

Bonus hoofdstukken

- 1 HTML, CSS en JavaScript**
- 2 Een website bouwen**

1 HTML, CSS en JavaScript

De belangrijkste punten van bonushoofdstuk 1

- ▶ HTML is een taal waarmee u de inhoud en structuur van een webpagina kunt maken.
- ▶ Tags zijn codewoorden die de verschillende onderdelen, de elementen, van een HTML-document markeren.
- ▶ Een HTML-document bestaat uit een head- en een bodysectie. In de sectie head vindt u bijvoorbeeld de titel en in de body komen zaken voor als koppen, paragrafen en tabellen.
- ▶ Een dialoog met behulp van HTML-formulieren is een veelgebruikte manier van interactie met een gebruiker.
- ▶ CSS staat voor *Cascading Style Sheets*. Deze stijlbladen bepalen het uiterlijk van een HTML-document.
- ▶ JavaScript gebruikt u om uw pagina gebruikersvriendelijker te maken.
- ▶ jQuery is een JavaScript-bibliotheek waarmee u gemakkelijk pagina's dynamisch kunt stylen en Ajax-functies kunt uitvoeren.

Wat leert u in dit hoofdstuk?

Met HTML kunnen we de structuur van een webpagina bepalen.

```
<!DOCTYPE html >
<html>
  <head>
    <title>Een simpel HTML-
document</title>
  </head>
  <body>
    <p>Hello world!</p>
  </body>
</html>
```

Met CSS maken we er iets moois van.

```
body {
  background: #f7f7f9;
  font-family: trebuchet ms;
  color: navy;
}
```

1.1 HTML 101

Een 101, spreek uit: ‘one - oo - one’, is een Amerikaanse uitdrukking voor een supersnelle cursus waarin u alleen die dingen leert die nodig zijn voor het begrijpen van het vervolg. Nu denkt u misschien: ‘HTML? We gingen toch programmeren in PHP?’ Dat klopt, maar het resultaat van het uitvoeren van een PHP-script is vaak een HTML-tekst. Enige kennis van HTML is dan ook nodig om te begrijpen hoe PHP werkt.

Terwijl HTML voor de structuur en inhoud van uw pagina zorgt, stelt CSS u in staat om de inhoud en de vormgeving van uw website uit elkaar te houden. Met JavaScript maakt u uw webpagina meer dynamisch en interactief. Mocht u na het lezen van dit hoofdstuk meer willen weten over HTML en CSS, dan verwijs ik u naar *HTML en CSS de basis*. Voor meer informatie over JavaScript is er *JavaScript de basis*.

We zullen hier kort aandacht besteden aan:

- Hoe ziet een HTML-document eruit?
- De tags *html*, *head*, *title*, *body*.
- Het maken van lijsten: de tags *ul*, *ol*, *li*, *dl*, *dt*, *dd*.
- Het maken van tabellen met de tags *table*, *thead*, *th*, *tbody*, *tr*, *td*.
- Het gebruik van CSS-stijlbladen en koppeling met een HTML-document met de tags *script*, *div*, *span*.
- Het invoegen van JavaScript.

1.2 De anatomie van een HTML-document

We herhalen het nog maar eens: met HTML bepaalt u de inhoud en de structuur van een webpagina. U bepaalt de inhoud van een pagina en vervolgens verdeelt u het document in paragrafen, u brengt koppen aan of een tabelstructuur, u kunt plaatjes invoegen enzovoorts, alles met behulp van HTML-tags.

De meeste browsers zijn heel vergevingsgezind tegenover slordig HTML. Als u een browser alleen het volgende aanbiedt als HTML-document:

 Dit is de inhoud van een HTML-pagina, er staat geen enkele HTML tag in.

Dan zal die browser dat keurig weergeven met een standaard vormgeving. Er zullen heel weinig pagina's zijn waarin dit voldoende is. Als u een pagina mooier wilt maken met een speciale structuur, opmaak, lettertypen enzovoort, dan zult u HTML-tags moeten gebruiken, CSS-stijlbladen en eventueel JavaScript.

1.2.1 Tags

De inhoud van een document bestaat uit een aantal onderdelen, zoals *titel*, *koptekst*, *paragraaf* et cetera. In een HTML-document worden deze onderdelen *gemarkeerd* door

paren *tags*, een begintag en een eindtag. Tags zijn codewoorden tussen hoekige haakjes: `< en >`. Dezelfde tag met een `/` achter de `<` geeft het einde van het bereik van een tag aan, dus alle tekst tussen `<body>` en `</body>` vormen de ‘body’ van een HTML-document. Een paragraaf en een titel zien er bijvoorbeeld zo uit:

```
<p>Dit is de paragraaftekst</p>
<title>Dit is een titel</title>
```

ELEMENTEN

Een onderdeel dat door tags gemarkeerd is zullen we een *HTML-element* noemen. Een *HTML-document* is dus eigenlijk niets anders dan een verzameling van HTML-elementen.

In oudere versies van HTML zijn tags niet hoofdlettergevoelig; het maakt daar niet uit of u `<BODY>`, `<body>`, `<Body>` of `<body>` schrijft. Wij zijn echter van plan zo veel mogelijk alleen nette HTML te gebruiken, en dan maakt het wel degelijk uit. Vergeet dus alle hoofdletters!

1.2.2 Attributen

Tags kunnen *attributen* hebben, die het HTML-element van extra informatie voorzien. Attributen zien er altijd zo uit: naam="waarde" of: naam='waarde'.

Voorbeeld: `<table border="1">`.

In oudere versies van HTML kon u vaak de ‘ of de “ weglaten, dus `<table border=1>`. De meeste browsers zullen dit nog wel goed weergeven, maar als u dynamisch met uw HTML-code aan de slag wil, kan dit een bron zijn van onbegrijpelijke fouten. Bij voorkeur gebruikt u “ om attribuutwaarden. Attributen staan altijd in de begin-tag van een element.

