

html 5

programmeren in HTML (5) CSS (3) JavaScript



Op dit lesmateriaal is een Creative Commons licentie van toepassing.

2010 Remie
remie.woudt@gmail.com

Woudt

Inhoudsopgave

1 HTML.....	1
1.1 Wat heb je nodig?.....	1
1.2 De basis.....	1
1.3 Tekst structureren.....	5
1.4 Blok- en inline elementen.....	9
2 De browser.....	11
3 CSS.....	15
3.1 De selector.....	15
3.2 Links: kleuren, decoratie en achtergrond.....	18
3.3 Het CSS box model.....	21
3.4 Nav in CSS.....	23
4 Links.....	26
4.1 Interne link	26
4.2 Externe link	26
4.3 Anchor URL.....	27
4.4 Afbeelding.....	27
5 Tabellen.....	29
6 Intermezzo.....	33
6.1 De programmeertaal.....	33
6.2 Programmastructuur diagrammen.....	33
6.3 Werken met Structorizer.....	35
6.4 Variabelen.....	38
6.5 Operatoren.....	40
6.6 Functies en methoden.....	41
6.7 Commentaar.....	41
6.8 Tenslotte.....	42
7 Interactief.....	43
7.1 JavaScript tussen HTML.....	44
7.2 Actie.....	44
7.3 Formulier.....	45
7.4 Functies.....	46
7.5 Keuze.....	48
7.6 Objecten.....	50
8 DOM.....	53

1 HTML

Html is de basis voor het maken van websites

1.1 Wat heb je nodig?

Hoewel je op heel veel manieren een website kunt bouwen is kennis van HTML altijd nodig. Vandaar dat we beginnen met een cursus HTML.

Om met HTML te kunnen werken hebben we een editor nodig waarmee we de HTML pagina's kunnen maken. Er zijn veel (vaak gratis) editors via internet te downloaden. De meest gebruikte voor Windows is wel Notepad++. Te downloaden via

<http://sourceforge.net/projects/notepad-plus/files/>

Eenmaal gedownload is het een kwestie van installeren door er op te dubbelklikken.

Wil je de editor liever gebruiken vanaf je usb-stick, maak dan gebruik van deze link:

http://portableapps.com/apps/development/notepadpp_portable

In deze cursus zullen in de voorbeelden van de code steeds kleuren worden gebruikt zoals je die ook ziet als je met Notepad++ werkt.

editor

Notepad++

1.2 De basis

HTML is de afkorting van HyperText Markup Language. Met HyperText wordt tekst bedoeld met directe verwijzingen in de vorm van aanklikbare links. Je hoeft dus niet meer via een index te zoeken maar kunt rechtstreeks van de ene pagina naar de andere springen. Markup betekent dat de taal uit markingstekens bestaat waarmee je aangeeft hoe de tekst opgemaakt moet worden.

hypertext

markup

Wanneer we een HTML-pagina (zeg maar een internet pagina) willen maken doen we dat door in een editor met gewone tekst en een aantal opdrachten aan te geven hoe de pagina eruit moet gaan zien. Willen we gewone tekst laten zien dan kunnen we dat ook min of meer normaal typen. Maar we hebben wel codes nodig die vertellen hoe de tekst eruit komt te zien en waar de tekst komt te staan. Bijvoorbeeld:

<title>Mijn eerste pagina</title>

Wat je hier ziet noemen we een element. In dit geval het element waarmee de titel van een pagina wordt gemaakt. Een element bestaat uit tags met daartussen tekst of iets anders dat je op jouw HTML-pagina wilt hebben. **<title>** en **</title>** zijn de tags.

tag

Tags staan altijd tussen `<` en `>`. Je hebt vaak een begintag en een eindtag die aan elkaar gelijk zijn behalve dat de eindtag tussen de haken begint met een `/`.

We kunnen echter niet volstaan met alleen maar elementen op de pagina te plaatsen. We zullen eerst moeten aangeven wat voor soort pagina het is. En daarnaast moeten we met de pagina ook de indeling en plaatsing van de elementen vastleggen.

Om een webpagina te maken moet je je houden aan een aantal regels. Hoewel de meeste browsers erg soepel zijn als je er niet aan houdt is het uiteraard alleen maar verstandig die regels zo goed mogelijk toe te passen.

HTML 5 *Let op: in deze cursus houden we ons aan de regels die gelden voor HTML5. Niet iedere browser ondersteunt HTML5. Met name Internet Explorer blijft op dit gebied nog erg achter. Het is dus verstandig tijdens deze cursus te werken met browsers die HTML5 wel goed ondersteunen. Dit zijn bijvoorbeeld Mozilla Firefox, Google Chrome, Opera en Safari. Overigens wordt op pagina 12 aangegeven hoe je de meeste HTML5 tags ook in een oudere browser goed kunt verwerken.*

Wanneer we beginnen met een HTML 5 pagina wordt op de eerste regel altijd het type document aangegeven. Dat doe je met:

<!DOCTYPE> **<!DOCTYPE html>**

Hiermee geef je aan dat de pagina van het documenttype HTML is. Je kunt hier ook een ander type gebruiken en afhankelijk van het type zal de browser strenger of minder streng zijn in het toepassen van de regels.

Overigens is **<!DOCTYPE>** de enige tag die gewoonlijk met hoofdletters geschreven wordt hoewel dit niet verplicht is in HTML 5. Alle andere tags schrijven we in kleine letters alhoewel ook hiervoor weer geldt dat tags in hoofdletters net zo goed werken. Ook is **<!DOCTYPE>** van het void-type. Dat betekent dat deze tag geen eindtag heeft.

Na de **<!DOCTYPE>** tag volgt de eigenlijke webpagina. Iedere webpagina begint en eindigt met de HTML-tags. En binnen die HTML tags bevinden zich het head gedeelte en de body. In het head gedeelte komen vooral die zaken te staan die niet zichtbaar zijn op de pagina maar wel van belang zijn voor hoe de pagina op het scherm verschijnt. In de body komen de delen die straks wel zichtbaar zullen zijn.

Samengevat ziet een webpagina er dus tenminste zo uit:

```
<!DOCTYPE html>
<html lang="nl">
  <head>
    <meta charset="utf-8">
    <title>Mijn pagina</title>
  </head>
  <body>
    Hier komt de inhoud van de webpagina
  </body>
</html>
```

In het vervolg van deze cursus zul je nog veel zaken tegenkomen die in de head sectie of in de body komen. Maar altijd zal de opbouw van een pagina gelijk zijn aan dit voorbeeld.

Opdracht 1

Laten we eens beginnen met het maken van deze pagina.

- Start jouw editor (bijvoorbeeld Notepad++)
- Klik op Bestand – Nieuw
- Typ nu de code zoals hierboven staat exact in jouw editor

Maar... je ziet geen kleuren in jouw editor? Dat kan kloppen. De kleuren die je wel of niet ziet zijn afhankelijk van de instellingen van jouw editor. In Notepad++ kun je aangeven dat je bezig bent met een HTML-pagina. Dat doe je via de keuzes: Syntaxis – H – HTML. Daardoor weet Notepad++ dat je werkt aan een HTML-bestand. Het kan overigens nog eenvoudiger. Door het bestand te bewaren met de extensie `.html` snapt Notepad++ automatisch dat het om een HTML bestand gaat.

- Bewaar jouw bestand onder de naam `opdracht1.html`
- Bekijk het bestand nu in jouw browser, bijvoorbeeld door er in de verkenner met de rechter muisknop op te klikken en dan te kiezen voor openen

En, zie je je pagina? Mocht je foutmeldingen krijgen dan moet je de code nog eens goed door-nemen en kijken of je alles wel precies zo hebt ingetypt. Maar laten we eerst even kijken wat je nu eigenlijk aan code hebt gemaakt.

Allereerst is het belangrijk dat begin- en eindtags steeds recht onder elkaar staan. Alleen wanneer de begin- en eindtag op één regel staan hoeft dat niet (al mag het wel). Doe je dat niet dan zal de webpagina net zo goed werken. Maar als je dan fouten moet zoeken wordt dat heel erg lastig. Dus zorg voor een goede structuur in de opbouw van jouw website in de editor. In een goede editor kun je instellen dat dit automatisch gebeurt.

Alle tags tussen de begin- en eindtag worden ingesprongen. Dat zorgt ervoor dat de code goed leesbaar is. In Notepad++ kun je ook zo stukjes code in elkaar klappen door op het minnetje aan het begin van de regel te klikken.

Als je met de cursor een tag aan klikt zie je in Notepad++ ook de bijbehorende begin- of eindtag oplichten. Zo kun je ook heel snel zien welke tag bij welke hoort. Voor een stukje code zoals je hier ingetypt hebt lijkt dat niet zo belangrijk maar uiteraard komt er bij een gevulde website nogal wat code bij. En dan is het heel prettig om te zien of jouw structuur goed is.

En dan de regels. De pagina begint met:

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

De HTML tag dus maar er wordt meteen mee aangegeven dat het om een pagina in het Nederlands gaat. Aangezien je kunt zoeken op pagina's die in een bepaalde taal geschreven zijn en bovendien moderne browsers ook pagina's kunnen vertalen is het handig om aan te geven in welke taal de pagina geschreven is.

lang noemen we hier het attribuut en **"nl"** is dan de attribuutwaarde.

attribuut
attribuutwaarde

Daarna komt de head sectie:

```
<head>
  <meta charset="utf-8">
  <title>Mijn pagina</title>
</head>
```

Zoals je ziet wordt binnen de head sectie alles ingesprongen. Overigens had je het **<title>** deel ook op de volgende manier kunnen invoeren:

```
<head>
  <meta charset="utf-8">
  <title>
    Mijn pagina
  </title>
</head>
```

Maar wanneer begin- en eindtag op één regel passen doen we dat meestal niet.

Met **<meta charset="utf-8">** geven we informatie over de website. De meta-tag wordt voor allerlei informatie gebruikt. Zoekmachines gaan vaak op zoek naar de meta informatie van

de website om die zo te indexeren. In dit voorbeeld wordt met de meta tag aangegeven in welke karakterset de pagina geschreven is en dus moet worden afgebeeld. In de utf-8 karakterset zijn de unicode karakters vastgelegd. Daarmee kunnen we er redelijk vanuit gaan dat alle karakters wereldwijd in de browsers goed worden afgebeeld.

Met `<title>Mijn pagina</title>` wordt de paginatitel vastgelegd. In de huidige browsers met al die tabs is de pagina titel vaak niet eens zo goed zichtbaar. Toch is het wel prettig als een pagina een titel heeft.

Zoals je verder gezien hebt volgt na de head sectie de body sectie. Daar staat in dit voorbeeld geen bijzondere code in.

Samenvatting

- Een tag staat altijd tussen `<` en `>`
- Bij een tag hoort (vrijwel altijd) een afsluitende tag, herkenbaar aan de `/` voor het woord. Wanneer begin- en eindtag op meerdere regels staan zorg je ervoor dat de code daartussenin ingesprongen wordt.
- Maak er een gewoonte van de tags altijd in kleine letters te typen.
- Een HTML-pagina bestaat altijd uit een head en een body
- Een HTML-pagina sla je altijd op met de extensie `.html` of `.htm`.

Opdracht 2

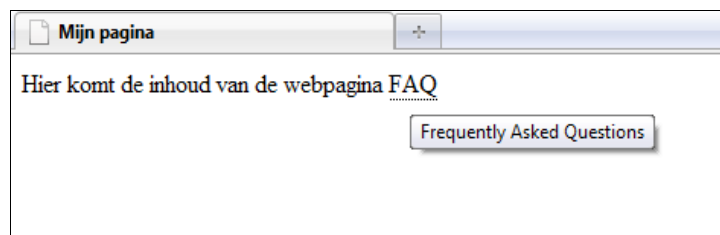
Geef een beschrijving van de volgende termen:

- element
- tag
- attribuut
- attribuutwaarde

Opdracht 3

Voeg de regel: `<abbr title="Frequently Asked Questions">FAQ</abbr>` toe aan de body van jouw eerste pagina en bekijk het resultaat. Ga daarna, zonder te klikken, met je muis over FAQ. Wat gebeurt er?

Als het goed is zie je iets soortgelijks gebeuren als hiernaast.



De letters FAQ kwamen meteen achter jouw tekst te staan. Niet even netjes op een nieuwe regel.

Afbeelding 1: opdracht 3

Een nieuwe regel krijg je met de tag `
` (br is de afkorting voor break of linebreak). Zet in jouw pagina ergens tussen de regel: `"Hier komt de inhoud van de webpagina"` en `"FAQ"` de tag `
` en bekijk het resultaat.

linebreak

Opdracht 4

- a Wat is in de `<abbr>` tag nu het attribuut en wat is de attribuutwaarde?
- b Bedenk nog een toepassing voor de `<abbr>` tag.

Je weet nu:

- Wat een element, een starttag, een eindtag, een attribuut en een attribuutwaarde is
- Hoe je een HTML-pagina aan kunt maken, opslaan, wijzigen in jouw editor
- Hoe je een HTML pagina kunt bekijken in jouw browser

1.3 Tekst structureren

Wanneer we tekst op een website weergeven, willen we die uiteraard niet gewoon achter elkaar zetten. Bovendien willen we de site ook wat aantrekkelijker maken, bijvoorbeeld met verschillende lettertypen of lettergrootten. In HTML hebben we daar een beperkt aantal mogelijkheden voor.

Opdracht 5

Maak een HTML-pagina met de volgende regels in de body:

```
<h1>Het laatste nieuws</h1>
<h2>Inflatie in januari hoger door energieprijzen</h2>
De Nederlandse inflatie is in januari 2010 uitgekomen op 0,8
procent.
De inflatie bedroeg in april 2010 1,1 procent.
```

Sla de pagina op onder de naam opdracht5.html en bekijk hem.

Als het goed gegaan is zie je duidelijk de effecten van `<h1>` en `<h2>`. Deze zijn niet zozeer bedoeld om grotere letters te maken maar vooral om bepaalde koppen aan te geven. Daarmee kun je een overzichtelijker geheel van jouw pagina maken. Verderop in deze cursus ga je leren hoe je die koppen ieder hun eigen stijl kunt gaan geven.

De tekst daaronder staat nog wel raar achter elkaar terwijl je toch vast wel op Enter hebt gedrukt toen je de tekst intypte. Nu hebben we daar in de vorige paragraaf al een oplossing voor gezien met de `
` tag. Maar een veel betere oplossing krijg je door gebruik te maken van de alinea indeling met behulp van de paragraph tag oftewel de `<p>` tag.

paragraph

Zet nu de volgende tags in de tekst:

```
<p>De Nederlandse inflatie is in januari 2010 uitgekomen op 0,8
procent.</p>
<p>De inflatie bedroeg in april 2010 1,1 procent.</p>
```

Bekijk het effect ervan. Je ziet dat je met de `<p>` tag de tekst kunt onderverdelen in alinea's. Je zult je misschien afvragen waarom dat zo ingewikkeld moet. Waarom niet gewoon op Enter drukken als je een nieuwe regel wilt beginnen?

Het probleem met een website is dat je vooraf niet weet hoe groot het venster zal zijn. Er zijn computers met kleine schermpjes maar ook met grote breedbeeld schermen. Bovendien kun je jouw browserscherm zelf vergroten of verkleinen. Om hiermee rekening te houden moet de tekst zich kunnen aanpassen aan de schermgrootte. Maak bij de bovenstaande pagina het scherm maar eens kleiner en bekijk wat er met de tekst gebeurt. Iedere keer wanneer jij van het venster van jouw browser de grootte verandert, wordt opnieuw bepaald hoe de tekst daarin moet verschijnen.

