tools nog niet actief waren.

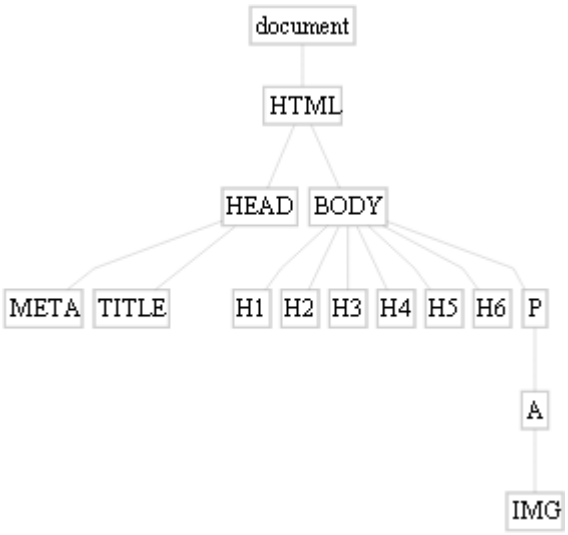
We komen bij de bespreking van CSS nog uitgebreid terug op het DOM-tree concept.

2.2 Opdracht 1

Open de volgende pagina en bekijk vervolgens de bijbehorende HTML-code :

http://HTMLdog.com/examples/headings1.HTML

De structuur van de DOM-tree voor deze pagina ziet er als volgt uit :



Figuur 1: Voorbeeld DOM Tree

Vergelijk het schema en de HTML-code. De structuur van de DOM-tree is gebaseerd op de nesting structuur van het HTML-document. Waarmee komen de blokjes overeen? En de lijntjes ertussen?

2.3 Opdracht 2

Normaliter wordt de DOM-tree aangepast door een javascript programma, wat pas later in de cursus aan bod komt. Om het effect van dergelijke aanpassingen te demonstreren gaan we de DOM-tree eens handmatig bewerken met de Chrome Developer tools.

Open in Chrome de 'voorbeeld headings.HTML' pagina die we in het vorige labo maakten.

Ga met de muis over de elementen in de DOM-tree en kijk hoe onderdelen in de pagina in een highlight geplaatst worden.

Rechtsklik op een van de elementen in de DOM-tree, kies "Delete" en merk op dat dit onderdeel uit de pagina verdwijnt.

Dubbelklik op de content (i.e. de tekst tussen begin- en eindtag) van een hoofdingselement en pas de tekst aan. In de webpagina verschijnt de nieuwe tekst.

Wijzigingen aan de DOM-tree leiden dus tot wijzingen in de webpagina. Dit gebeurt allemaal lokaal in de
browser, er is geen server interactie mee gemoeid!

We onderscheiden vanuit dit oogpunt twee soorten HTML-documenten

- **Statische HTML-documenten**
  Hun inhoud verandert niet eenmaal ze in de browser worden getoond
- **Dynamische HTML-documenten**
  Door de DOM-tree te bewerken met javascript wordt de pagina in de browser aangepast.

Inzicht: in een statisch HTML-document zal de page source, i.e. de HTML-code die de browser in de HTTP-response ontving, altijd netjes overeenkomen met de inhoud van de DOM-tree. Bij een dynamisch HTML-document is dit doorgaans NIET zo.

Dit dynamische karakter is een van de kenmerken van wat men "Web 2.0" noemt.

2.4 Opdracht 3

JQuery UI is een CSS+javascript library die je kan gebruiken om populaire UI-componenten in je webapplicatie te krijgen. Je zou dit natuurlijk ook allemaal zelf kunnen bouwen, maar een (goede) library van iemand anders gebruiken is goedkoper en sneller.

Bekijk eens de demo van hun accordion widget op:

https://jqueryui.com/accordion/

En bekijk op het Elements tabblad van de Chrome Developer Tools hoe dit accordion widget wordt voorgesteld in de DOM-tree.

Uit welke elementen bestaat elke accordion section in de DOM-tree?

Als je op een gesloten section in de accordion klikt, klapt deze open terwijl een andere zich sluit.

Welke wijzingen gebeuren er in de DOM-tree als je op een gesloten section klikt?

Zo'n accordion ziet er heel complex uit, maar we keken nu eigenlijk onder de motorkap van deze library. Zeer leerrijk, maar om hun accordion in je eigen webapplicatie te gebruiken moet je de precieze details gelukkig niet kennen!

Klik eens op de 'View source' link onderaan de demo om te zien hoe simpel de elementen in je HTMLdocument zijn als je een accordion wil gebruiken.

Ook hier zie je een groot verschil tussen wat er in de broncode van je HTML-document moet staan en wat de uiteindelijke structuur in de DOM-tree is. Je ziet bv. dat er heel wat attributen aan de oorspronkelijke elementen zijn toegevoegd. Hoe zou dit gebeurd zijn?

2.5 Opdracht : Personal homepage assignment

Maak de personal homepage assignment (zie aparte pdf).

Let erop dat je niet de verkeerde HTML-elementen gebruikt om toch maar een mooiere visualisatie of layout te bekomen! De correcte manier om dit te doen is m.b.v. CSS maar dat komt pas in een latere les aan bod. Het is dus volkomen normaal dat je persoonlijke pagina er voorlopig nog niet erg gelikt uitziet.

Bewaar deze in een aparte folderstructuur in je Webstorm project, je zult er later nog aan verder moeten werken. Als er later aan verder werkt, maak dan een kopie om aan te passen!