1.2.3 HTML-voorbeeld

Het allersimpelste welgevormde HTML-document dat u kunt maken ziet er ongeveer zo uit:

```
<!DOCTYPE html >
<html>
  <head>
    <title>Een Simpel HTML-document</title>
  </head>
  <body>
```

```
<p>Hello world!</p>
</body>
</html>
```

U ziet dat het document begint met te vertellen wat het voor soort document is, de `<!DOCTYPE`. Voor eerdere versies van HTML dan HTML5 en voor XHTML is er eigenlijk een langere DOCTYPE declaratie nodig, bijvoorbeeld:

```
C <!DOCTYPE HTML
PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
```

Maar als u dit als eerste regel in uw HTML-document zet, dan hebt u de kans dat de browser over uw HTML-document valt omdat er een foutje in staat en een foutboodschap laat zien in plaats van uw pagina. Dat is natuurlijk erg gebruikersonvriendelijk, zeker omdat 90% van alle webpagina's niet welgevormd schijnt te zijn. Daarom zal elke browser uw pagina zo goed mogelijk laten zien, ook met fouten, als u alleen zegt dat het HTML is, of helemaal geen DOCTYPE specificeert. Alleen hebben niet alle browsers hetzelfde idee over hoe met niet-welgevormde pagina's om te gaan.

In het voorbeeld hierboven ziet u een HTML-document dat uit twee gedeelten bestaat, een hoofd en een lichaam, toepasselijk gemarkeerd met een `<head>`- respectievelijk een `<body>`-tag. Dit geldt voor ieder welgevormd HTML-document.

Laten we beginnen te bekijken wat er in de *head* van een HTML-document kan staan.

1.2.4 HTML-head tags: title, meta, link

De belangrijkste tags die u binnen een *head*-sectie kunt gebruiken zijn:

- `<title>` Definieert de titel van het document.
- `<link>` Een referentie naar een extern stijlblad of JavaScript-bestand.
- `<meta>` Definieert meta-informatie, zoals beschrijving, auteur, taal, et cetera.
- `<style>` Een stijlblad opgenomen in het document.
- `<script>` JavaScript in het document, of een verwijzing naar een bestand met JavaScript dat geladen moet worden.

Voorbeeld:

```
C <head>
<title>Weblampjes Voorbeeld</title>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
<meta name="Keywords" content="PHP,MySQL, cursus" />
<meta name="Description" content="PHP en MySQL cursus boek" />
<link rel="stylesheet" type="text/css" href="/weblampjes.css" />
<script type="text/javascript" src="lib/jquery-1.6.2.min.js"></script>
</head>
```

We gaan hier niet in detail in op de head-tags. Van belang is de `title`-tag, die u in ieder document zult willen gebruiken om aan te geven waar de pagina over gaat. Als u uw webpagina vindbaar wilt maken voor zoekmachines, dan is het een goed idee om de meta-informatie zo goed mogelijk in te vullen.

Kijk ook naar de `link`-tag, waarmee u kunt aangeven dat er een extern stijlblad gebruikt moet worden, zoals in het voorbeeld. Voor een intern stijlblad gebruikt u de tag `style`.

JavaScript dient u te omhullen met `script`-begin- en eindtags. Het voorbeeld geeft aan dat de jQuery-bibliotheek uit een bestand geladen moet worden. Vaak worden scripttags aan het eind van de body van een HTML-document geplaatst in plaats van in de head, omdat dit het laden van de pagina sneller schijnt te maken en omdat u dan zeker weet dat het document geladen is voordat de JavaScript-code wordt uitgevoerd.

1.2.5 HTML-body tags: headings, paragraphs, break

Alle inhoud van een HTML-pagina is in de body van het HTML-document te vinden. We zien koppen, paragrafen, tabellen, lijsten, afbeeldingen, links naar andere pagina's, video-inhoud, animaties enzovoorts.

Als eerste voorbeeld van bodyinhoud laten we een overzicht zien van verschillende kop-regels, paragrafen en het begin van een nieuwe regel:

```
<body>
  <h1>dit is een h1 kop</h1>
  <h2>dit is een h2 kop</h2>
  <h3>dit is een h3 kop</h3>
  <p>...</p>
  <h6>dit is een h6 kop</h6>
  <p>Hier start een nieuwe paragraaf</p><p>en hier houdt hij op.
    En hier<br />start een nieuwe regel.
  </p>
</body>
```

(We laten hier alleen de inhoud tussen de begin- en eindtag van `<body>` zien, niet het complete HTML-document). Dan is dit wat de browser laat zien:



Afbeelding 1.1
HTML-overzicht.

Dit voorbeeld heeft verder weinig uitleg nodig. Merk op dat er geen tekst is die niet tussen tags staat, en dat alle begintags ook een eindtag hebben. De tag *br*, een *line break*, is een speciaal geval, en een voorbeeld van een gecombineerde begin- en eindtag.

1.2.6 HTML-bodytags: lijsten

Tabellen worden veel gebruikt om elementen van een document netjes op een pagina te plaatsen. Wij zullen dat ook regelmatig doen om bijvoorbeeld de velden van een formulier recht onder elkaar te krijgen. Eigenlijk zou u een HTML-tabel alleen moeten gebruiken als u ook tabelgegevens wilt laten zien, anders is het beter via een stijlblad de gegevens te formatteren. De andere structuurelementen die u tussen `<body>` en `</body>` kunt plaatsen zijn `` voor een puntenlijst en `` voor een genummerde lijst. Met een definitielijst, de tags *dl*, *dt*, *dd*, kunt u bijvoorbeeld een *glossary* (verklarende woordenlijst) maken. We laten eerst een voorbeeld zien (we laten de head-sectie van het HTML-document ook hier weg) waarbij de verschillende soorten lijsten gebruikt worden:

```
C <body>
    <ul>
        <li> item1 </li>
        <li> item2 </li>
    </ul>
    <ol>
        <li> item1 </li>
```

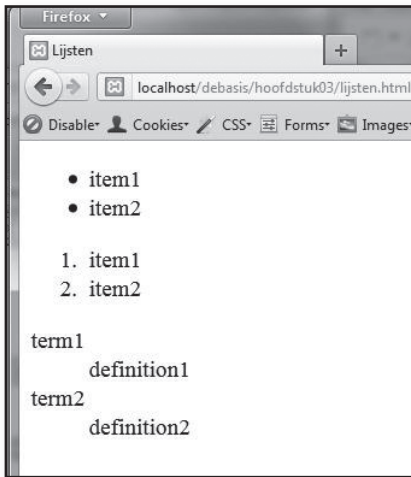


```

        <li> item2 </li>
    </ol>
    <dl>
        <dt> term1 </dt> <dd> definition1 </dd>
        <dt> term2 </dt> <dd> definition2 </dd>
    </dl>
</body>

```

De browser maakt er dit van:



Afbeelding 1.2
Lijsten.