Een andere manier om structuur aan te brengen in jouw HTML-pagina is met behulp van de div tags. Div is een afkorting voor division en daarmee wordt dan een gedeelte van de webpagina bedoeld.

division

Opdracht 6

Maak een HTML-pagina met de volgende code:

```
<!DOCTYPE html>
<html lang="nl">
  <head>
    <meta charset="utf-8">
    <title>De groentenman</title>
  </head>
  <body>
    <div id="bovenste">
      <h1>Dit is het bovenste deel van de webpagina</h1>
    </div>
    <div id="menu">
      <ul>
        <li>Appels</li>
        <li>Peren</li>
        <li>Bananen</li>
      </ul>
    </div>
    <div id="inhoud">
      <p>Hier komt de inhoud van de webpagina</p>
    </div>
  </body>
</html>
```

Sla de pagina op onder de naam `opdracht6.html` en bekijk hem.



Afbeelding 2: Opdracht 6

Tussen `<div>` en `</div>` bevindt zich steeds een gebied van jouw webpagina. Dat gebied geven we een naam met `id=...`. Daarmee kunnen we dat gedeelte van die website later ook weer aanroepen. Zo zou je iedere div-gedeelte van jouw pagina een andere opmaak kunnen geven. Maar ook kun je van een pagina alleen een div-gedeelte met een bepaalde naam vernieuwen.

Naast de div-tags zien we in deze pagina nog wat nieuws, de `` tag met daarbinnen de `` tag. Met `` krijgen we een zogenaamde unordered list oftewel een ongeordende lijst en met `` geven we de elementen van die lijst aan. Je ziet ook dat die elementen tussen de begin- en eindtag staan van de ongeordende lijst. Een gewone unordered list herken je vaak aan de zwarte stippen of bullets ervoor.

unordered list

Het tegenovergestelde van de unordered list is de ordered list. Deze geeft je aan met de `` tag. Wijzig maar eens de `` tag in de `` tag (vergeet niet de eindtag ook aan te passen!) en bekijk het resultaat.

ordered list

Zet hem daarna wel weer terug naar de unordered list want die gaan we verderop nog gebruiken.

Dat lijkt leuk, dat gebruik van div's. Maar al te veel gebruik ervan maakt het er niet overzichtelijker op. Sommige pagina's staan vol met allerlei geneste div's. Met geneste div's worden div's bedoeld die weer binnen andere div's staan. Lang niet altijd is een div nodig. Vaak is een ander blok-element (zie pagina 9) ook genoeg om de benodigde structuur aan te brengen. Programmeurs noemen het overdadige gebruik van div's wel divitis. HTML5 geeft geen oplossing voor divitis. Wel is er een aantal nieuwe div-achtige tags die beter aangeven waar zo'n tag voor staat.

block element
divitis

Opdracht 7

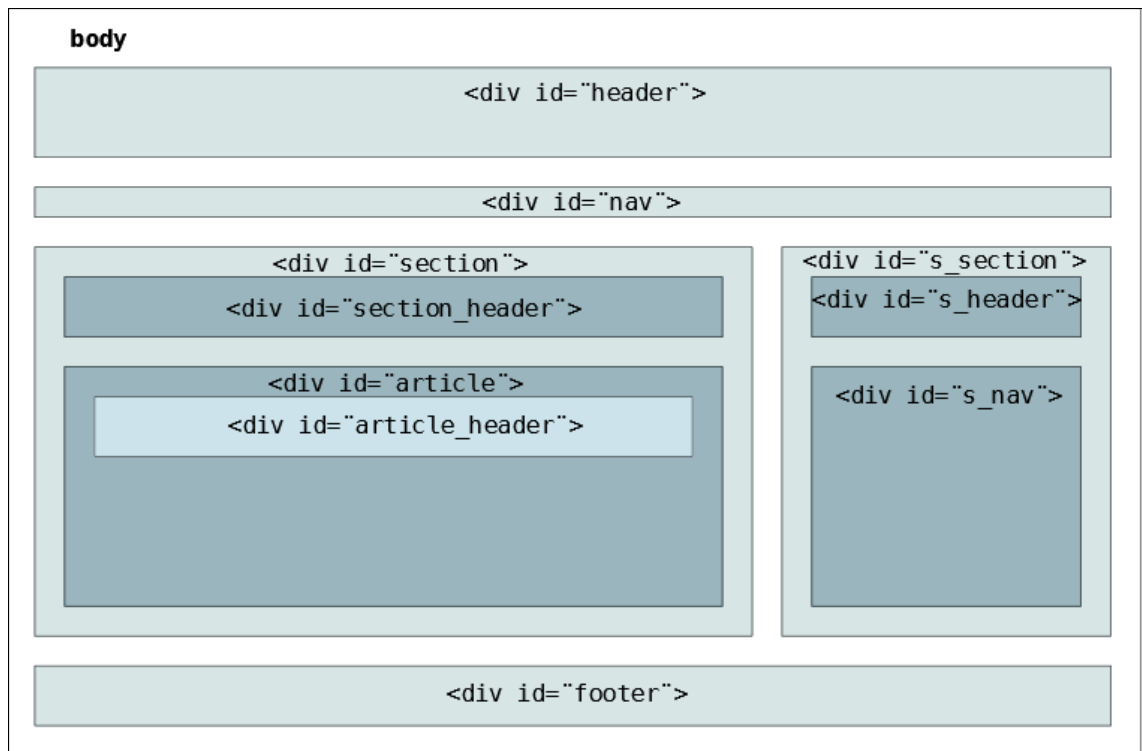
Wijzig opdracht 6 als volgt:

```
<!DOCTYPE html>
<html lang="nl">
  <head>
    <meta charset="utf-8">
    <title>De groentenman</title>
  </head>
  <body>
    <header id="bovenste">
      <h1>Dit is het bovenste deel van de webpagina</h1>
    </header>
    <nav id="menu">
      <ul>
        <li>Appels</li>
        <li>Peren</li>
        <li>Bananen</li>
      </ul>
    </nav>
    <section id="inhoud">
      <p>Hier komt de inhoud van de webpagina</p>
    </section>
  </body>
</html>
```

Sla de pagina op onder de naam `opdracht7.html` en bekijk hem.

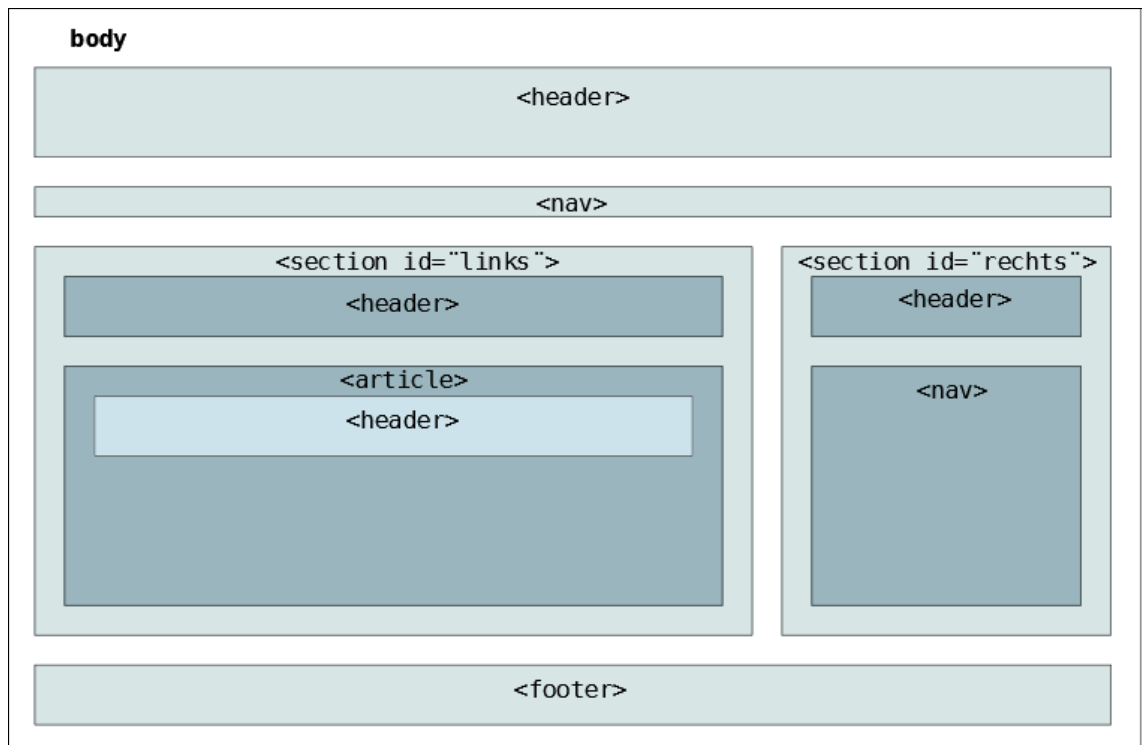
Hé, geen verschil met opdracht 6. Maar wel een verschil in de code. De div-tages zijn vervangen door de nieuwe tags `<header>`, `<nav>` en `<section>`. De namen van deze tags geven beter aan waarvoor ze bedoeld zijn en hebben daarvoor ook aanvullende eigenschappen. Er zijn overigens nog meer tags die div's kunnen vervangen. Zoals de `article`-tag in afbeelding 4.

Laten we nog eens even kijken naar het verschil tussen het gebruik van div zoals dat in HTML4 gebeurde ten opzichte van de nieuwe tags in HTML5. De volgende afbeeldingen maken het duidelijker:



Afbeelding 3: Indeling webpagina met div's

Wat we zien zijn de vaste onderdelen die heel vaak op een webpagina voorkomen. In HTML4 zijn er alleen via div's mogelijkheden om deze onderdelen apart aan te duiden. In HTML5 gaat het zo:



Afbeelding 4: Indeling webpagina met de nieuwe html 5 tags

Je ziet dat het nogal wat overzichtelijker wordt. In plaats van onduidelijke `div`'s heeft nu ieder deel van de webpagina zijn eigen type tag. Overigens kun je vaak deze tags ook weer in elkaar gebruiken. Een `<section>` kan weer een `<header>` hebben enz.

Wel even opletten: `<head>` en `<header>` zijn verschillende tags met ieder een heel andere functie. `<head>` geeft informatie over de webpagina terwijl `<header>` een kop is binnen de body of een gedeelte van de body!

Je weet nu:

- Hoe je structuur aan kunt brengen in jouw website
- Hoe je geordende en ongeordende lijsten kunt maken
- Hoe je `div`'s gebruikt en welke alternatieve tags daarvoor zijn in HTML5.

1.4 Blok- en inline elementen

Het lijkt wel leuk, die plaatjes over hoe je een pagina kunt indelen, maar onze pagina heeft nog helemaal geen indeling. Alles komt onder elkaar, er is geen sprake van secties of een navigatie-blok. Maar de opmaak en de layout van een pagina doen we ook niet met HTML. Daarvoor hebben we CSS. Maar voor we (in hoofdstuk 3) aan CSS beginnen eerst nog wat andere zaken die we nodig hebben voor het gebruik ervan. Belangrijk is de beide soorten HTML elementen te herkennen.

Blok-elementen (block-level elements) hebben als het ware een eigen gebied binnen de HTML pagina. Een blok-element begint altijd op een nieuwe regel. Wel kunnen blok-elementen zich binnen andere blok-elementen bevinden maar ook inline elementen kunnen zich in een blok bevinden. Het bekendste blok-element is de `<body> ...</body>`. Andere voorbeelden zijn:

```
<div>...</div>
<header>...</header>
<section>...</section>
<article>...</article>
<p>...</p>
<ul>...</ul> enz.
```

**block-level
elements**

Inline elementen (inline elements of text-level elements) zijn elementen die je vaak in de tekst van een HTML pagina terugvindt. Ze bevatten tekst, gegevens of andere inline elementen. Ze beginnen meestal niet op een nieuwe regel.

Voorbeelden hiervan zijn `...` (accentueert tekst) en `<abbr>...</abbr>`

**inline elements of
text-level elements**

Eigenlijk is er nog een derde type element, de elementen die helemaal niets op het scherm laten zien. Voorbeelden hiervan zijn meta en head. Maar hoewel we die zeker moeten gebruiken en al hebben gebruikt, zijn ze niet van belang voor de opmaak van jouw pagina.

Wanneer we het straks gaan hebben over de layout van een pagina, dus hoe en waar de diverse delen van de pagina hun plek krijgen, dan heeft dat altijd betrekking op de blok-elementen! Maar voor we ons aan de opmaak gaan wagen, kijken we eerst even naar de browser.

Meer weten over HTML tags? Op het internet zijn duizenden sites te vinden die daarover gaan. Bijvoorbeeld <http://www.w3schools.com/html/>

Liever in het Nederlands?

<http://www.mijnhomepage.nl/htmlcursus/lessenoverzicht.php> of
<http://www.handleidinghtml.nl/>.

Helaas zijn deze nog niet afgestemd op HTML5 maar toch kun je er veel informatie vinden.

Je weet nu:

- Wat de eigenschappen van een block-level-element zijn
- Wat de eigenschappen van inline elementen zijn.

2 De browser

De ontwikkeling van HTML en CSS als een duo waarmee steeds fraaiere websites gemaakt kunnen worden zou uiteraard helemaal niets worden als niet ook continu de browsers worden verbeterd. De vijf belangrijkste browsers zijn:

- Internet Explorer (Microsoft)
- Firefox (Mozilla foundation)
- Chrome (Google)
- Safari (Apple)
- Opera

Maar de namen van de browsers zegt nog niet zoveel, belangrijker is nog welke versie het is. Uiteraard kan de meest recente versie van een browser het beste de nieuwste technieken weergeven.

Als we opdracht 7 een echte opmaak willen geven dan zou het er bijvoorbeeld in Firefox zo uit kunnen zien:



Afbeelding 5: De groentenman in Firefox 3.6.6

Ook Opera (versie 10.6), Safari (versie 5.0) en Chrome (versie 5.0) laten de pagina op dezelfde manier zien als Firefox. Maar bekijken we exact dezelfde code in Internet Explorer 8 dan ziet het er zo uit:



Afbeelding 6: De groentenman in Internet Explorer 8

Dus Microsoft loopt hierin wat achter maar werkt er hard aan om de achterstand in te halen. Zo is er al een bèta versie beschikbaar van Internet Explorer 9 en zie hieronder het resultaat:



Afbeelding 7: De groentenman in Internet Explorer 9

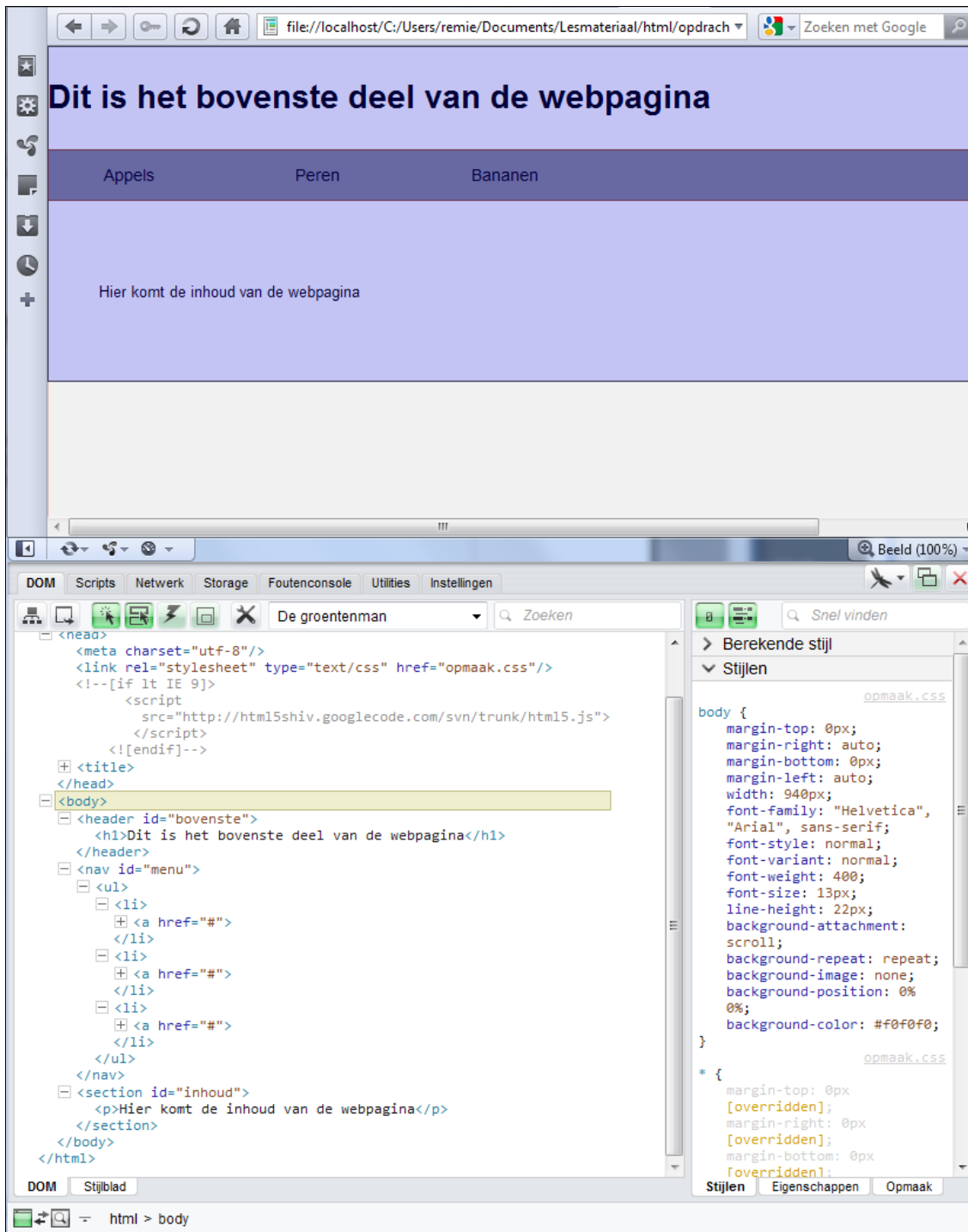
Met behulp van JavaScript, waar we in hoofdstuk 7 mee aan de slag gaan, is het overigens heel goed mogelijk de tekortkomingen van de diverse browsers op te vangen waardoor de website die jij bouwt in iedere browser goed werkt.