Dit is een perfecte opdracht om eens te experimenteren met de verschillende mogelijkheden in HTML, hou het dus vooral niet minimalistisch.

2.6 HTML tables

Lees sectie 3.6 uit de HTML-cursus.

Een zeer belangrijk advies bij dit alles is :

Gebruik nooit tables om een pagina layout te doen kloppen!

Vroeger werden tabellen nml. wel eens misbruikt om een pagina in meerdere kolommen in te delen, dit behoort men echter met CSS te doen.

Tabellen gebruik je als getabelleerde informatie wil weergeven, bv. uurrooster, weersvoorspelling voor de komende dagen, puntenlijsten, etc.

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.

Houd bovenstaande in het achterhoofd als je de sectie over tabellen in de cursus leest (zie p.37).

Merk op dat de attributen colspan en rowspan natuurlijk wel nog steeds nuttig zijn, deze veranderen de tabel immers inhoudelijk.

Indien een cel geen inhoud heeft, zet dan tenminste een   tussen begin- en eindtag van het <td> element, m.a.w. gebruik <td> </td> in plaats van een lege <td></td>.

En voor de goede orde nog eens: **gebruik nooit tables om een pagina layout te doen kloppen!**

2.7 Opdracht: Contact information assignment

Maak de contact information assignment (zie aparte pdf).

Bewaar je oplossing in een aparte folderstructuur in je Webstorm project.

2.8 Semantische elementen

Lees sectie 3.4.4 t.e.m. 3.4.6 over semantische elementen.

Vroeger gebruikte men vooral elementen om gerelateerde inhoud te bundelen of af te bakenen (vergelijkbaar met in een paragraaf).

Stukjes inhoud van een pagina bundelen is interessant om naar 'de bundel als geheel' te kunnen verwijzen vanuit CSS en javascript. Dit impliceert natuurlijk ook een bepaald verband tussen de inhoud van het gebundelde, anders zouden we die niet als 1 geheel willen behandelen in CSS of javascript. Het nadeel is dat een element geen verdere betekenis aan de gebundelde inhoud geeft, het dient enkel ter groepering.

In HTML5 werden veel nieuwe elementen toegevoegd waarmee betekenis kan worden gegeven aan een onderdeel van een pagina. Men noemt dit semantische elementen (ter info : het woordje "semantiek" betekent "betekenis").

Bijvoorbeeld :

- <section> als groeperingsmiddel van gerelateerde inhoud (heeft typisch ook een header)
- <article> als speciale section, nml met inhoud die op zichzelf kan staan buiten de context van het huidige document
- <header> en <footer> voor hoofding en voettekst
- <nav> voor een blok met navigatie links
- <aside> dient voor secundaire informatie en heeft 2 toepassingen (afhankelijk of het in een <article> staat of niet)
- <time> met eventueel een pubdate attribuut

Enkele van bovenstaande voorbeelden kunnen gebruikt worden om inhoud te bundelen en tegelijk aan te geven wat die bundel eigenlijk voorstelt.

Het element is trouwens ook nog steeds bruikbaar als groeperingsmiddel, maar dan louter om technische redenen en zonder speciale betekenis.

Merk op dat articles en sections beiden in elkaar genest mogen worden :

- een <section> die meerdere <article>s bevat (bv. frontpagina van een nieuws site)
- een <article> dat meerdere <section>s bevat (bv. een lang artikel met subsecties)

Een <aside> dient voor secundaire informatie, dwz informatie die niet de primaire focus is van een article of pagina maar er wel mee te maken heeft. We onderscheiden twee soorten secundaire informatie:

- <aside> in een <article> is secundaire informatie die bij het article hoort
  bv. definitie van termen of toelichting bij een bepaald concept
- <aside> buiten alle <article>s is secundaire informatie die bij de pagina op zich hoort
  bv. <nav> met bijkomstige links naar gelijkaardige pagina's

Een <header> heeft niks te maken met <head>. Het is ook ruimer dan de klassieke headings <h1..6>, veelal zal een <header> een heading bevatten maar er kan ook meer inhoud aan gegeven worden zoals bv. een korte samenvatting van wat volgt, of een teaser of ...

Sommige elementen worden als *obsolete* gemarkeerd: deze zullen in volgende versies van HTML 5 inactief of verwijderd worden. De lijst van zulke elementen kan je hier vinden: https://www.w3.org/HTML/wg/drafts/HTML/master/obsolete.HTML.

Het bovenste code fragment in de sectie over het <header> element, is niet correct omdat een <ul> element geen <img> rechtstreeks mag bevatten zoals we reeds eerder zagen (dit moet in een <li> genest worden).

Let op het gebruik van <figure> en <figcaption> in het voorbeeld uit sectie 3.4.5 !

2.9 Opdracht Opleidingsaanbod

Maak de opleidingsaanbod opdracht (in aparte pdf) en hou rekening met semantische elementen.

2.10 Outline

Overloop Sectie 3.5 over outlines (algoritme en voorbeelden), dit stuk is echter van minder belang.

2.11 Opdracht Uitbreiding personal homepage

Maak een kopie van je oplossing voor de 'Personal homepage assignment' en voeg een 'Prognose' tabel toe waarin je jouw score voorspelt voor drie vakken uit de opleiding alsook de punten van drie medestudenten voor deze vakken.

Herwerk je oplossing voor de personal homepage assignment door dit keer rekening te houden met semantische elementen.