1.2.7 HTML-body tags: tabellen

De definitie van een tabel ziet er zo uit (we hebben nu ook de body-tag weggelaten):

```

<table border="1">
  <thead>
    <tr>
      <th>kop1</th>
      <th>kop2</th>
      <th>kop3</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td> a1 </td>
      <td> a2 </td>
      <td> a3 </td>
    </tr>

```

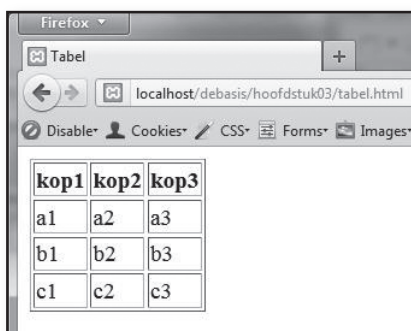
```
<!-- meer rijen -->
</tbody>
</table>
```

We maken een paar opmerkingen over tabellen:

- Het zal u opvallen dat ook tabellen over een hoofd en een lijf beschikken.
- In de *thead*-sectie kunt u kolomkoppen kwijt, en in de *tbody*-sectie de inhoud van de tabel. In de praktijk zult u `<thead>` en `<tbody>` weinig tegenkomen, want alle browsers laten een tabel zonder deze tags ook goed zien. Maar als u dynamisch met een tabel aan de slag wilt met behulp van JavaScript, dan is het belangrijk om deze tags wel te gebruiken, om er zeker van te zijn dat alle browsers, met name oudere versies van Internet Explorer, meedoen.
- Verder zijn er nog een aantal variaties toe te voegen, zoals tabellen binnen tabellen, rijen die hier en daar over meer dan één kolom reiken, of kolomwaarden die over meer rijen heen gaan. Let op: lege tabelcellen kunnen problemen geven. We wijzen hier alleen op een aantal mogelijkheden, voor details is het al eerder genoemde HTML-boek een aanrader, of het onvolprezen W3Schools: www.w3schools.com/html/ zal zeker uitkomst brengen.

Dit voorbeeld zondigt tegen de regels dat er geen presentatieattributen in de HTML-tekst mogen staan. Borders (randen), padding en spacing horen in een stijlblad te worden aangegeven. De `border="1"` is het enige attribuut dat nog niet verouderd is verklaard voor de `table`-tag. Het attribuut `border` is bijvoorbeeld handig om snel te debuggen waarom een geneste tabel uit de pas is gaan lopen door het vergeten van een eind-tag, alhoewel dat tegenwoordig ook heel handig gaat met de developer tools van Firefox.

Onze Firefox-browser toont het voorbeeld als in afbeelding 1.3.



kop1	kop2	kop3
a1	a2	a3
b1	b2	b3
c1	c2	c3

Afbeelding 1.3
Een tabel.

Erg fraai is deze tabel niet vormgegeven. Verderop in dit hoofdstuk als we het over CSS hebben komen we terug op dit voorbeeld.

1.2.8 HTML-body tags: div en span

Rondom een willekeurig paar HTML-tags in de body-sectie van een document kunt u `<div>...</div>` of `...` plaatsen. Deze tags zijn bedoeld om blokken tekst bij elkaar te houden en gezamenlijk vorm te geven. De *div*-tag kan ook voor positionering gebruikt worden.

Een verschil tussen *div* en *span* is, dat *div* een nieuwe regel veroorzaakt bij het vormgeven en *span* niet. U kunt *span* bijvoorbeeld gebruiken om een stukje tekst in een ander lettertype te plaatsen of een speciale kleur te geven.

Een *div* wordt vooral gebruikt om secties in een HTML-document te groeperen om deze als een geheel te kunnen positioneren of van een stijl voorzien. Vooral in gegenereerde HTML (door PHP bijvoorbeeld) ziet u soms een overvloed aan *div*'s. Dit maakt uw pagina niet sneller of overzichtelijker, dus probeer zuinig met *div*'s om te gaan. We zullen genoeg voorbeelden van een *div* tegenkomen, dus we slaan deze nu even over.

1.2.9 Standaardattributen

Er is een aantal attributen die u bij bijna elke HTML-tag kunt gebruiken, maar die voornamelijk interessant zijn om *div* en *span* van betekenis te voorzien.

- *id* geeft de tag een unieke identificatie. Voorbeeld: `<div id="menu-sectie">`. Een extern stijlblad of stijldefinities in de head-sectie van het document kunnen de *id* gebruiken voor vormgeving en positionering. Een *id* moet uniek zijn binnen een HTML-document. Als dat niet zo blijkt te zijn, zullen sommige browsers een foutmelding geven, andere zullen stilletjes het laatst gevonden element gebruiken. We zullen in de hierna volgende paragraaf over CSS vertellen hoe u stijlelementen voor een HTML-element met een *id* kunt aangeven.
- *class* geeft de tag een classificatie. Voorbeeld: ``. Ook voor *class* geldt dat u daar elders een stijl voor zult definiëren en dat we hierna zullen laten zien hoe dat moet.
- *style* kan gebruikt worden om in een element een stijl te definiëren. Voorbeeld: `<div style="color: blue; font-style: italic">`. Deze stijl zal cursieve blauwe tekst als uitvoer tot gevolg hebben. Gebruik dit attribuut liefst niet, maar in plaats uw stijl-attributen in een stijlblad en geef uw *div* of ander element een *id* of een *class*.