Wil je toch met een oudere Internet Explorer werken of wil je dat jouw moderne website ook in oudere versies van Internet Explorer goed werkt voeg dan de volgende code aan de head toe:

```
<head>
  <meta charset="utf-8" />
  <!--[if lt IE 9]>
    <script
      src="http://html5shiv.googlecode.com/svn/trunk/html5.js">
    </script>
  <![endif]-->
  <title>De groentenman</title>
</head>
```

Voor de rest van deze cursus wordt ervan uitgegaan dat je werkt met een moderne browser. Moet je nog een keuze maken dan is Opera een goede omdat deze standaard een toevoeging heeft (een zogenaamde DOM inspector met de naam Dragonfly) waarbij je webpagina's als het

ware van binnen kunt bekijken. Ook voor het zoeken naar fouten in JavaScript een onmisbaar hulpmiddel. Maar ook de andere browsers hebben zo'n DOM inspector of deze is eenvoudig toe te voegen.



Afbeelding 8: Dragonfly in Opera

Je weet nu:

- Dat browsers verschillen in hoe ze HTML code weergeven
- Hoe je in een oudere browser toch HTML5 elementen kunt gebruiken.

3 CSS

houd inhoud en opmaak altijd gescheiden!

Toen internet steeds populairder werd kwam ook de behoefte aan mooiere websites. HTML was daar niet echt voor gemaakt en dus zocht men naar een uitbreiding van HTML om de opmaak van de pagina's te kunnen verbeteren. De oplossing kwam met de mogelijkheid om style sheets te kunnen gaan gebruiken. Hoewel er meerdere soorten style sheets mogelijk zijn is CSS eigenlijk de enige die gebruikt wordt. CSS is de afkorting van Cascading Style Sheet. Style sheet zou je kunnen vertalen met Opmaak beschrijving. Let wel, het Nederlandse woord stijl is iets anders dan het Engelse style. Uitleggen van het begrip Cascading is wat lastiger omdat we dan in moeten gaan op begrippen die later pas aan de orde komen. Ga er simpelweg maar van uit dat we in een CSS-bestand de opmaak van de pagina beschrijven.

Cascading Style Sheet

Je kunt CSS op drie manieren toepassen namelijk:

- inline stylesheet waarbij je de opmaakcode meteen bij de tag plaatst
- internal (of embedded) stylesheet waarbij je de opmaakcode zoals die voor jouw pagina geldt in de head plaatst
- external stylesheet waarbij je de opmaakcode in een apart bestand met de extensie `.css` plaatst

Omdat het in de moderne manier van website bouwen heel belangrijk wordt geacht de inhoud en de opmaak van de website gescheiden te houden zullen we hier alleen gebruik maken van het externe opmaakbestand.

3.1 De selector

CSS code bestaat uit drie dingen:

- De selector
- Een property of eigenschap
- De waarde die die eigenschap moet krijgen

Met de selector wordt het element bedoeld waarvan je de opmaak wilt aangeven. Wil je bijvoorbeeld de opmaak van alle paragrafen, dus alle elementen met de tag `<p>`, hier bepalen dan wordt de selector `p`. Daarna komen binnen accolades de eigenschappen met hun waarden te staan. Dat ziet er dan zo uit:

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

selector

Willen we bijvoorbeeld alle `<h2>` gedeelten, dus allee tekst die tussen `<h2>` en `</h2>` staat de kleur rood meegeven, dan doe je dat met de volgende code:

```
h2 {  
    color: red;  
}
```

Voor een selector kun je tegelijkertijd van meerdere eigenschappen de waarde vastleggen:

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

Opnieuw een voorbeeld met de `<h2>` tag. Hier geef je naast een rode letterkleur ook meteen aan dat de tekst in italic (cursief) moet worden afgebeeld.

```
h2 {  
    color: red;  
    font-style: italic  
}
```

Maar je kunt ook van meerdere selectoren tegelijkertijd dezelfde waarde vast leggen:

```
selector1, selector2, selector3 {  
    eigenschap: waarde;  
    eigenschap: waarde;  
    eigenschap: waarde;  
}
```

Wil je zowel aan de `<h2>` blokken als aan de `<h3>` blokken dezelfde eigenschappen toekennen dan doe dat dus zo:

```
h2,h3 {  
    color: red;  
    font-style: italic  
}
```

Laat je de komma weg tussen meerdere selectoren dan betekent dat dat je de eigenschappen van selector 3, die een onderdeel is van selector 2 die weer een onderdeel is van selector 1 wilt vastleggen. Dus zo:

```
selector1 selector2 selector3 {  
    eigenschap: waarde;  
    eigenschap: waarde;  
    eigenschap: waarde;  
}
```

Deze laatste is het lastigst. In de code op bladzijde 24 wordt deze gebruikt. Dan zal ook die duidelijker worden. Laten we het maar gewoon proberen.

Opdracht 8

Maak een nieuwe html-pagina aan en geef deze de naam `opdracht8.html`. Typ de volgende code in die pagina:

```
<!DOCTYPE html>  
<html lang="nl">  
  <head>  
    <meta charset="utf-8">  
    <link rel="stylesheet" type="text/css">
```

```

    href="opdracht8.css">
    <title>Een pagina met stijl</title>
  </head>
  <body>
    <h2>Even kijken op een gestileerde pagina</h2>
    <p>
      Met de style die we in opdracht8.css hebben vastgelegd
      heeft de body lettertype Verdana gekregen en een
      achtergrond die lichtblauw is. Het gedeelte binnen
      &lt;h2&gt; heeft een rode kleur gekregen en het font is
      schuin gedrukt.
      Tenslotte hebben we iedere paragraaf de font variant small
      caps meegegeven wat kleine hoofdletters betekent.
    </p>
  </body>
</html>

```

Maak nu een nieuw bestand aan geef deze de naam `opdracht8.css`. Typ de volgende code in die pagina:

```

body {
  font-family: Verdana;
  background-color: lightblue
}
h2 {
  color: red;
  font-style: italic
}
p {
  font-variant: small-caps
}

```

Je kunt nu de pagina uittesten door `opdracht8.html` in jouw browser te bekijken.

In het HTML bestand staat niet zoveel nieuws behalve de regel:

`<link rel="stylesheet" type="text/css" href="opdracht8.css">`. In die regel wordt een relatie gelegd met het bestand `opdracht8.css`. In de tekst zelf wordt de code van het CSS bestand uitgelegd. Probeer iedere regel te begrijpen!

Heb je gezien dat we in de tekst de code `<h2>` gebruiken? Waarom zou dit nodig zijn? Waarom konden we niet gewoon `<h2>` typen?

Lettertypen (fonts)

We hebben al gezien dat het aanpassen van een lettertype niet zo moeilijk is. Met `font-family` kun je het lettertype aangeven. Maar niet ieder lettertype is op elke computer aanwezig. Vandaar dat men meestal een reeks van fonts aangeeft. De browser kan dan kijken of het lettertype op de computer voorkomt en als dat niet het geval is, het volgende lettertype uit het rijtje kiezen.

Er zijn wel fonts die op vrijwel iedere computer aanwezig zijn, de zogenaamde generieke fonts. Dat zijn deze vijf:

- serif
- sans-serif
- cursive
- fantasy
- monospace

Als je een rijtje mogelijke fonts opgeeft voor jouw website, plaats dan altijd een generiek font als achterste. Bijvoorbeeld zo:

```
font-family: Arial, Courier, Verdana, sans-serif;
```

Let op, als er spaties in de naam van het font staan, moet je dat font tussen aanhalingstekens zetten. Zoals:

```
font-family: Arial, "Courier New", Verdana, sans-serif;
```

Opdracht 9

Zoek op internet de betekenis van de volgende font-eigenschappen:

- font-variant
- font-size
- font-style
- font-weight
- line-height

Je weet nu:

- Hoe je een CSS bestand maakt
- Hoe je een CSS bestand koppelt aan jouw HTML bestand
- Wat selectors zijn in een CSS bestand
- Hoe je waarden toe kent aan de eigenschappen van selectors
- Hoe je lettertypen aangeeft in een CSS bestand.

3.2 Links: kleuren, decoratie en achtergrond

In hoofdstuk 4 bekijken we de diverse soorten links maar met CSS kunnen we ook het uiterlijk van een link kunnen beïnvloeden. Hier alvast wat mogelijkheden:

- **a:link** (normale link)
- **a:visited** (bezochte link)
- **a:hover** (de muis gaat over de link)
- **a:active** (als je op de link klikt)

Wil je je link normaal in het groen hebben, bezocht in het rood, ga je er met de muis over in het blauw en als je op de link klikt dan zwart, dan neem je de volgende regels op in jouw CSS deel:

```
a:link { color: green }  
a:visited { color: red }  
a:hover { color: blue }  
a:active { color: black }
```

Je merkt wel, als je zulke korte regels hebt is het handiger de accolades op dezelfde regel te zetten en de eigenschap met de waarde er tussen. Nog een paar voorbeelden:

Voeg de eigenschap text-decoration toe en zet zijn waarde op none en je krijgt er geen streep onder de link:

```
a:link { color: red; text-decoration: none }
```

Voeg de eigenschap background-color toe en zet zijn waarde op een kleur, bijvoorbeeld op black en je krijgt een link met een gekleurde achtergrond.

```
a:link { color: red; background-color: black }
```

Opdracht 10

Maak een CSS bestand, geef dat de naam `opdracht10.css` en zorg voor de volgende stijlen:

- De body heeft lettertype Arial met als tweede keuze Verdana en als generiek font sans-serif
- De achtergrond van de body wordt zwart
- De tekst die gemarkeerd is met `<h2>` krijgt de kleur geel
- De letters in een paragraaf krijgen de kleur wit
- Een link heeft een zalmkleur. Een reeds bezochte link heeft de kleur lichtblauw. Ga je met de muis over de link van krijgt hij de kleur geel en als je op de link klikt wordt de kleur wit.

Maak nu een HTML pagina, geeft dat de naam `opdracht10.html` en plaats daar de volgende code in:

```
<!DOCTYPE html>
<html lang="nl">
  <head>
    <meta charset="utf-8">
    <!-- hier mag je zelf even nadenken hoe je de link maakt
         met het bestand opdracht10.css -->
    <title>Een pagina met stijl</title>
  </head>
  <body>
    <h2>Vruchtbrengend woordenboek</h2>
    <hr>
    <p>
      Dit project documenteert uit andere talen afkomstige
      woorden die in het Nederlands worden gebruikt, met daarbij
      Nederlandse alternatieven, bijvoorbeeld Backup -
      reservekopie.
      Het gaat daarbij niet alleen om het documenteren van
      bestaande alternatieven, maar uitdrukkelijk ook om het
      bedenken van nieuwe.
    </p>
    <p>
      In de meeste Europese landen stelt men een enorme
      toevloed aan vreemde woorden vast, die bijna uitsluitend
      afkomstig zijn uit de Verenigde Staten. Terwijl in het
      verleden het aantal vreemde woorden nog te overzien was,
      is hun aantal op exponentiële wijze toegenomen tijdens
      de laatste decennia.
    </p>
    <p>
      Meer hierover via <a href="http://nl.wikibooks.org/wiki/
      Vruchtbrengend_woordenboek" target=_blank>wikipedia.</a>
    </p>
  </body>
</html>
```

Test je pagina, test vooral ook de links en zie hoe ze van kleur veranderen. Werkt het niet? Heb je de link naar het CSS bestand wel juist aangegeven?

Meer over links in hoofdstuk 4.

Inmiddels weten we het e.e.a. van CSS. De vraag is nu hoe we dat kunnen toepassen op de eerder gemaakte oefening met die appels, peren en bananen. Daar gaan we nu mee aan de slag.

Opdracht 11-1

Open het bestand `opdracht7.html` en wijzig de regels van het navigatie deel (het gedeelte tussen de `nav`-tags) als volgt:

```
<nav id="menu">
  <ul>
    <li><a href="#">Appels</a></li>
    <li><a href="#">Peren</a></li>
    <li><a href="#">Bananen</a></li>
  </ul>
</nav>
```

Sla de pagina op onder de naam `opdracht11.html` en bekijk de pagina in je browser.

anchor URL

Met de `<a>` tag maak je een link. Met `href` geef je aan waar hij naar toe moet linken. In dit geval wordt er gelinkt naar `"#"`. Dit noemen we een anchor URL oftewel een link naar een plek op dezelfde pagina. Zo zien we wel anchor URL's als `"#top"`. Ergens anders op de webpagina staat dan ``. Klik je op zo'n link van een anchor URL dan springt de pagina naar de plek waar de `` code staat.

In dit voorbeeld is het een lege verwijzing omdat er na het hekje niets staat. We kunnen daar later de juiste URL invullen maar op deze manier zal de browser dit al zien als een link en daar gaat het hier ook om.

Een URL is de aanduiding voor een link. De afkorting staat voor **Uniform Resource Locator**. Het kan het adres zijn van een website maar ook van een andere webpagina van de site en zoals hiernaast aangegeven, een plaats op de huidige pagina.

Nu we van de elementen in de lijst links hebben gemaakt kunnen we er opmaak elementen aan toe kennen. De opmaak regelen we in een apart bestand met de extensie `.css` en in de webpagina vragen we dat opmaakbestand op.

Opdracht 11-2

Allereerst moeten we er voor zorgen dat de `css` code die we straks aan gaan maken in het bestand `opmaak.css` in onze webpagina wordt ingelezen.

Open `opdracht11.html` en voeg in de `head` sectie de link naar `opdracht11.css` toe.

```
<head>
  <meta charset="utf-8" />
  <!-- Voeg hier de link naar opdracht11.css toe-->
  <title>De groentenman</title>
</head>
```

Zoals we ook al eerder gedaan hebben maak je met de `<link>` tag een koppeling tussen het HTML-bestand en het CSS-bestand. Je kunt op die manier meerdere stylesheets aan het HTML-bestand koppelen maar wij doen het voorlopig wel met één.