1.3 HTML-formulieren

De meest gebruikte en simpelste manier van interactie met een gebruiker is een dialoog met behulp van HTML-formulieren. Met HTML-formulieren kunt u tekstvelden, tekstblokken, keuzevakjes, keuzerondjes, meerkeuzelijsten, verstuurknoppen en meer maken. We laten hier alleen een erg simpel formulier zien.

De basisvorm van een HTML-formulier blijkt uit het voorbeeld:

```

c <form action="foo.php" method="get">
    voor- en achternaam:
    <input type="text" name="username" />
    <br />
    email:
    <input type="text" name="email" />
    <br />
    <input type="submit" name="submit" value="Stuur op!" />
</form>

```

1.3.1 Formulier – kenmerken, input type: text en submit

We noemen een aantal punten:

- Een formulier wordt omgeven door `<form>` en `</form>`.
- In het form staan input - tags. Het attribuut *type* geeft aan wat voor soort inputveld er getoond wordt.
- *Type text* geeft een tekstveld aan.
- *Type submit* geeft een button aan. Erop klikken heeft het indienen van een HTTP-verzoek aan de server tot gevolg.
- Andere mogelijke typen zijn: checkbox, password, radio button en meer. Het is bijvoorbeeld mogelijk om bestanden te uploaden en om een plaatje als submit-knop te gebruiken.
- Er kan ook beschrijvende tekst in een formulier geplaatst worden, zoals voor- en achternaam: en **e-mail**: die u in het voorbeeld zag.
- De actie die uitgevoerd wordt bij een submit staat aangegeven in het attribuut *action* van de form -tag. In het voorbeeld zal **foo.php** worden uitgevoerd.
- *method="get"* geeft aan *hoe* het PHP-script (**foo.php**) de waarden van de invoervelden ter beschikking krijgt. De andere manier om invoergegevens ter beschikking te stellen is via *"post"*. Hoe u met PHP de invoergegevens kunt bekijken en bewerken behandelen we in hoofdstuk 5.

Het zou natuurlijk leuk zijn als we in de browser een mooi formulier te zien zouden krijgen, zoiets als afbeelding 1.4.



Afbeelding 1.4
Een simpel formulier.

Helaas is dat zonder aanpassing niet het geval. Probeert u zelf maar *bonus hoofdstuk01/simpelform1.html* in de browser te openen en dan zal blijken dat alle velden scheef onder elkaar staan. In de volgende sectie, die over CSS gaat, zullen we dit oplossen.

1.3.2 Formulier velden, input type: radio button en checkbox

Hieronder laten we een voorbeeldje zien met radiobuttons (keuzerondjes) en checkboxen (selectievakjes):

```
<form action="foo.php" method="get">
  Soort vakantie? <br />
  <input type="checkbox" name="vakantie" value="stedentrip" />
  Stedentrip
  <br />
  <input type="checkbox" name="vakantie" value="nederland" />
  Nederland
  <br/><br/>
  Links- of rechtshandig? <br />
  <input type="radio" name="handig" value="links" /> Links<br />
  <input type="radio" name="handig" value="rechts" /> Rechts
  <br/><br/>
  <input type="submit" name="submit" value="Stuur op!" />
</form>
```

We hebben de HTML buiten de form-tag weggelaten, het complete voorbeeld kunt u vinden in *bonushoofdstuk01/radiocheck.html*. In de browser zou u dit kunnen zien:



Afbeelding 1.5
Checkboxes en radio buttons.

Het kenmerkende verschil tussen checkboxes en radio buttons is dat u uit een groep checkboxes er meerdere mag kiezen, en uit een groep radio buttons maar één.

1.3.3 Formulier velden: select en textarea

Regelmatig ziet u op webpagina's een drop-down lijstje waaruit u iets moet kiezen, zoals het land waar u woont. Dit kunt u zelf maken met behulp van *select*- en *option*-tags.

```
<select>
  <option>Andorra</option>
```

```
<option>Belgie</option>
<option>Nederland</option>
<option>Luxemburg</option>
</select>
```

In een voorbeeld dat we later zullen behandelen komt een textarea voor. Voor onze toepassing ziet de HTML er zo uit:

```
<textarea name="vraagtekst" rows="5" cols="50">
    Voorbeeld van vraagtekst
</textarea>
```

- Een *textarea* heeft een aantal rijen en kolommen, in ons voorbeeld 5 en 50, die de afmetingen van het tekstblok vaststellen. Dit in tegenstelling tot de *input type text*, waarbij er alleen een lengte nodig is.
- Als er bij het tonen van het tekstblok al tekst in gezet moet worden, dan staat deze tekst tussen `<textarea>` en `</textarea>`. In ons voorbeeld is dat "Voorbeeld van vraagtekst".

Als we deze twee code fragmenten combineren tot één voorbeeld in *bonus hoofdstuk01/selecttextarea.html*, dan zien we:



The screenshot shows a web form. At the top is a dropdown menu with a small arrow icon. The menu is open, showing a list of countries: Andorra, Belgie, Nederland, and Luxemburg. A mouse cursor is pointing at 'Nederland'. Below the dropdown is a text area with the placeholder text 'Voorbeeld van vraagtekst'. At the bottom of the form is a button labeled 'Stuur op!'.

Afbeelding 1.6
Drop-down lijst en textarea

LET OP!

Als u de gebruiker tekst wil laten invoeren in een *textarea* dan moet u er goed over nadenken hoe u de invoer gaat filteren zodat u geen ongewenste effecten krijgt. Dit geldt natuurlijk ook voor andere tekstinvoer in een *input*-element. Zorg dat uw gebruikers geen Trojaanse paarden binnensmokkelen! We gaan hier in hoofdstuk 5 nader op in.

Er valt nog heel wat meer te vertellen over HTML, en er zijn een heleboel tags die we niet konden behandelen in deze paar pagina's. In de volgende oefening laten we u wat uitzoekwerk doen over een tag die we nog niet gehad hebben, de *img*-tag.

OEFENING 1.1

Maak een webpagina waarin u iets over uzelf vertelt. Zorg dat er een foto van uzelf, of een andere afbeelding in komt te staan. Gebruik in ieder geval een paar tekstkoppen, paragraaftekst, en een puntenlijst of een definitielijst.

1.4 Invoegen van een CSS-stijlblad

Nu we de structuur van een webpagina kunnen maken, wordt het tijd om ook over de opmaak na te denken. We hebben al gezien dat het hulpmiddel om dit te doen, CSS is.

CSS betekent *Cascading Style Sheet*. Cascading (aaneengeschakeld in een serie), geeft aan dat u meerdere stijlbladen kunt maken die elkaar aanvullen of overschrijven, afhankelijk van de volgorde waarin de stijlen gedefinieerd worden. Stijlen bepalen het uiterlijk van een HTML-document. Als u geen stijl opgeeft, dan zal de browser een standaardvormgeving toepassen, zie bijvoorbeeld afbeelding 1.1.

CSS is een declaratieve taal, dat betekent dat u aangeeft hoe een bepaald element vormgegeven moet worden, maar dat de volgorde waarin u uw CSS-code opschrijft er minder toe doet. Behalve als u twee keer hetzelfde zegt over een bepaald element, dan zal de laatste of de meest specifieke regel gelden. Ik geef toe dat dit een beetje vaag is en soms kunnen ingewikkelde stijlbladen onbegrijpelijke vormgeving opleveren. De enige raad die we kunnen geven is om stijlbladen zo eenvoudig mogelijk te houden.

Een stijlelement ziet er zo uit:

```
c selector {eigenschap: waarde; eigenschap: waarde}
```

De *selector* geeft de tag-naam en eventueel een *class*-naam aan, de eigenschappen of *properties* zijn de stijlelementen zoals kleur, lettertype enzovoort. We zullen hier in gaan op drie soorten selectoren:

- Een selector voor *HTML-tags*.
- Selectoren voor elementen met een *id*.
- Selectoren voor elementen met een *class*.

In latere versies van CSS kunt u de selectoren verfijnen door bijvoorbeeld op te geven dat de regel alleen geldt voor tags van een bepaald type, zoals input-element van het type "text" (en dus niet voor een submit-knop input-element).

Er zijn steeds meer browsers die een goede ondersteuning voor CSS bieden, ook voor de laatste versie (versie 3), maar als uw website ook oudere browsers moet ondersteunen, dan is het raadzaam goed te onderzoeken wat u wel en niet kunt gebruiken.

1.4.1 Voorbeeld CSS-stijlelement:

```
C body {
    background: #f7f7f9;
    font-family: trebuchet ms;
    color: navy;
}
```

In dit voorbeeld zeggen we dat de achtergrondkleur voor de HTML-pagina binnen de hele *body* #f7f7f9 is, het lettertype is trebuchet ms, en de kleur van de tekst donkerblauw.

De beste en meest flexibele manier om stijlen te gebruiken is om een extern stijlbestand te maken, dat u als volgt in uw HTML-document kunt invoegen:

```
C <head>
    <title>LAMPjes Site</title>
    <link rel="stylesheet" type="text/css" href="lampjes.css" />
</head>
```

Als alle pagina's van een site van hetzelfde stijlblad gebruikmaken, dan is het eenvoudig om stijlveranderingen op een site door te voeren. Bovendien kan de HTML vaak korter en eenvoudiger worden als de stijl strikt gescheiden van de HTML in een stijlblad wordt opgeslagen. Voor het ontwikkelen van een stijlblad is het vaak nuttig om de stijl eerst in het HTML-document op te nemen, omdat het anders lastig experimenteren is met verschillende stijlopties.

LET OP DE BROWSERCACHE

Browsers doen vaak moeilijk over het verversen van hun cache omdat ze niet in de gaten hebben dat u de stijl hebt aangepast, zodat u zich achter het oor krabt waarom een wijziging niet werkt. In sommige browsers kunt u de cache uitschakelen, soms zult u echter de browser opnieuw moeten opstarten om wijzigingen te zien.

Denk ook aan het inzetten van Firebug bij het ontwikkelen van stijlbladen. Als het goed is, hebt u Firebug geïnstalleerd bij het lezen van het vorige hoofdstuk. Als u Firebug activeert, dan kunt u voor elk HTML-element precies zien wat de geldende stijlregels zijn, en hoe ze zijn overgeërfd of berekend. Ook kunt u tijdelijk en dynamisch veranderingen aanbrengen, zodat u kunt experimenteren met verschillende mogelijkheden.

1.4.2 Het attribuut id

Behalve voor HTML-tags van een bepaald type kunnen we ook stijl definiëren voor een specifiek element dat een *id* heeft. Denk er aan dat een id uniek moet zijn binnen een HTML-document. Een HTML-fragment met een id zou er zo uit kunnen zien:

```
c <div id="content-area ">
    <p>Deze site bevat informatie over het LAMP platform</p>
</div>
```

En we zouden een stijlregel kunnen maken als volgt:

```
c #content-area {
    border: solid 1px black;
    margin: 10px;
    padding: 10px;
    width: 200px;
}
```

We hebben hier gezegd dat het element met `id="content-area"` een dunne zwarte rand moet krijgen, het hele element is 200 pixels breed, er is een marge om het element van 10 pixels en een padding binnen het element van ook 10 pixels. Zie afbeelding 1.8 voor hoe het eruit zou kunnen zien.

1.4.3 Het attribuut class

Soms willen we een bepaalde klasse van elementen van een bepaalde stijl voorzien. Dit is een voorbeeld van een stijl, gedefinieerd door een class-attribuut:

```
c .contents {color: red;}
```

De HTML zou zoiets als het volgende kunnen bevatten:

```
c <div class="contents">
    <p>Deze site bevat informatie over het LAMP platform</p>
</div>
```

Alle elementen waarvoor `class="contents"` is gedefinieerd, zullen nu rood gekleurde tekst hebben.

1.4.4 Een samengesteld voorbeeld

Het is ook mogelijk om samengestelde regels te maken, bijvoorbeeld:

```
c div.contents p { font-style: italic; font-size: 24px; }
```

Deze regel zegt dat voor `p` (paragraph) tags die binnen een `div` met de class `"contents"` te vinden zijn, de tekst cursief moet worden weergegeven met een lettergrootte van 24 pixels.