Maak in jouw editor een nieuw bestand en sla dat op onder de naam `opdracht11.css`.

In dat bestand komt de volgende tekst:

```
* {
  margin: 0;
  padding: 0;
}
```


Bekijk nu het bestand opdracht11.html in jouw browser. Zie je verschil met de vorige keren dat je opdracht11.html bekeek?

Laten we even naar de CSS code kijken. Met de ***** wordt bedoeld dat alle code die daarna tussen de accolades staat van toepassing is op alle elementen. Dit sterretje noemen we wel de universele selector.

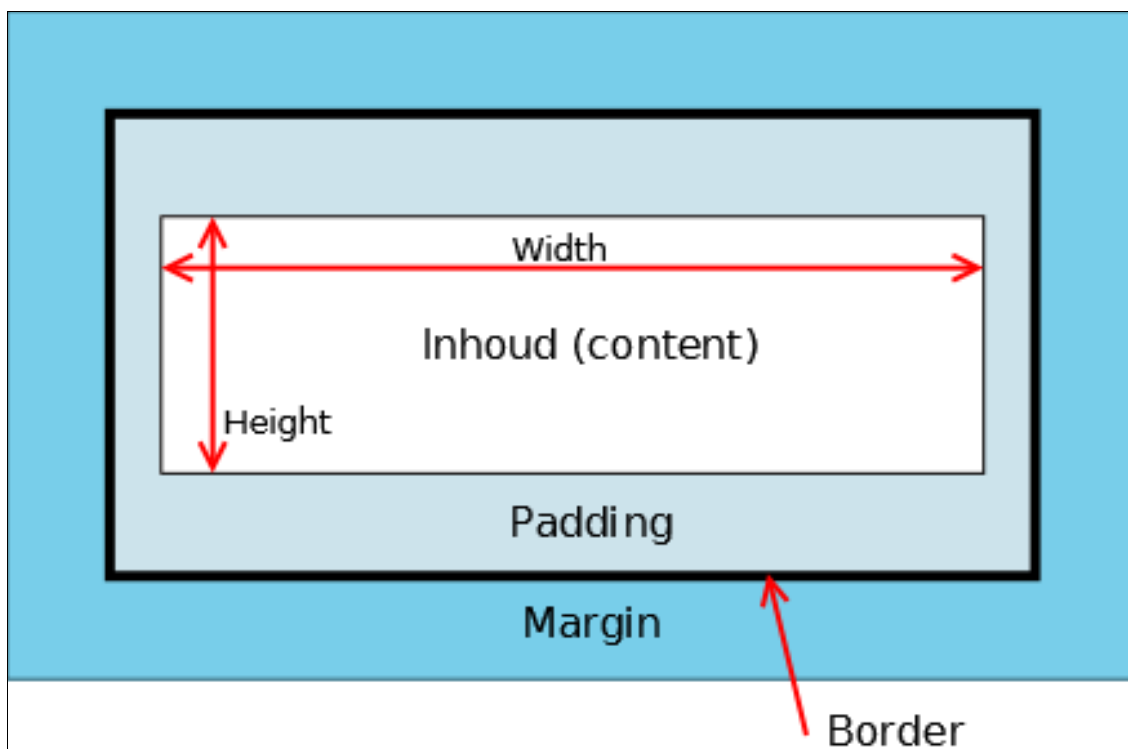
**universele
selector**

Kortom, met de code hier zetten we voor alle elementen de **margin** en de **padding** op 0. Maar om te begrijpen wat hiermee wordt bedoeld, eerst enige uitleg over het CSS box model.

CSS box model

3.3 Het CSS box model

Op alle blok-elementen (zie pagina 9) van een HTML pagina is het box model van toepassing. Een blok-element neemt altijd een rechthoekige ruimte in op de webpagina. En van die rechthoekige ruimte kunnen we een aantal eigenschappen definiëren. **margin** en **padding** zijn twee van die eigenschappen. Later zullen we nog meer gebruiken.



Afbeelding 9: Het CSS box model

Afbeelding 9 laat het CSS box model zien. Naast de breedte (**width**) en de hoogte (**height**) van de eigenlijke inhoud (content) kun je dus van een box de rand (**border**) vastleggen. Daarna geef je met **padding** de ruimte aan tussen de content en de rand en met **margin** geef je de ruimte aan tussen de rand en de overigen delen van de webpagina. Verderop meer over de positie van een dergelijk element.

Maar waarom werden ze hier op 0 gezet? De verschillende browsers kennen allen verschillende beginwaarden o.a. wat betreft de grootte van de marges en de padding-ruimte. Om er nu voor te zorgen dat voor alle browsers waar jouw website in moet werken de beginwaarden gelijk zijn, maakt men vaak gebruik van een zogenaamde reset css code. Wij hebben niet zo'n uitgebreide code nodig maar op het internet zijn diverse reset stylesheets te vinden. Bijvoorbeeld hier: <http://html5doctor.com/html-5-reset-stylesheet/>

Als het goed gegaan is heb je dus bij het bekijken van jouw opdracht gezien dat er geen ruimte meer tussen de tekst en de zijkant van de webpagina zit.

Laten we nog wat CSS code toevoegen:

Opdracht 11-3

Voeg de volgende code toe aan het bestand `opdracht11.css`. (De grijze code staat er al.)

```
* {
    margin: 0;
    padding: 0;
}

header, nav, section {
    display: block;
}

/*
display: block betekent dat een element afgebeeld wordt als een
blok. Daarmee zorg je er voor dat deze elementen witruimte boven
en onder zich hebben. Verder zal een blok geen ander HTML ele-
ment naast zich willen hebben tenzij dat andere element met een
zogenaamde float opdracht juist de opdracht krijgt zich er wel
naast te nestelen. In deze CSS opdracht leg je dus vast dat hea-
der, nav en section de eigenschappen van een blok meekrijgen.
*/

body {
    margin-top: 0;
    margin-right: auto;
    margin-bottom: auto;
    margin-left: auto;
    width: 940px;
    font-size: 13px;
    line-height: 22px;
    font-family: Helvetica, Arial, sans-serif;
    background: #f0f0f0;
}

/*
Je ziet hier dat de body een blok-element is waarvan je de mar-
gin in kunt stellen. Bij een margin waar auto achterstaat wordt
dit aan de browser overgelaten en dat betekent dat de body, hier
940 pixels breed (width), gecentreerd wordt als het browser-
scherm groter is dan die 940 pixels.
*/

h1 {
    font-size: 28px;
    line-height: 44px;
    padding-top: 22px;
    padding-right: 0px;
    padding-bottom: 22px;
    padding-left: 0px;
}
```

```
p {
  padding-bottom: 22px;
}

/*
h1 en p zijn beide blok-elementen en dus kun je o.a. de padding-
waarde van het blok instellen.
*/
```

Opdracht 11-4

- Ga op het internet op zoek naar de betekenis van: **background: #f0f0f0;**
- #f0f0f0** is een waarde in een bepaald talstelsel. Welk?

In de code is hier en daar tekst toegevoegd die begint met `/*` en eindigt met `*/`. Op die manier kun je opmerkingen ter verduidelijking van de programmacode toevoegen. Deze opmerkingen zijn niet zichtbaar op jouw website.

commentaar

Heb je opdracht11.html met de vernieuwde opdracht11.css al bekeken? Als het goed is zal jouw pagina er ongeveer zo uit zien:



Afbeelding 10: De oefening na de wijzigingen van opdracht 11-3

Nog niet echt spectaculair. Maar we zijn er ook nog niet. We gaan nu de links, die we in een nav blok hebben gevat, aanpakken.

3.4 Nav in CSS

Het belangrijkste deel van onze groentenman website is het navigatiedeel. Zoals aan ieder blok-element kunnen we zowel aan het blok zelf als aan de elementen in dat blok opmaak codes toe-kennen. Hier de zeer uitgebreide CSS opmaakcodes voor het **nav** blok.

Opdracht 11-5

Voeg de volgende code toe aan opdracht11.css

```
p {
  padding-bottom: 22px;
}

nav {
  position: absolute;
  /*zet de positie van het nav-blok op een vaste plek*/
```

```

left: 0;
/*het blok begint links aan de linkerkant van de body*/
width: 100%;
/*het navigatieblok neemt de gehele breedte in van de
browser*/
background: #808080;
}

nav ul {
/*hiermee geeft je aan dat het om de unordered list van het
nav blok gaat*/
margin-top: 0;
margin-right: auto;
margin-bottom: auto;
margin-left: auto;
width: 940px;
/*de eigenschappen van de ul-blok*/
list-style: none;
/*hiermee verdwijnen bij de unordered list de bolletjes*/
}

nav ul li {
/*hiermee geeft je aan dat het om de list elementen van de
unordered list van het nav blok gaat*/
float: left;
/*de lijst gaat hierdoor van links naar rechts i.p.v. van
boven naar beneden*/
}

nav ul li a {
/*hiermee geeft je aan dat het om de a-elementen van de
list-elementen van de unordered list van het nav-blok gaat*/
display: block;
/*de a-elementen worden hier gedefinieerd als een blok en dus
kunnen we er blokeigenschappen aan toekennen*/
margin-right: 20px;
width: 140px;
font-size: 14px;
line-height: 44px;
text-align: center;
text-decoration: none;
color: #000000;
}

nav ul li a:hover {
/*:hover is een zogenaamde pseudo-klasse (pseudo-class).
pseudo-klassen worden gebruikt om speciale effecten
aan een selector toe te voegen. De :hover pseudo-klasse
reageert als je met de muis over dat element gaat*/
color: #ffffff;
}

nav ul li.selected a {
/*hier gaat het om de geselecteerde link*/
color: #ffffff;
}

```

Test de code opnieuw uit door `opdracht11.html` te starten. Begint het al te lijken op afbeelding 5? Als het goed is wel. Maar we zijn ineens de onderste sectie kwijt geraakt. De oorzaak hiervan zit hem in het feit dat we het navigatieblok met **absolute** op z'n plek hebben gezet. De andere blokken hebben een zogenaamde relatieve positie, een positie ten opzichte van elkaar. Maar een blok met een absolute positie kan als het ware overal overheen geplaatst worden. En zo is het navigatieblok over de tekst geplaatst. Je kunt dat controleren door de achtergrondkleur van het navigatieblok even weg te halen, dan zie je hem wel weer.

Maar hoe maken we dat blok nu weer zichtbaar? Dat doen we in de volgende opdracht:

Opdracht 11-6

Plaats onderaan `opdracht11.css` de volgende code:

```
nav ul li.selected a {
  /*hier gaat het om de geselecteerde link*/
  color: #ffffff;
}
#inhoud {
  position: relative;
  margin-top: 66px;
  padding: 44px;
}
```

Bij het gedeelte dat we net kwijt waren hadden we in de code een **id** vastgelegd met de naam **inhoud**. Je ziet hierboven hoe je in jouw CSS code specifiek aan die sectie bepaalde opmaakeigenschappen kunt toekennen.

Ziet jouw website er nu ongeveer net zo uit als in afbeelding 5? Dan is het goed gegaan.

Probeer goed te begrijpen hoe je dus van elk blok-element met alle elementen die daar weer binnenin zitten de opmaak kunt bepalen.

Bewaar de code van opdracht 11 goed. In hoofdstuk 8 gaan we hier verder mee aan de slag!

4 Links

Eén van de belangrijkste eigenschappen van HTML is de mogelijkheid naar andere pagina's te kunnen linken. Onder een link verstaan we een aan te klikken stukje tekst of plaatje waarmee je naar een andere pagina of deel van een pagina springt.

We onderscheiden diverse soorten links:

- Interne link
- Externe link
- Anchor URL
- Link naar een afbeelding
- Link naar JavaScript functie (zie hoofdstuk 7)

4.1 Interne link

Wanneer je een website bouwt die bestaat uit meerdere pagina's zet je al die pagina's op een webserver. Om dan van de ene pagina naar de andere te kunnen springen gebruik je een interne link. Een interne link is dus een link die een andere pagina of een afbeelding opent die op dezelfde webserver staat. In de meest eenvoudige vorm ziet een interne link er als volgt uit:

```
<a href="pagina1.html">Pagina 1</a>
```

Er komt dan **Pagina 1** op jouw pagina te staan en als je daarop klikt wordt het bestand `pagina1.html` geopend. Dit bestand staat in dezelfde map op de webserver als waar ook de vorige pagina staat.

4.2 Externe link

Daarmee zal meteen ook wel duidelijk worden wat een externe link is. Daarmee kun je naar een pagina springen die ergens anders op internet staat. Het commando daarvoor is dus vrijwel hetzelfde als voor de interne link alleen moet de naam van het bestand voorafgegaan worden door de plaats waar de pagina staat. Bijvoorbeeld:

```
<a href="http://htmlhulp.com/nl/">html-hulp</a>
```

Bij een link op deze manier zal jouw huidige pagina verdwijnen en de nieuwe pagina daarvoor in de plaats komen. Wil je echter dat de pagina in een nieuw venster verschijnt voeg dan het attribuut **target** als volgt toe:

```
<a href="http://htmlhulp.com/nl/" target="_blank">html-hulp</a>
```

Eigenlijk wordt hier helemaal niet gelinkt naar een andere pagina maar naar een complete website. In feite betekent dat dat je linkt naar de hoofdpagina van die website. Meestal is dat een pagina met de naam `index.html` of `index.php`.

4.3 Anchor URL

Het derde type link is een link naar een plaats op dezelfde pagina. Als je op je pagina een link maakt naar een ander gedeelte op die pagina, geef je de referentie van de link een naam met een `#` ervoor. Dat gaat als volgt:

```
<a href="#deel1">Naar deel 1</a>
```

```
<a href="#deel2">Naar deel 2</a>
```

Op een andere plaats in de pagina komen dan de plaatsen waar je naar wilt linken. Die plaatsen worden herkend door een id. Dat kan er dan als volgt uit zien:

```
<p id="deel1">Hier komt de inhoud van deel 1</p>
```

En op dezelfde manier maak je ook de link naar deel 2. Let op: de naam die je achter `#` en met `id` gebruikt moet met een letter beginnen. De naam is bovendien hoofdletter-gevoelig, dus `"#deel1"` is een andere referentie dan `"#Deel1"`.

4.4 Afbeelding

De link naar een afbeelding is eigenlijk van een andere soort. Bij een afbeelding gaat het er om op de plaats van de link een plaatje in jouw pagina in te voegen. Veronderstel dat je een plaatje hebt met de naam `appels.jpg` die je op een bepaalde plaats in jouw website zichtbaar wilt maken. Je zet dan op die plaats de volgende code:

```

```

Aan die code kun je dan nog wat extra attributen toevoegen als je bijvoorbeeld de breedte (`width`) of de hoogte (`height`) wilt beïnvloeden. Wil je bijvoorbeeld dat het plaatje een breedte heeft van 300 pixels dan wordt jouw code:

```

```

Opdracht 12

- Maak drie html pagina's met de namen `appels.html`, `peren.html` en `bananen.html`
- Ga op internet op zoek naar afbeeldingen van appels, peren en bananen en plaats die afbeeldingen in dezelfde map waar ook jouw HTML bestanden staan
- Plaats op de drie pagina's de code waarmee de betreffende plaatjes worden opgeroepen. Stel eventueel de breedte of de hoogte bij zodat de plaatjes er op alle pagina's ongeveer even groot uit zien
- Roep `opdracht11.html` op, wijzig de links daarin dusdanig dat bij het aanklikken van de links de andere pagina's verschijnen en sla `opdracht11.html` op met de nieuwe naam `opdracht12.html`
- Maak op de drie nieuwe pagina's een link die de naam `home` krijgt waardoor je terugkomt op de beginpagina.
- Voeg tekst over de fruitsoorten toe aan de nieuwe pagina's en zorg voor een opmaak die voor alle pagina's gelijk is.
- Bij een afbeeldingslink kun je ook het attribuut `alt` gebruiken. Bijvoorbeeld:

```

```

Zoek op wat de bedoeling is van **alt** en wanneer de tekst zichtbaar wordt.