We laten hier een voorbeeld zien waarbij we drie keer een `div` met dezelfde tekstinhoud hebben, eerst zonder `id` of `class`, dan een `div` met `class="contents"`, vervolgens een met `id="content-area"`, en ten slotte nog een `p`-tag met tekst, niet binnen een `div`, maar wel met `class="contents"`. De stijl is als volgt:

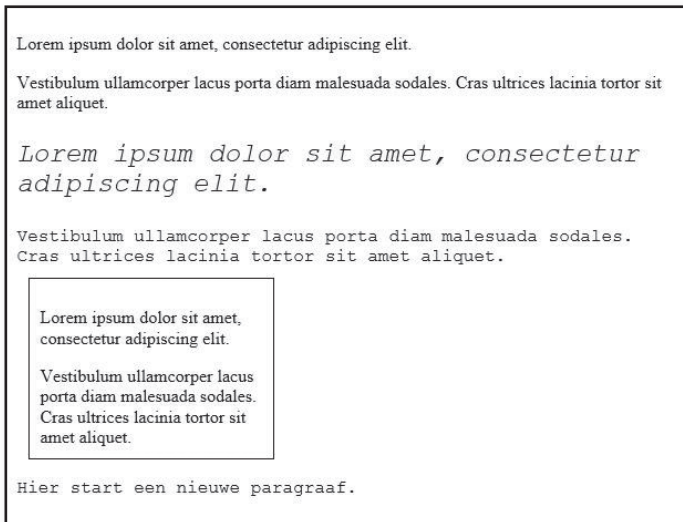
```
c body {border: solid 1px black; padding: 10px;
    width: 600px; margin: auto; margin-top: 10px;}
div.contents p { font-style: italic; font-size: 24px; }
.contents { font-family: courier; }
#content-area { border: solid 1px black; margin: 10px;
    padding: 10px; width: 200px; }
```

De HTML ziet er, enigszins verkort, als volgt uit:

```
c <body>
    <div>
        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
        Vestibulum ullamcorper lacus porta diam malesuada sodales.
        Cras ultrices lacinia tortor sit amet aliquet.
    </div>
    <div class="contents">
        <!-- inhoud hetzelfde als hierboven -->
    </div>
    <div id="content-area">
        <!-- inhoud hetzelfde als hierboven -->
    </div>
    <p class="contents">Hier start een nieuwe paragraaf.</p>
</body>
```

En dan nu het resultaat:

Let ook op de styling van de `body`-tag. Het komt vaak voor dat u een webpagina wilt maken die een beperkte breedte heeft, maar die wel in het midden van het browser scherm staat, onafhankelijk van de breedte daarvan. De truc is om dan `margin: auto;` te gebruiken, er moet dan wel ook een `width` gedefinieerd zijn.



Afbeelding 1.7
Invloed van CSS op
opmaak voor 3 div's

1.4.5 Een tabel mooier maken met CSS

We willen u een paar trucjes om een tabel mooier te maken niet onthouden. We nemen hetzelfde HTML-document dat ons afbeelding 1.3 opleverde (*bonushoofdstuk01/tabel.html*) en we voegen het volgende stijlblad toe aan de head-sectie:

```

<style type="text/css">
  table { border-collapse: collapse;}
  td, th {
    border: 1px solid #96969D;
    padding-left: 5px;
    padding-right: 5px;
  }
  tr.odd { background-color:#CFDBEC;}
  th { background-color: #3399FF; color: white;}
</style>

```

Verder voegen we aan de tr-tags van de oneven rijen in de tabel toe: `class="oneven"` en halen we het attribuut `border` bij de table-tag weg. Bekijk *bonushoofdstuk01/tabel-css.html* voor het eindresultaat. In onze browser ziet de tabel er nu zo uit:

kop1	kop2	kop3
a1	a2	a3
b1	b2	b3
c1	c2	c3

Afbeelding 1.8
Een tabel na toepassing van CSS.

Dat is heel wat beter, nietwaar? Laten we naar de CSS kijken om te begrijpen hoe we dit voor elkaar hebben gekregen. De eerste regel, `table { border-collapse: collapse; }`, geeft aan dat de borders van de cellen moeten samenvallen. Als u dit niet doet, dan komt er een kleine ruimte tussen de cellen. U kunt dit gemakkelijk proberen door `/* en */` om de regel heen te zetten, waardoor het commentaar wordt, en daarna de HTML in uw browser op te roepen. Dan zijn er een paar regels die een lichtgrijze rand maken om de cellen en die de cellen links en rechts wat ruimte geven. De rijen die we van `class="oneven"` hebben voorzien, krijgen een lichtblauwe achtergrondkleur, en de kolomkoppen (de `th`-tags), krijgen een felblauwe achtergrond.

1.4.6 Formulieren en CSS

Als afsluiting van dit stukje over CSS willen we het formulier dat we zagen in paragraaf 1.3 mooier maken, zodat het eruit ziet zoals in afbeelding 1.4. Als u naar de bron-code kijkt van veel formulieren op het web, dan zult u ontdekken dat er veel van tabellen gebruikt wordt gemaakt om de velden van formulieren netjes onder elkaar te krijgen. Dat is natuurlijk erg verleidelijk, zeker als u eerst uren geworsteld heeft met floating div's enzovoorts en dan met een kleine wijziging uw hele structuur weer in de war gooit. Maar het hoort niet, tabellen zijn er om tabelgegevens te laten zien en niet om formulieren te formatteren. Er zijn ook korte oplossingen, waarvan we er één hieronder laten zien. De auteur ervan zegt dat u ontslagen wordt als u hierna ooit nog een tabel gebruikt om een formulier te formatteren, echter, als u bijvoorbeeld lange tekst voor een formulierveld heeft staan, dat in meerdere regels getoond moet kunnen worden, dan zou ik nog geen andere manier weten. In minder exotische gevallen is de styling hieronder heel effectief. Om hem te kunnen toepassen moeten we ook de HTML aanpassen:

```
C <form action="foo.php" method="get">
    <div class="field_container">
        <label for="username">voor- en achternaam:</label>
        <input type="text" name="username" id="username">
    </div>
    <div class="field_container">
        <label for="email">email:</label>
        <input type="text" name="email" id="email">
    </div>
    <div class="field_container">
        <label for="submit">&nbsp;</label>
        <input type="submit" name="submit" value="Stuur op!" />
    </div>
</form>
```

We hebben de labeltekst in een label-tag geplaatst. Dat stelt ons in staat om de tekst een vaste breedte te geven en dat is de oplossing om formulierelden netjes in het gelid te krijgen.

De label-tag stelt een *for*-attribuut verplicht, dat aangeeft op welk veld het label van toepassing is. Het *for*-attribuut grijpt aan op de *id* van een veld, dus moeten we ook aan elk veld *id*'s toevoegen. Dat is eigenlijk jammer, want met veel formulieren en veel velden komt er ook een onoverzichtelijke hoeveelheid *id*'s, die ook nog eens uniek moeten zijn. Maar het voordeel is nu, dat als u op labeltekst klikt, het bijbehorende formulerveld focus krijgt. Om velden met hun labels bij elkaar te houden zijn ze binnen *div*'s geplaatst met `class="field_container"`. Zonder verdere omhaal is hier de CSS tekst:

```
C label { width: 200px; float: left; text-align: right; padding-right: 10px;}
    form { width: 450px; border: solid 1px; padding-bottom: 5px;
          font-family: verdana;}
    .field_container { padding-top: 5px;}
    input[type="text"] { width: 200px; }
```

Kijk maar in *bonushoofdstuk01/simpelform2.html* en laat het in een browser zien. U zult nu afbeelding 1.4 te zien krijgen.

1.5 JavaScript en jQuery

We zullen heel kort zijn over JavaScript en jQuery. Misschien is het wel zo dat u voor een webpagina die aan de eisen van vandaag kan voldoen, meer tijd kwijt ben met het schrijven van JavaScript- en/of jQuery-code (wat ook JavaScript is) dan met PHP-code, maar dit boek gaat daar nu eenmaal niet over. We zullen later in dit boek wel wat jQuery-code tegen komen, met name om Ajax-communicatie mogelijk te maken, maar we zullen u niet vragen die code zelf te schrijven.

1.5.1 JavaScript

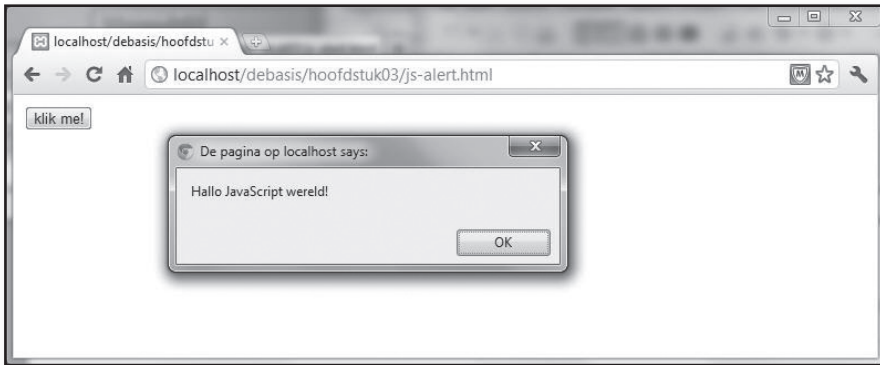
JavaScript is de webtaal waarmee u dynamisch dingen in de browser kan veranderen. U kunt animaties maken, HTML-elementen aanpassen en formulieren checken op geldige inhoud. Maar let op! Als u iets checkt met JavaScript, betekent dat niet dat u hetzelfde niet nogmaals moet checken in PHP, want een hacker kan altijd de informatie aanpassen voordat het naar de server gestuurd wordt.

JavaScript lijkt wat betreft syntaxis op PHP, maar als taal is het ook weer zo verschillend dat u zich makkelijk kunt vergissen. Objecten in JavaScript zijn anders dan objecten in PHP, sommige ingebouwde functies zijn hetzelfde, maar de meeste niet. Als u in JavaScript elementen van de webpagina wilt veranderen, dan krijgt u te maken met de interne structuur van het HTML-document, het zogenaamde *Document Object Model*, of *DOM*. Er is een toenemend aantal boeken, dat u graag uit de doeken wil doen hoe dit allemaal in elkaar zit, dus hier beperken we ons tot een eenvoudig "Hallo wereld" voorbeeld.

```
C <html>
    <head>
```

```
<script type="text/javascript">
    function welkom() {
        alert("Hallo JavaScript wereld!");
    }
</script>
</head>
<body>
    <input type="button" onclick="welkom()" value="klik me!" />
</body>
</html>
```

Dit is een erg simpele pagina met één knop er op. Als u daar op klikt, dan verschijnt er een pop-upschermd met "Hallo JavaScript wereld!" erin:



Afbeelding 1.9
Hallo JavaScript!

Als u goed naar de HTML-code kijkt, dan zal u opvallen dat de input-tag een extra attribuut heeft gekregen: `onclick="welkom()"`. Als op de knop wordt geklikt dan ziet de browser dat. In dit geval is het een *onclick event*. Met het attribuut *onclick* geeft u aan wat de browser moet doen als het click event optreedt.

EVENT

Event is weer zo'n mooi Neder-Engels woord dat we niet vertalen in dit boek. Het betekent natuurlijk *gebeurtenis*, en we spreken hier meestal over bewegingen en klikken met de muis over het browserscherm.

De bekendste events zijn: *click*, *mouseover*, *mousedown*, *mouseup*.

We hebben aangegeven dat als het click event optreedt, de functie `welkom()` aangeroepen moet worden. In deze functie staat de aanroep van de in JavaScript ingebouwde functie `alert()` en deze veroorzaakt het verschijnen van het pop-upschermpje.

Een nadeel van deze aanpak is dat er nu mogelijk allerlei JavaScript-code in ons HTML-document terecht komt en dat betekent dat iemand die het gebruik van JavaScript heeft uitgeschakeld, mogelijk niets te zien krijgt.