- h Zoek ook op wat je met de attributen **align** en **border** kunt doen.

5 Tabellen

Vaak moeten er gegevens op een website komen in een bepaalde structuur. Bijvoorbeeld getallen keurig in kolommen of koppen ervan keurig in een rij. Je kunt dat realiseren met een tabel. Een tabel is niet zo heel moeilijk. Je moet hem alleen heel systematisch opbouwen. En denk om het inspringen anders verdwaal je al gauw!

Een tabel begint met `<table>` en eindigt dus met `</table>`. Daar tussenin komen de rijen. Iedere rij leggen we vast met `<tr>` en `</tr>` (tablerow). Als je drie rijen wilt maken krijg je dus:

```
<table>
  <tr>
    ... hier komt de eerste rij ...
  </tr>
  <tr>
    ... hier komt de tweede rij ...
  </tr>
  <tr>
    ... hier komt de derde rij ...
  </tr>
</table>
```

In een rij komen de gegevens, de data. Daarvoor gebruiken we de tags `<td>` en `</td>` (table-data). De gegevens komen onder elkaar te staan in de vorm van kolommen. In de volgende oefening maken we vier kolommen.

Opdracht 13

Maak een HTML-pagina met de volgende code:

```
<!DOCTYPE html>
<html lang="nl">
  <head>
    <meta charset="utf-8" />
    <title>Tabellen</title>
  </head>
  <body>
    <table border=4>
      <tr>
        <td>Kolom 1</td>
        <td>Kolom 2</td>
        <td>Kolom 3</td>
```

```

        <td>Kolom 4</td>
    </tr>
    <tr>
        <td>getal 1</td>
        <td>getal 2</td>
        <td>getal 3</td>
        <td>getal 4</td>
    </tr>
    <tr>
        <td>getal 5</td>
        <td>getal 6</td>
        <td>getal 7</td>
        <td>getal 8</td>
    </tr>
</table>
</body>
</html>

```

Sla de pagina op onder de naam opdracht13.html en bekijk de pagina in je browser.

Ziet het er ongeveer zo uit?

Kolom 1	Kolom 2	Kolom 3	Kolom 4
getal 1	getal 2	getal 3	getal 4
getal 5	getal 6	getal 7	getal 8

Afbeelding 11: Opdracht 10, een tabel

Als je heel goed kijkt klopt er iets niet. In het plaatje hiernaast heeft iedere kolom een vet lettertype in de kop. En dat heb jij niet. Je kunt jouw koppen ook vet maken door de `<td>` en `</td>` tags van de eerste rij te vervangen door `<th>` en `</th>`. De h in die tag staat voor header en zorgt ervoor dat de koppen vetgedrukt worden.

Datzelfde kun je toen met een kolom. Door in een rij het iedere eerste paar `<td>` en `</td>` tags te vervangen door `<th>` en `</th>` tags zal de eerste kolom vetgedrukte waarden krijgen.

Opdracht 14

Die laatste zin klinkt ingewikkeld. Probeer toch eens de volgende tabel te maken en let op welke cellen een vetgedrukte waarde moeten krijgen.

	Amsterdam	Rotterdam	Den Haag
Aantal inwoners	761.395	584.856	485.818
Aantal nationaliteiten	177	167	150

Afbeelding 12: Vetgedrukte koppen

Opdracht 15

De getallen in de cellen zijn links uitgelijnd. Dat wil zeggen dat ze, net als normale tekst, aan de linkerkant van zo'n cel beginnen. Maar getallen zien we het liefst rechts uitgelijnd. Wijzig bij de cellen waar getallen in staan de tag `<td>` in `<td align=right>` en bekijk het resultaat.

Opdracht 16

Verander de tag <table> in:

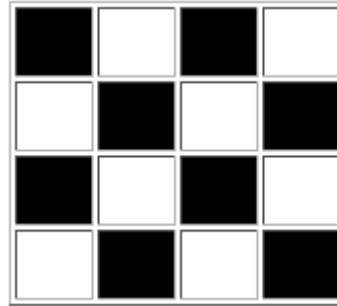
```
<table border=8 title="Inwoners" bgcolor="lightblue" width=500>
```

Vertel van elk attribuut wat het effect is.

Opdracht 17

De attributen zoals je die bij opdracht 13 gebruikt hebt kun je meestal ook op een enkele cel toepassen.

Maak met de table functie het plaatje als hiernaast. Neem voor de rijhoogte een hoogte van 50 met `<tr height=50>` en voor de kolombreedte een breedte van 50 met `<td width=50>`. Uiteraard moet je de juiste achtergrondkleur kiezen.



Afbeelding 13: zwart-wit



Voor we verder gaan met JavaScript moeten we eerst een duidelijker beeld krijgen van wat programmeren is en waar het uit bestaat. Zo komen we zaken tegen als variabelen, beslissing-structuren en sequenties. Genoeg dus om even het toetsenbord los te laten en ons te verdiepen in allerlei zaken die bij programmeren komen kijken.

6.1 De programmeertaal

Programmeren doe je met behulp van een programmeertaal. De programmeertaal bestaat uit (meestal Engelstalige) instructies die, vaak via een onderliggend programma, worden omgezet in voor de computer begrijpbare instructies. Zo'n onderliggend programma kan bijvoorbeeld jouw browser zijn. Deze “vertaalt” de HTML-code naar “iets op het scherm”.

Er zijn heel veel programmeertalen. Wanneer we aan websites gaan werken gebruiken we vooral HTML. Eigenlijk kun je HTML niet echt een programmeertaal noemen maar samen met JavaScript wordt het dat wel. Dan ga je namelijk gebruik maken van controlestructuren die je in heel veel programmeertalen tegenkomt. Deze controlestructuren zijn:

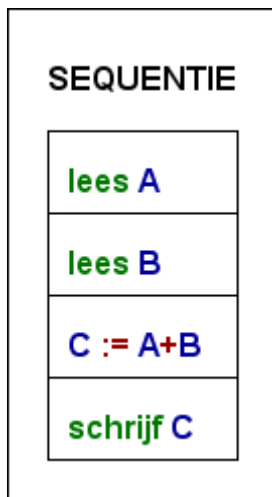
- sequentie of opeenvolging
- keuze of selectie
- herhaal zolang
- herhaal totdat

Een controlestructuur geeft aan in welke volgorde instructies in een programma worden uitgevoerd. Je kunt ze ook nog eens in elkaar gebruiken waarbij we spreken van geneste structuren. De controlestructuren kunnen we heel goed in kaart brengen met programmastructuur diagrammen (PSD's). Deze diagrammen worden ook wel Nassi Shneiderman diagrammen genoemd naar Isaac Nassi en Ben Shneiderman, twee heren die deze structuurdiagrammen in 1972 hebben ontworpen.

6.2 Programmastructuur diagrammen

Een programmastructuur diagram (PSD) is een schema waarmee je een programmeerprobleem kunt tekenen. Je laat er de structuur van een probleem mee zien. Het is geen programmeertaal maar bevat wel juist die elementen die je ook bij iedere programmeertaal tegenkomt.

In de afbeeldingen 14 t/m 17 zie je de schematische voorstelling van de belangrijkste structuren.



Afbeelding 14: Sequentie of opeenvolging

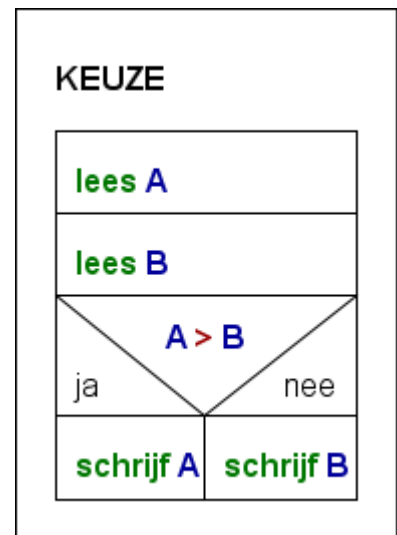
Sequentie of opeenvolging (zie afbeelding 14): de instructies worden achter elkaar uitgevoerd.

Keuze of selectie (zie afbeelding 15): hiermee wordt een keuze tussen alternatieven gemaakt

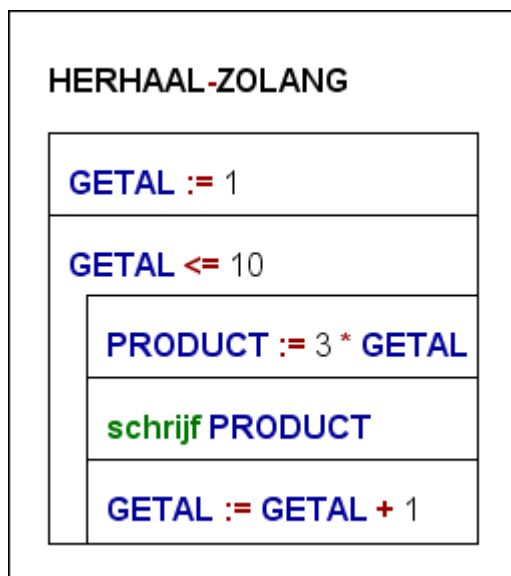
Herhaling of iteratie. Een groep instructies wordt een aantal malen herhaald. We kunnen deze onderscheiden in herhaling zolang (zie afbeelding 16) en herhaal totdat (zie afbeelding 17)

Bij herhaal zolang wordt eerst de voorwaarde geanalyseerd. Als aan de voorwaarde is voldaan wordt de opdracht of de sequentie van opdrachten uitgevoerd en begint een nieuwe herhaling. Is er niet aan de voorwaarden voldaan dan stopt de herhaling.

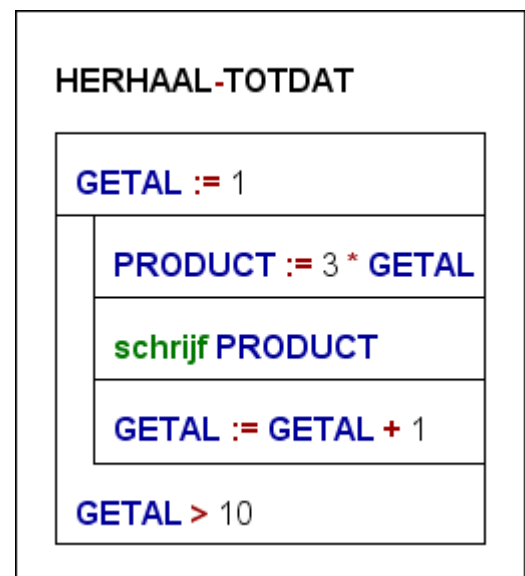
Bij herhaal totdat wordt eerst de instructie uitgevoerd en pas daarna wordt gecontroleerd of er aan de voorwaarde voldaan is. Is er niet aan de voorwaarden voldaan dan stopt de herhaling.



Afbeelding 15: Keuze of selectie



Afbeelding 16: Herhaal zolang



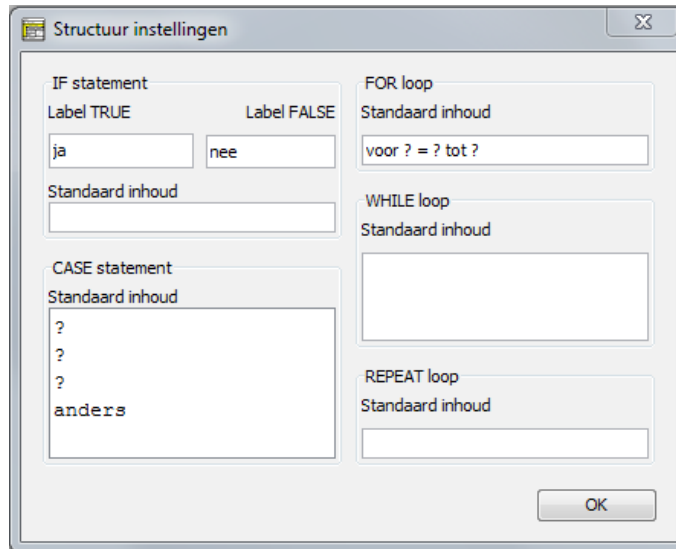
Afbeelding 17: Herhaal totdat

Deze structuur-elementen kun je op allerlei manieren samenvoegen. Om het onszelf een beetje gemakkelijker te maken gaan we ze tekenen met een computerprogramma. Maar daar moeten we eerst even mee leren werken.

6.3 Werken met Structorizer

Als je zelf PSD's gaat tekenen is het handig daarvoor een programma te gebruiken. Een goede is Structorizer die je vindt op <http://structorizer.fisch.lu/>. Als je daar kiest voor Structorizer Java Web Start hoeft je helemaal niets te installeren (zolang je maar een internet toegang hebt). In deze paragraaf gaan we aan de slag met Structorizer en gaan we dat programma instellingen meegeven die wij nodig hebben.

Als je Structorizer Java Web Start aan klikt dan verschijnt het lege scherm van Structorizer. Misschien heb je eerst nog even aan moeten geven waarmee het programma geopend moet worden. Klik daarbij gewoon op OK.



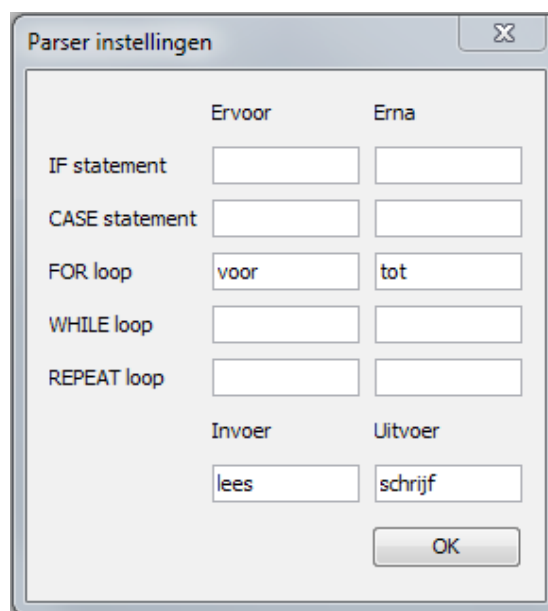
Afbeelding 18: De structuurinstellingen van Structorizer

Laten we eerst de taal instellen. Vooropgesteld dat hij bij jou nog niet op Nederlands staat. Klik daarvoor op **Preferences** en daarna **Language** en kies voor **Dutch**.

De volgende stap is het instellen van de structuren. Dat doe je via de keuzes: **Instellingen** en dan **Structuren**. Vul het scherm dat je dan krijgt in zoals in afbeelding 18. Probeer de woorden die je hier ingevuld hebt te onthouden want deze woorden gaan we straks ook in de structuurdiagrammen gebruiken. Op enkele plaatsen staan de haakjes (). Verwar die niet met een 0!

Het volgende dat we gaan doen is de parser instellen. De parser zet de code die wij in Structorizer typen (zoals we dat net hebben vastgelegd) om naar code van de programmeertaal.

Klik weer op **Instellingen** en kies dan voor **Parser**. Vul nu het scherm in zoals in afbeelding 19. Nu we deze instellingen klaar hebben kunnen we aan de slag met Structorizer.



Afbeelding 19: De parser

Opdracht 18

Maak in Structorizer de PSD's zoals die in de afbeeldingen 14 t/m 17 staan afgebeeld. Het zal best wel even zoeken zijn voor je het juiste element hebt gevonden. Via de keuze Diagram, Toevoegen enz. kun jij het juiste element selecteren. Maar uiteraard kan dat ook via één van de knoppen bovenin Structorizer. Als je een PSD klaar hebt, klik dan op



Op die manier kun je hem testen. Zorg ervoor dat hij werkt!