1.5.2 jQuery

jQuery is een JavaScript-raamwerk. Het adverteert zichzelf als “write less, do more” JavaScript-raamwerk. Het is dus bedoeld om het schrijven van JavaScript gemakkelijker te maken. En dat doet het inderdaad, want als u 17 keer `getElementsByClassName("my-class")` hebt geschreven, of een andere lange functienaam, dan is het een verademing om `$('.myclass')` te kunnen zeggen, wat in jQuery hetzelfde betekent. Er zijn veel JavaScript-raamwerken in omloop, maar jQuery is waarschijnlijk het meest populair op dit moment, het is snel en compact en er is veel documentatie over te vinden. Wat doet jQuery?

- Het werkt met de meeste browsers, zelfs Microsoft Internet Explorer 6, die berucht is om compatibiliteitsproblemen.
- jQuery ondersteunt CSS3, zelfs als de browser dat zelf niet doet.
- Het maakt navigatie door een HTML-document makkelijker.
- Het zorgt voor eenvoudige event-afhandeling.
- Het kan animaties maken.
- Het ondersteunt Ajax-interactie.

Voor al dit laatste punt, de Ajax-interactie, maakt het een stuk gemakkelijker om interactieve websites te ontwikkelen die in alle moderne browsers werken. In het hoofdstuk over browser-serverinteractie vertellen we daar meer over. Hier laten we alleen een erg eenvoudig voorbeeld zien, hetzelfde voorbeeld als in de vorige paragraaf, maar nu met jQuery en met het verschil dat we geen alert pop-up laten zien, maar de tekst in het HTML-document toevoegen.

```
C <html>
    <head>
        <title>jQuery Hallo wereld</title>
        <!-- voeg de jQuery library toe -->
        <script type="text/javascript" src="lib/jquery-1.6.2.min.js"
            charset="utf-8"></script>
        <script type="text/javascript">
            $(document).ready(function() {
                $("input").click(function() {
                    $('#hallo-area').append('<br/>Hallo jQuery wereld!');
                });
            });
        </script>
    </head>
</html>
```

```

        </script>
    </head>
    <body>
        <input type="button" value="klik me!" />
        <div id="hallo-area">
        </div>
    </body>
</html>

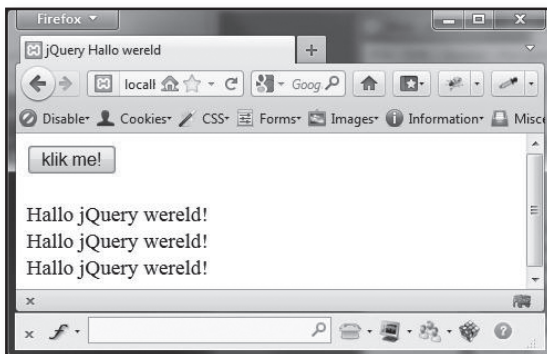
```

U ziet dat er nu geen JavaScript-code tussen de HTML staat, wat betekent dat de pagina nog steeds te zien is als JavaScript is uitgeschakeld. Dat is in dit voorbeeld weinig zinvol, maar voor grote sites die voor een breed publiek toegankelijk moeten zijn, waaronder mensen die slecht of helemaal niet kunnen zien, of mobiele apparaten die geen JavaScript ondersteunen, kan dit van groot belang zijn. Zoekt u eens naar “*progressive enhancement*” als u meer over dit onderwerp wil weten.

We hebben de jQuery-bibliotheek geactiveerd door het bestand waarin hij te vinden is, `lib/jquery-1.6.2.min.js`, in te voegen. Overtuig u ervan dat dit bestand ook echt staat waar u zegt dat het staat. Geloof me, dat zal u veel hoofdbreken besparen.

Code in jQuery begint vaak met: `$(document).ready(function() {...})`. Dit betekent dat de code tussen de gekrulde haken pas uitgevoerd mag worden als het complete HTML-document geladen is door de browser. Dat voorkomt dat de code naar elementen gaat zoeken die nog niet geladen zijn.

In plaats van het attribuut `onclick` hebben we nu `$("#input").click(function() {...})`. In feite zal de code binnen de functie geactiveerd worden voor alle inputelementen in het HTML-document. Dat zou u kunnen veranderen door de knop een id te geven. Onder de knop vindt u een div met de id “hallo-area”. In de code in de clickfunctie staat dat we bij ieder click event de tekst: ‘`
Hallo jQuery wereld!`’ toevoegen onderaan de div. Ik denk dat de code verder voor zichzelf spreekt. Nadat u 3 keer op de knop geklikt hebt, zal uw browser-venster er als volgt uitzien:



Afbeelding 1.10
Hallo jQuery!

Natuurlijk raakt dit voorbeeld nauwelijks aan de oppervlakte van wat u met jQuery kunt doen, we hopen alleen dat u een indruk hebt gekregen van de structuur en stijl van een HTML-document waaraan jQuery code is toegevoegd. We sluiten het hoofdstuk af met een paar oefeningen waarmee u voor uzelf kunt nagaan of uw kennis van HTML en CSS voldoende is voor het maken van aardige webpagina's.

OEFENING 1.2

Neem het HTML-document dat u gemaakt hebt met oefening 1.1. Maak er iets moois van! Als toegift: als u met uw muis boven uw foto komt, verschijnt er 'Hallo ...(vul hier uw naam is)..'. Hint: gebruikt het mouseover event. Tweede toegift: nu met jQuery.

OEFENING 1.3

Probeer afbeelding 1.4 te reproduceren, door het formulier uit paragraaf 1.3 in een HTML-tabel te zetten.

OEFENING 1.4

Neem het resultaat van het HTML-documenten uit oefening 1.2 en 1.3. Maak er één document van, waarbij uw persoonlijke gegevens in het midden komen te staan, en het formulier links. Maak er een paginakop boven met groot uitgevoerde tekst: 'Dit is de pagina van ...' (vul uw naam in). Probeer enkele variaties in de layout, met tabellen of met CSS.