Zoals je gezien hebt kun je de structuur-elementen kun je op allerlei manieren samenvoegen. Maar de volgorde van het uitvoeren van de opdrachten is steeds van boven naar beneden. In afbeelding 20 zie je een voorbeeld van een eenvoudig PSD. Er wordt hier alleen maar gebruik gemaakt van het opdrachtblok. Zo'n serie elementen noemt men een sequentie.

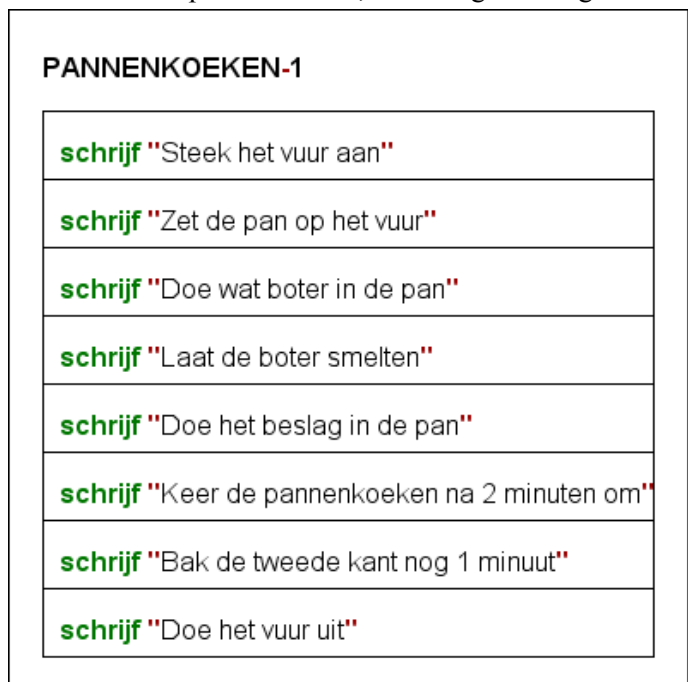
Als we naar zo'n probleem kijken als het bakken van pannenkoeken, komt al gauw de gedachte bij je op dat 2 minuten voor kant één en 1 minuut voor kant 2 misschien wel wat rigide is. Beter zou



Afbeelding 21: Een PSD met herhaal-zolang blokken

Bij het uitwerken van een ingewikkeld probleem kun je het beste de stappen, die moeten gebeuren, eerst globaal beschrijven. Dus eerst schrijf je in grote lijnen de volgorde op waarin de dingen moeten gebeuren. Dit doe je in gewone Nederlandse woorden.

In feite worden dat de opdrachten. Als je die nu in rechthoekige blokken zet ontstaat als vanzelf al een sequentie.



Afbeelding 20: Een sequentie van elementen

zijn gewoon te kijken of de pannenkoek al bruin is. Hier hebben we dus te maken met een voorwaarde. Daarvoor kunnen we een herhaal zolang blok of een herhaal totdat blok gebruiken. De herhaling zit in het steeds weer bekijken of een kant al bruin genoeg is. Maar als we een voorwaarde willen gebruiken, dat hebben we ook een variabele nodig. Want in de voorwaarde kunnen we zien of de variabele wel of niet een bepaalde waarde heeft.

In afbeelding 21 zie je het gebruik van de variabele **BRUIN**. De beide herhalingen gaan steeds net zolang door totdat tijdens zo'n herhaling "ja" ingetypt wordt.

Opdracht 19

Maak het programmastructuur diagram zoals in afbeelding 21 maar deze keer met herhaal-totdat blokken.

Bij opdracht 19 had je, dankzij afbeelding 21, al een goed voorbeeld van de aanpak van het probleem bij de hand maar in de meeste gevallen is dat natuurlijk niet zo. Hoe doe je dat dan?

Daarna ga je de onderdelen verfijnen. We verfijnen het probleem door het steeds op te splitsen in deelproblemen. Die deelproblemen verdelen we ook weer in stappen en we gaan daar zover als maar kan mee door. We noemen dit stapsgewijze verfijning en je bent dan bezig met top-down-programmeren.

Bij een herhaling is er dus sprake van een voorwaarde. De herhaling gaat door tot aan de voorwaarde is voldaan of de herhaling gaat door zolang nog niet aan de voorwaarde is voldaan. In het Nederlands zouden we kunnen zeggen: Doe die opdracht zolang aan de voorwaarde wordt voldaan. In een programmeertaal zoals JavaScript ziet *herhaal zolang* er als volgt uit:

```
while ( voorwaarde ) {
  opdracht;
}
```

while – loop

De opdracht wordt hier alleen maar uitgevoerd als er aan de voorwaarde voldaan wordt. We noemen dit ook wel een while-loop. Maar je kunt de voorwaarde ook achteraan zetten. Daarmee geef je aan dat de opdracht tenminste één keer wordt uitgevoerd en daarna totdat niet meer aan de voorwaarde wordt voldaan. In JavaScript ziet *herhaal totdat* er als volgt uit:

```
do {
  opdracht;
}
while ( voorwaarde );
```

do – loop

We noemen dit ook wel een do-loop.

Binnen heel veel programmeertalen kun je herhalingen nog verder inperken. Bijvoorbeeld door er voor te zorgen dat een herhaling een vooraf vastgesteld aantal keren wordt uitgevoerd. Maar in al die gevallen kun je het ook met de hier genoemde structuren doen.

Vaak is het zo dat de opdrachten afhangen van een bepaalde voorwaarde of keuzes. Is de uitkomst van de voorwaarde Ja of Nee. Is de bewering Waar of Niet Waar. Je kunt het dan vaak in het

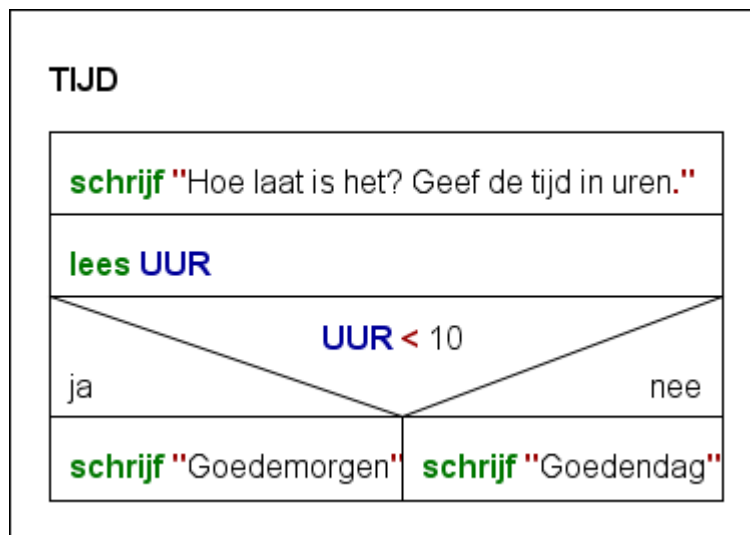
Nederlands als volgt zeggen: Als er aan die voorwaarde voldaan wordt dan moet er dit gebeuren maar anders moet er dat gebeuren.

In het PSD van afbeelding 22 wordt de tijd gevraagd. Op basis van die uitkomst wordt een keuze gemaakt tussen de beide welkomstgroeten.

In vrijwel iedere programmeertaal komt het keuzeblok voor in een vorm die lijkt op:

```
if ( voorwaarde ) then {
  opdracht;
}
else {
  opdracht;
}
```

if – else blok



Afbeelding 22: Het keuzeblok

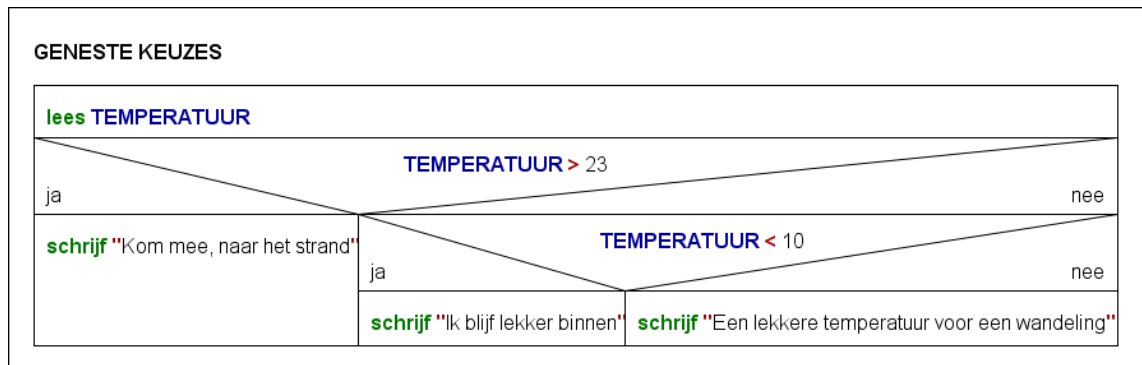
Per programmeertaal verschilt de manier van schrijven (de syntax) wel (in JavaScript komt het woordje **then** er niet tussen) maar het principe komt steeds op hetzelfde neer. Het zal duidelijk zijn dat het hier om het keuzeblok gaat.

Opdracht 20

Maak een PSD waarin je de structuur vastlegt om de tafel van 7 af te drukken.

Opdracht 21

Zoals al eerder is verteld kun je PSD elementen ook binnen andere PSD elementen gebruiken. Bijvoorbeeld een tweede keuzeblok in het geval de keuze Nee wordt gekozen. Kijk maar naar afbeelding 23.



Afbeelding 23: Een keuzeblok binnen een keuzeblok

Pas nu het PSD van afbeelding 22 dusdanig aan dat er, als het na 18 uur is, de boodschap “goedenavond” wordt afgedrukt.

In afbeelding 22 hebben we het woord tijd gebruikt om aan te geven dat we eigenlijk het uur van de dag wilden weten. Tijd kunnen we hier een variabele noemen. Net als temperatuur in afbeelding 23. Maar wat is eigenlijk een variabele?

6.4 Variabelen

Wanneer je in je programmeertaal een bepaalde waarde of een object of een stukje tekst op meerdere plaatsen wilt gebruiken dan geven we zo’n waarde, zo’n object of stukje tekst, vaak een naam. We noemen het dan een variabele.

Laten we eens naar het voorbeeld van de tijd kijken uit afbeelding 22. We gebruiken daarin het begrip tijd bij het invoeren van de tijd en verderop in het keuzeblok hebben we de tijd weer nodig. We zouden het PSD als volgt aan kunnen passen:

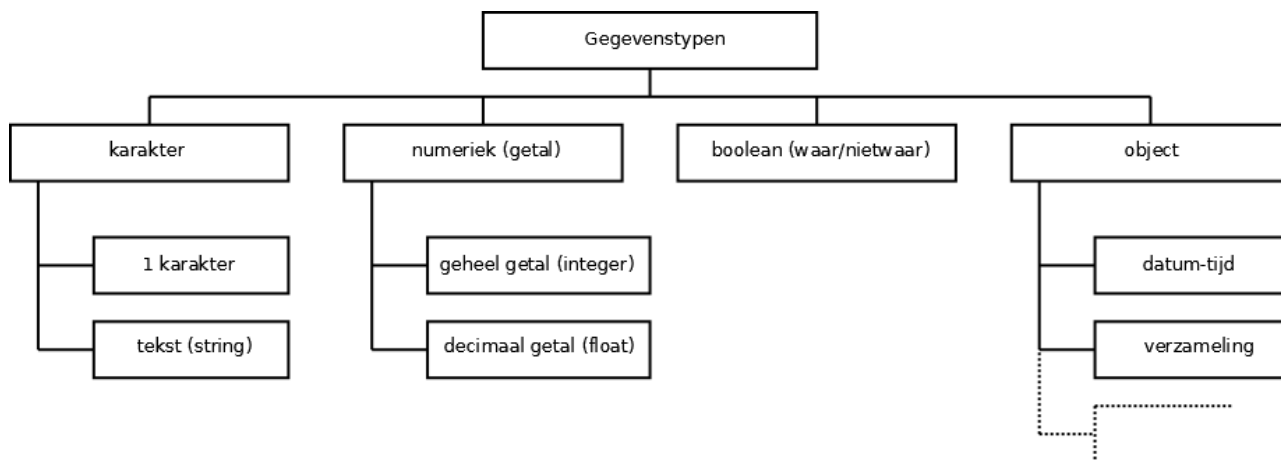
We maken hier dus gebruik van de variabele *t*. De computer slaat de tijd ergens op in zijn geheugen en verderop kunnen we de tijd weer oproepen aan de hand van de variabele *t*.

Variabelen is een heel belangrijk begrip. Een programmeertaal zonder variabelen is nauwelijks voorstelbaar. Wel gaan de diverse programmeertalen verschillend met variabelen om.

In sommige programmeertalen kun je een variabele zomaar gebruiken. Door simpel aan een variabele een waarde toe te kennen bestaat ineens die variabele. In andere programmeertalen moet je de variabele eerst maken. Declareren noemen we dat. Tijdens het declareren krijgt de variabele een naam, wordt aangegeven welk type variabele het is en wordt ook geregeld waar de variabele allemaal bereikbaar is. Laten we dat eens even beter bekijken.

Gegevenstypen

Wanneer de computer met gegevens werkt moet hij wel weten om welk soort gegevens het gaat. Met een tekst zal hij anders om gaan dan met een getal. Maar iets wat een getal lijkt kan eigenlijk best een tekst zijn. Zo wordt een telefoonnummer meestal beschouwd als tekst omdat je er niet mee gaat rekenen. We onderscheiden de volgende gegevenstypen:



Afbeelding 24: Gegevenstypen

Toch wel een bijzonder gegevenstype is het object. Eigenlijk kan een object van alles zijn. Meestal gaat het om een verzameling gegevens. Een compleet adres, met straat, huisnummer, post-code en woonplaats zou een object kunnen zijn. Maar ook een matrix kan als één object vastgelegd worden. Veel programmeertalen kennen een aantal standaard objecten zoals een datum-tijd object.

Declareren en initialiseren

Het declareren van een variabele is het vastleggen van de variabele met het bijbehorende gegevenstype. Het initialiseren geeft een beginwaarde aan de variabele. In JavaScript declareer en initialiseer je meestal in één stap. Dat gaat dat als volgt:

```
var x=10;
var straatnaam="Boterstraat";
```

Binnen JavaScript wordt het gegevenstype dus afgeleid van de invulling van de variabele. Dus doordat je aan een variabele een waarde toekent “weet” de computer ook om welk gegevenstype het gaat.

Hier nog een voorbeeld:

```
<script type="text/javascript">
  var datum = new Date()
  var tijdstip = datum.getHours()
  if (tijdstip<10) {
    document.write("<b>Goedemorgen</b>");
  }
  else if (tijdstip>10 && t<16) {
    document.write("<b>Goedendag</b>");
  }
  else {
    document.write("<b>Hallo Wereld!</b>");
  }
</script>
```

Met `var datum = new Date()` wordt aan de variabele `datum` een datum-object toegevoegd. (Eigenlijk moet je hier zeggen dat er een nieuwe instantie van het datum-object wordt

gemaakt.) Het datum-object is een standaard in JavaScript aanwezig object waarin allerlei zaken aanwezig zijn die betrekking hebben op de datum en de tijd.

Op de volgende regel wordt met `datum.getHours()` de methode `getHours()` van het datum-object `datum` aangeroepen. Daarmee wordt aan de variabele `tijdstip` een getal toegekend waarmee het uur wordt aangegeven. Kun je de rest van de code volgen?

Opdracht 22

Maak een PSD van het JavaScript programma hierboven.

Opdracht 23

Wat gebeurt er in dit programma als `tijdstip` de waarde 10 krijgt?

In de JavaScript code hebben we een aantal zogenaamde operatoren gebruikt. Tijd om die ook eens op een rijtje te zetten.

6.5 Operatoren

Met operatoren kun je bewerkingen uitvoeren op variabelen. Bijvoorbeeld een waarde toekennen aan een variabele maar ook met variabelen rekenen. We kennen een aantal verschillende operatoren:

Toekeningsoperator

Met de toekeningsoperator kun je (nieuwe) waarden toekennen aan variabelen en objecteigenschappen. De toekeningsoperator in JavaScript is het `=` teken.

Rekenkundige operatoren

Met de rekenkundige operatoren kun je getallen optellen, aftrekken, delen en vermenigvuldigen. Meestal gebruik je hiervoor de tekens `+`, `-`, `/` en `*`.

Relationele operatoren

Met de relationele operatoren kun je relaties tussen waarden of variabelen testen met als resultaat *waar* of *niet-waar*. Zo heb je groter dan (`>`), groter dan of gelijk aan (`>=`), kleiner dan (`<`) en kleiner dan of gelijk aan (`<=`).

Verder heb je nog is gelijk aan (`==`) en is ongelijk aan (`!=`).

Logische operatoren

Met de relationele operatoren kun je de relaties testen en kijken of aan een bepaalde voorwaarde wordt voldaan. Maar soms wil je dat aan meerdere voorwaarden voldaan wordt. In de code op bladzijde 39 zie je daar een voorbeeld van bij: `if (t > 10 && t < 16)`. Hier wordt dus gekeken of `t` groter is dan 10 maar ook kleiner is dan 16. Als aan beide voorwaarden voldaan wordt is de uitdrukking tussen de haakjes waar. De `&&` noemen we de and-operator. Daarnaast kennen we de or-operator die we weergeven met twee verticale streepjes (`||`). De derde logische operator is de not-operator. Die geven we aan met het uitroepteken (`!`).

6.6 Functies en methoden

De benamingen functies en methoden worden vaak door elkaar gebruikt. Ze doen dan ook min of meer hetzelfde. Een methode is een onderdeel van een object terwijl een functie niet gebonden is aan een object. In de code op bladzijde 39 wordt de methode **getHours()** die onderdeel is van een datum-object aangeroepen. Een methode of een functie leveren vaak een resultaat op. In het geval van de methode **getHours()** levert dit het getal op dat het uur aangeeft. Soms geef je bij het aanroepen van een functie (of methode) waarden mee. De functie doet daar dan iets mee en geeft soms een resultaat terug. Een functie om van twee getallen het grootste getal te bepalen zou er als volgt uit kunnen zien:

```
function max(x,y) {  
    if (x > y){  
        return x  
    } else {  
        return y  
    }  
}
```

Zo'n functie kun je dan in de code aanroepen met bijvoorbeeld:

```
var a=6;  
var b=8;  
hoogste = max(a,b);
```

De variabele **hoogste** zal in dit voorbeeld de waarde 8 krijgen. Het aanroepen van een functie doen we dus door de functienaam te gebruiken met daarachter de haakjes. Tussen de haakjes komen eventuele parameters.

In de meeste gevallen zul je een functie zelf moeten schrijven maar JavaScript kent ook enkele ingebouwde functies. Bijvoorbeeld de functie **parseInt()**. Met **parseInt()** kun je van een getal uit een tekstblok (een string dus) een integer maken zodat je er mee kunt rekenen. Met **parseFloat()** kun je van een getal uit een tekstblok een getal met decimalen terugkrijgen.

Zo geeft **parseInt("3")** het getal 3

Ook **parseInt("3.14")** geeft het getal 3, immers **parseInt()** geeft altijd een integer terug. Zodra hij in die functie iets anders dan een getal tegenkomt stopt de functie en geeft het resultaat tot zover terug.

parseFloat("3.14") zal als resultaat het getal 3.14 geven.

6.7 Commentaar

Als je een computerprogramma schrijft weet je precies waar je mee bezig bent. Maar als je na enkele maanden die code weer ziet zul je versteld staan hoe ingewikkeld die code dan weer lijkt. Maar het kan ook voorkomen dat jouw code door iemand anders bekeken wordt of iemand anders moet er wijzigingen in aanbrengen. Om deze redenen is het belangrijk commentaarregels aan jouw code toe te voegen. Eigenlijk is het beter te spreken van opmerkingen maar de algemene term die in Nederland gebruikt wordt is commentaar. Het is afgeleid van het Engelse comment dat naast commentaar ook vertaald kan worden met opmerking of uitleg.

CSS

Op bladzijde 23 heb je al kunnen lezen hoe we commentaar in CSS code kunnen toevoegen, namelijk beginnend met **/*** en eindigen met ***/**.

HTML

Commentaar in HTML code voeg je toe door deze tussen `<!--` en `-->` te plaatsen.

JavaScript

In JavaScript kun je op twee manieren commentaar toevoegen. Je kunt zowel een regel commentaar toevoegen als een blok commentaartekst. Dat gaat als volgt.

`//` Alles wat hier staat tot aan het einde van de regel wordt door de browser genegeerd.

`/*` Alles wat hier staat tot aan de afsluiting wordt genegeerd, ook al beslaat de tekst die je hebt meerdere regels, de afsluiting is `*/`.

6.8 Tenslotte

Nog enkele opmerkingen over het gebruik van JavaScript. Sluit een opdrachtregel altijd af met een punt-komma (`;`) tenzij het einde van de opdrachtregel(s) duidelijk is, bijvoorbeeld als je een blok afsluit met een accolade (`}`).

Laat de code altijd goed inspringen. Dit bevordert de leesbaarheid van jouw code en je ziet veel sneller waar de fouten zitten.

Gebruik namen voor variabelen die een duidelijke betekenis hebben zodat herkenbaar is waar het om gaat.

JavaScript heeft meer mogelijkheden dan hier behandeld is. Wil je meer weten, kijk dan bijvoorbeeld op <http://www.w3schools.com/JS/default.asp>. Maar op internet vind je nog veel meer JavaScript cursussen, ook in het Nederlands.

7 Interactief

Tot nu toe hebben we ons bij het maken van een website bezig gehouden met de inhoud en de opmaak. De inhoud was de tekst zoals we die in HTML konden typen. De opmaak waren de codes die we door middel van CSS meegaven aan de diverse elementen van de website.

We hebben nog maar een heel klein stukje van HTML en CSS behandeld maar eerst gaan we nog even naar een ander belangrijk punt van een website kijken, namelijk interactiviteit. Een website is vrijwel nooit alleen maar een pagina met tekst. Vaak gaat het er om delen van de website te kunnen aanklikken en zo ervoor te zorgen dat er iets gebeurt. Dat kan het verschijnen van een nieuwe pagina zijn maar ook het uitvoeren van een berekening of het invoeren van gegevens.

Ook interactiviteit kan op verschillende manieren tot stand komen. Maar heel vaak wordt daarvoor gebruik gemaakt van JavaScript. Dus tijd om een eerste blik te werpen op JavaScript.

JavaScript

Bij CSS schreven we al: houd inhoud en opmaak altijd gescheiden.

Eigenlijk geldt dat ook voor de code die we met JavaScript gaan toevoegen. Het zou goed zijn de JavaScript code in een apart bestand te plaatsen. En dat gebeurt ook veel. We lezen dan in de header een `.js` bestand in en daarmee wordt de JavaScript code in dat `.js` bestand een onderdeel van onze webpagina. En we hebben het al eerder zo gedaan. Op bladzijde 12 gebruikten we deze code: `src="http://html5shiv.googlecode.com/svn/trunk/html5.js"` en daarmee lazen we het `html5.js` bestand in om een oudere browser te laten voldoen aan een aantal regels van HTML 5.

Toch zullen we in onze voorbeelden meestal de JavaScript code in de HTML pagina plaatsen. Het grote voordeel daarvan is dat je de code daar neer zet waar je hem ook nodig hebt. Want de code die wij schrijven wordt vaak maar op één webpagina gebruikt in tegenstelling tot de CSS code die vaak voor een hele website geldig is.

JavaScript is een object georiënteerde programmeertaal om kleine stukjes programma in een website uit te kunnen voeren. Met JavaScript kun je een website veel leuker maken omdat je meer ter beschikking hebt dan alleen maar tekst, links en plaatjes. We zeggen ook wel dat je met JavaScript een website interactiever kunt maken.

Maar niet altijd gaat het bij JavaScript om kleine stukjes programma. Bijna iedereen kent Gmail wel. Het Gmail programma is een website die 443.000 regels JavaScript (of 978.000 regels als je de commentaarregels meetelt) bevat.

JavaScript is in 1995 ontwikkeld door Brendan Eich van Netscape. De combinatie van HTML, CSS en JavaScript wordt ook wel Dynamic HTML (DHTML) genoemd. Voor de moderne websites is JavaScript van groot belang. Omdat ook JavaScript, net als CSS, een toevoeging is aan HTML moet je wel een browser hebben die JavaScript kan draaien. Maar alle moderne browsers kunnen dat ook al zitten er nog grote verschillen in de snelheid waarmee de JavaScript opdrach-

ten verwerkt worden. Bovendien wordt soms uit veiligheidsoverwegingen het uitvoeren van scripts geblokkeerd. Je moet dan apart aangeven of zo'n script wel gebruikt mag worden.

7.1 JavaScript tussen HTML

Als je JavaScript code wilt gebruiken kun je dat in jouw html-pagina plaatsen. Maar om ervoor te zorgen dat de browser begrijpt dat het om JavaScript gaat moet het tussen `<script>` tags geplaatst worden. Kijk maar eens naar dit voorbeeld:

```
<!DOCTYPE html>
<html lang="nl">
  <head>
    <meta charset="utf-8">
    <title>Dit is JavaScript</title>
  </head>
  <body>
    Dit is gewone html tekst<br>
    <script type="text/javascript">
      document.write("Dit is JavaScript tekst!")
    </script>
    <br>
    En dit is weer HTML!
  </body>
</html>
```

Opdracht 24

Maak een nieuwe html-pagina en sla deze op onder de naam `opdracht24.html`.
Typ daarin bovenstaand voorbeeld en test de pagina uit.

Kreeg je het volgende op het scherm?

Dit is gewone html tekst
Dit is JavaScript tekst!
En dit is weer HTML!

Je zult je misschien afvragen waarom we hier JavaScript gebruiken. Want er is in de browser geen verschil te zien tussen de manier waarop de regels afgebeeld worden. En dat verschil is er ook inderdaad niet. Toch zullen we zien dat de opdracht **`document.write`** een belangrijke opdracht in JavaScript is.

In dit voorbeeld gaat het er alleen maar om te laten zien hoe JavaScript in een html-pagina kan worden gebruikt. Belangrijk is de eerste regel van de `<script>` tag waarin wordt aangegeven welke scripttaal er gebruikt gaat worden.

7.2 Actie

Niet altijd hoeven we de `<script>` tag te gebruiken om JavaScript te laten uitvoeren.

Opdracht 25

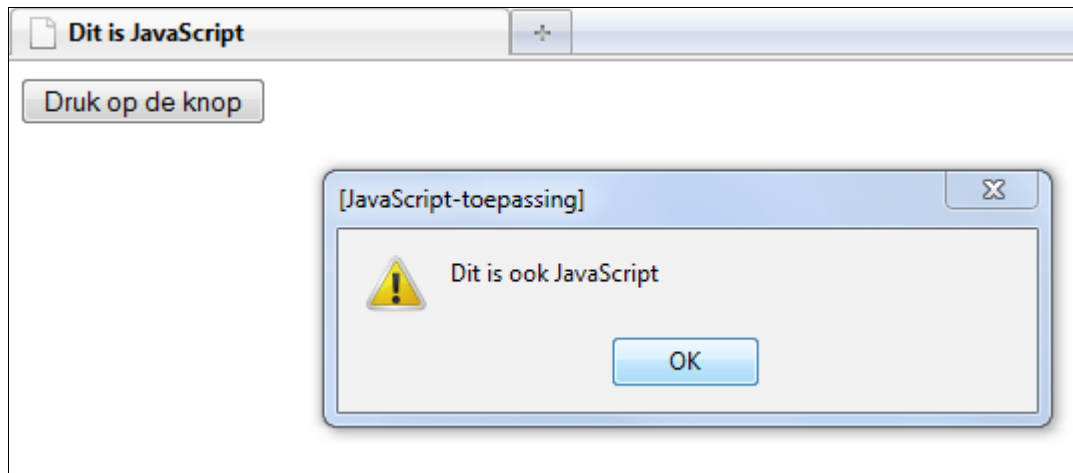
Verander het bestand `opdracht24.html` door in de body de volgende code te plaatsen en sla de pagina dan op onder de naam `opdracht25.html`.

```
<body>
```



```
<input type="button" value="Druk op de knop"
      onClick="alert('Dit is ook JavaScript')">
</body>
```

In Firefox ziet de uitvoer er dan zo uit:



Afbeelding 25: Actie met JavaScript

De `<input>` tag is een HTML opdracht die we nog niet hebben behandeld. Toch zal duidelijk zijn wat er mee wordt bedoeld. Met `type="button"` wordt er een knop op het scherm geplaatst. De tekst die op de knop komt kunnen we aangeven achter `value` en bij het `onClick` attribuut kunnen we de pagina vertellen wat er moet gebeuren.

Zoals je ziet kan na het `onClick` attribuut zomaar tussen de aanhalingstekens een stukje JavaScript code staan. Hier zijn de `<script>` tags niet eens voor nodig.

Zie je ook dat de hele JavaScript code tussen dubbele aanhalingstekens (") staat maar de tekst 'Dit is ook JavaScript' tussen enkele? Dat is iets om goed in de gaten te houden. Zouden we namelijk de tekst ook tussen dubbele aanhalingstekens plaatsen, dan zou voor de browser de JavaScript code na `alert(` al weer ophouden omdat het dan afsluitende aanhalingstekens zouden zijn van het eerste gedeelte. en dat is natuurlijk niet de bedoeling.

Met `onClick` heb je gezien dat je vrij eenvoudig wat actie op jouw pagina kunt aanbrengen. Laten we nog eens een voorbeeld nemen van actie of, beter gezegd, een gebeurtenis of event.

7.3 Formulier

Vaak heb je mogelijkheden op een website om iets in te vullen. Een naam, een getal, het kan van alles zijn. In opdracht 26 maak je zo'n formulier en doe je iets met dat wat er ingevuld wordt.

Opdracht 26

Verander de body van `opdracht25.html` met de volgende code en sla de pagina op onder de naam `opdracht26.html`.

```
<body>
  <form>
    Hoe heet jij?
    <input type="text" name="naam" value="">
    <br><br>
```

```

    <input type="button" value="Zeg eens vriendelijk gedag!"
      onClick="alert('Hallo ' + naam.value)">
  </form>
</body>

```

Nieuw hierbij is de tag `<form>`. Tussen `<form>` en `</form>` wordt als het ware een formulier gedefinieerd. En binnen dat formulier kunnen elementen een naam krijgen waarmee we ze kunnen oproepen. Zo geven we het `input` element de naam: `"naam"`. Verderop, in de `alert` van de knop, roepen we met `naam.value` de waarde die we ingetypt hebben weer op en zetten die in het pop-up venster. Omdat we hier `naam.value` aanroepen binnen hetzelfde form als waar hij ook gemaakt is, kunnen we met deze eenvoudige oproep volstaan. Maar we zullen zien dat dat niet altijd kan. Soms heb je een uitgebreidere aanroep nodig als:

`document.mijnForm.naam.value`.

Hier is `document` de html-pagina en `mijnForm` de naam die we aan het formulier gegeven hebben (dat doe je met: `<form name="mijnForm">`).

Probeer steeds iedere regel helemaal te begrijpen. Ieder woordje heeft zijn betekenis. Als je die betekenis begrijpt, snap je al gauw hoe je het ook in andere situaties moet toepassen.

7.4 Functies

Een functie is een stukje programmacode dat we uit kunnen laten voeren. Laten we simpel beginnen. Stel we willen twee getallen bij elkaar op gaan tellen.

Opdracht 27

Maak een nieuwe html-pagina, sla die op onder de naam `opdracht27.html` en typ daarin de volgende code:

```

<!DOCTYPE html>
<html lang="nl">
  <head>
    <meta charset="utf-8">
    <title>Een functie in JavaScript</title>
    <script type="text/javascript">
      function berekening() {
        var x= 5;
        var y= 9;
        var resultaat= x + y;
        alert(resultaat);
      }
    </script>
  </head>
  <body>
    <form>
      <input type="button" value="Bereken"
        onClick="berekening()">
    </form>
  </body>
</html>

```

Hé, dat is nieuw. De JavaScript functie met de naam `berekening()` zit in de head sectie. Dat zie je vaak. Functies worden, om zo de body overzichtelijk te houden, in de head sectie geplaatst. Een functie herken je (overigens in heel veel programmeertalen) aan de `()` aan het eind. Deze functie neemt twee variabelen, `x` en `y`, en geeft die een waarde (`5` en `9`). Daarna wordt er

een variabele **resultaat** gemaakt die de waarde van **x** en de waarde van **y** bij elkaar op telt (**resultaat** wordt dus 14). Tenslotte wordt met een **alert** de variabele op het scherm geplaatst.

Maar zo'n functie in de head sectie doet uit zichzelf niets. Die moet worden aangeroepen en dat gebeurt met de **onClick**-event van de knop die in de body op het scherm wordt geplaatst. Zie je die aanroep staan?

Spannend? Nou niet echt. Want alsmäär dezelfde getallen bij elkaar optellen is niet zo'n nuttige functie. Het zou leuker zijn wanneer we de beide getallen eerst zelf in zouden kunnen typen. Eigenlijk weet je al een beetje hoe dat moet. Laten we eerst twee invulvelden maken op het scherm. Daar komen de getallen in. En als we dan op de knop klikken moet hij het totaal geven.

Opdracht 28

Neem opdracht27.html, sla deze eerst op onder de naam opdracht28.html en verander het form gedeelte als volgt:

```
<form name="mijnForm">
  <input type="tekst" name="getal1" value="">
  <br>
  <input type="tekst" name="getal2" value="">
  <br>
  <br>
  <input type="button" value="Bereken" onClick="berekening()">
</form>
```

Daarmee hebben we het schermgedeelte al aangepast. Nu moeten we alleen de functie **berekening()** nog vertellen dat hij de getallen die we invoeren gaat gebruiken. Pas daarvoor de functie in de head sectie als volgt aan:

```
<script type="text/javascript">
  function berekening() {
    var resultaat= document.mijnForm.getal1.value +
      document.mijnForm.getal2.value;
    alert(resultaat);
  }
</script>
```

Je ziet dat ik de **value** van **getal1** neem uit **mijnForm** dat een onderdeel is van het gehele document. Daar tel ik de **value** van **getal2**, ook uit **mijnForm**, bij op. Maar is het resultaat ook wat je had verwacht?

Nee, dat is het niet. Als je de getallen 2 en 3 hebt ingevuld geeft hij 23 als antwoord. Hij zet dus de getallen domweg achter elkaar, alsof het gewone tekst is. Maar rekent er niet mee. Dus we zijn nog niet klaar. We moeten hem per getal vertellen dat het ook echt om een getal gaat. Dat kunnen we doen met **parseInt** als het gaat om hele getallen of met **parseFloat** als het gaat om getallen met cijfers achter de komma. In dit geval weten we niet welke getallen ingetypt zullen worden dus kiezen we voor **parseFloat**. Pas de volgende regel in de functie als volgt aan:

```
var resultaat= parseFloat(document.mijnForm.getal1.value) +
parseFloat(document.mijnForm.getal2.value);
```

Als het goed is, klopt het resultaat nu wel. Probeer het ook uit met cijfers achter de komma (pas op, op de computer moeten we vaak de punt gebruiken op die plaats waar we in Nederland gewend zijn een komma te gebruiken).

Je hebt daarnet al gezien hoe je twee (tekst)velden achter elkaar kunt laten schrijven. Bij de getallen was dat niet te bedoeling maar bij de volgende opdracht wel.

Opdracht 29

Maak een nieuwe html-pagina, sla die op onder de naam `opdracht29.html` en zorg ervoor dat die pagina drie invoervelden en een knop bevat zoals hiernaast.

Bij het klikken op de knop komt de naam op de juiste manier achter elkaar te staan in een alert venster.

Pas op: JavaScript maakt onderscheid tussen hoofd- en kleine letters. Dus `mijnForm` is iets anders dan `mijnform`.



A screenshot of a web form. It consists of three text input fields stacked vertically. The first field contains the text 'Margreeth', the second contains 'de', and the third contains 'Boer'. Below these fields is a button with the text 'Voeg samen'.

Afbeelding 26: Opdracht 28

7.5 Keuze

We kunnen door middel van JavaScript ook iets laten gebeuren als er aan een bepaalde voorwaarde wordt voldaan. Voorwaarden worden meestal afgehandeld met `if...else...` oftewel het keuzeblok. Tegelijkertijd kijken we wat er allemaal met input mogelijk is binnen HTML. We gaan daarvoor een pagina maken met daarin zogenaamde radiobuttons en checkboxes.

Opdracht 30

Maak een nieuwe html-pagina, sla die op onder de naam `opdracht30.html` en typ daarin de volgende code:

```
<!DOCTYPE html>
<html lang="nl">
  <head>
    <meta charset="utf-8">
    <title>Restaurant Pisa</title>
    <link rel="stylesheet" type="text/css"
      href="opdracht30.css">
    <script language="JavaScript">
      function bestellen() {
        mijnTekst = "U heeft besteld: ";
        if (document.mijnForm.eten1.checked) {
          mijnTekst += "pizza";
        } else if (document.mijnForm.eten2.checked) {
          mijnTekst += "pasta";
        } else {
          mijnTekst += "calzone";
        }
        mijnTekst += " met ";
        if (document.mijnForm.vlees.checked) {
          mijnTekst += "vlees ";
        }
        if (document.mijnForm.vis.checked) {
          mijnTekst += "vis ";
        }
        if (document.mijnForm.kaas.checked) {
          mijnTekst += "kaas.";
        }
        alert(mijnTekst);
      }
    </script>
  </head>
```

```

<body>
  <h2>Restaurant Pisa</h2>
  <br>
  <form name="mijnForm">
    <input type="radio" id="eten1" name="eten"
      value="Pizza" checked> Pizza
    <br>
    <input type="radio" id="eten2" name="eten"
      value="Pasta"> Pasta
    <br>
    <input type="radio" id="eten3" name="eten"
      value="Calzone"> Calzone
    <br>
    <br>
    <input type="checkbox" id="vlees" name="vlees"
      checked> Vlees
    <br>
    <input type="checkbox" id="vis" name="vis"> Vis
    <br>
    <input type="checkbox" id="kaas" name="kaas"> Kaas
    <br>
    <input type="button" name="knop" value="Bestellen"
      onClick="bestellen()">
  </form>
</body>
</html>

```

Maak ook een bestand aan met de naam opdracht30.css en typ daarin de volgende code:

```

body {
  background-image:
    url("http://www.moorsmagazine.com/images/
    leaning-tower-pisa.jpg");
  background-attachment: fixed;
  background-repeat: no-repeat;
  background-position: 200px 20px;
}

```

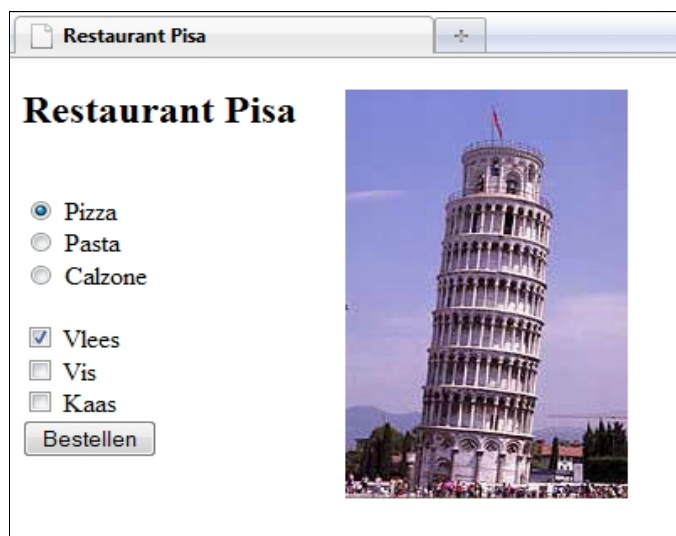
Pas op dat de link achter background-image helemaal doorloopt op één regel!

Als het goed gegaan is ziet jouw pagina eruit als in afbeelding 27.

Zoals je ziet is het een pagina met een functie in de head sectie en een aantal aanklikbare gedeelten in de body. Laten we allereerst maar eens naar de body kijken.

Je ziet daar drie radiobuttons en drie checkboxen. Bij een radiobutton kies je één van de drie, bij een checkbox kun je één, twee of alle drie aanvinken.

Binnen HTML kun je legio radiobuttons op je scherm plaatsen maar die horen dan nog niet automatisch bij elkaar. Maar dat moet wel be-



Afbeelding 27: Opdracht30

kend zijn. Immers, als je één rondje van de radiobutton aan klikt moeten de anderen uit gaan. Door radiobuttons die bij elkaar horen dezelfde naam te geven snapt HTML dat ze op elkaar moeten reageren. Hier hebben alle radiobuttons `name="eten"` waardoor ze ook echt als radiobuttons gaan werken.

Om nu te achterhalen welke van de radiobuttons en welke van de checkboxen geselecteerd zijn maken we gebruik van `if...else...` in de JavaScript functie.

Laten we de eerste if eens bekijken. Er staat daar:

```
if (document.mijnForm.eten1.checked) {  
    mijnTekst += "pizza";  
}
```

In iets meer gewone mensentaal betekent dat: als er aan de voorwaarde tussen haakjes voldaan is, doe dan de code die na dat gedeelte staat.

De voorwaarde is: `(document.mijnForm.eten1.checked)`. Zo'n voorwaarde is waar of niet waar. Computerprogrammeurs spreken van True of False. De vraag vertalend kunnen we hier lezen: is op deze pagina (`document`), in `mijnForm` het element met id `eten1` gecheckt? Met gecheckt bedoelen we of die is geselecteerd. En als hij inderdaad gechecked is, moet de code die erachter staat, uitgevoerd worden.

Ook die code is wat cryptisch. Eerder in de functie is de variabele `mijnTekst` al vastgelegd en kreeg toen de tekst mee: `"U heeft besteld: "`. Door daar `+= "pizza"` achter te plaatsen wordt de tekst `pizza` achter die eerste tekst geplaatst en komt er dan in de variabele de tekst: `"U heeft besteld: pizza"` te staan. Zo kunnen we dus stukjes tekst samenvoegen.

Maar wat nu als aan de voorwaarde niet voldaan is? Dus als het deel tussen haakjes niet waar is? In dat geval wordt er verder gekeken. Staat er een `else` opdracht daarna? Dan wordt de code na `else` uitgevoerd. In ons geval is dat soms weer een `if` dus dan wordt er opnieuw een waar of niet waar vraag gesteld, een keuzeblok dus. De `if`-voorwaarde eindigt tenslotte met een `;` (punt-komma).

Kijk ook nog eens even naar het CSS-deel in het bestand `opdracht30.css`. Daar wordt een achtergrondplaatje ingeladen. Kijk maar eens wat het effect is als je `no-repeat` verandert in `repeat`. En ook wat het effect is als je de getallen achter `background-position` aanpast.

7.6 Objecten

Met JavaScript kun je ook objecten gebruiken. Wil je bijvoorbeeld werken met een datum of met de tijd dan gebruik je een instantie van het `date()` object. Hieronder een programma waarin de datum en de tijd worden gebruikt. De code wordt verklaard in de commentaar-regels.

Opdracht 31

Maak een nieuwe html-pagina, sla die op onder de naam `opdracht31.html` en typ daarin de volgende code:

```
<!DOCTYPE html>  
<html lang="nl">  
  <head>  
    <meta charset="utf-8">  
    <title>De klok</title>  
    <script type="text/javascript">  
      /* Het script begint met het vastleggen van twee
```

```

tekstvariabelen, tijdStr en datumStr*/
var tijdStr, datumStr;
/* Hieronder begint de functie klok*/
function klok() {
    /* In de variabele nu wordt een datumobject geplaatst.
    Automatisch wordt die gevuld met de huidige datum*/
    nu = new Date();
    /* Eerst gaan we de tijd opbouwen.
    De uren worden uit de datum gehaald en in de variabele
    uren geplaatst.*/
    uren = nu.getHours();
    /* Ook de minuten en de seconden worden uit datum
    gehaald.*/
    minuten = nu.getMinutes();
    seconden = nu.getSeconds();
    /* Dan wordt nu tijdStr opgebouwd.
    Eerst worden de uren erin geplaatst*/
    tijdStr = "" + uren;
    /* en nu de minuten en de seconden.
    Dit deel wordt verderop nog uitgelegd.*/
    tijdStr += ((minuten < 10) ? ":0" : ":") + minuten;
    tijdStr += ((seconden < 10) ? ":0" : ":") + seconden;
    /* Nu wordt tijdStr in het tekstblok op het scherm
    geplaatst */
    document.klok.tijd.value = tijdStr;
    /* Iets soortgelijks als met de tijd wordt nu met de
    datum gedaan.*/
    dag = nu.getDate();
    maand = nu.getMonth()+1;
    jaar = nu.getFullYear();
    datumStr = ((dag < 10) ? "0" : "") + dag;
    datumStr += ((maand < 10) ? "-0" : "-") + maand;
    datumStr += "-" + jaar;
    /* en nu wordt datumStr in het tekstblok op het scherm
    geplaatst.*/
    document.klok.datum.value = datumStr;
    /* Een timer roept elke 1000 milliseconden de functie
    klok aan. Daardoor ververs de klok zich elke seconde.*/
    Timer = setTimeout("klok()",1000);
}
</script>
</head>
<!-- Als de body gestart wordt, wordt meteen ook de
functie klok() gestart-->
<body onLoad="klok()">
    <form name="klok">
        Tijd:
        <input type="text" name="tijd" size="10" /><br />
        Datum:
        <input type="text" name="datum" size="10" /><br />
    </form>
</body>
</html>

```

In deze code komen we nog het volgende tegen:

```

tijdStr += ((minuten < 10) ? ":0" : ":") + minuten;

```

Je kunt het gedeelte tussen haakjes als volgt lezen: Als (**minuten** < **10**) dan schrijf je " : 0 ", anders schrijf je " : "

Dus als de minuten op 6 staan dan maakt hij er keurig 06 van. Datzelfde passen we ook toe bij de seconden en bij de datum. Dit is dus eigenlijk een verkorte schrijfwijze van *if... else...*

Er zijn nog veel meer objecten in JavaScript. Zoals het Array object en het Math object. Maar het gaat te ver om die hier allemaal te noemen. Met het datumobject heb je al een indruk gekregen wat je er mee kunt doen. Objecten zijn dus kant en klare stukjes code waaruit je gegevens kunt halen of gegevens in kunt stoppen.



Afbeelding 28: Opdracht 31

Wil je meer weten? Op het internet staan heel veel voorbeelden van JavaScript toepassingen en JavaScript objecten. Er zijn echter ook kant en klare bibliotheken met JavaScript code. Bijvoorbeeld YUI (van Yahoo) en jQuery. Dit is JavaScript code die vrij verkrijgbaar is, vaak goed getest op alle browsers en waar veel documentatie over beschikbaar is.

Opdracht 32

Maak een website om het eindbedrag in te vullen als we een bepaald bedrag op een spaarrekening zetten en we rente ontvangen over het bedrag. De website bevat drie invulvelden. Een veld voor het beginbedrag, een veld voor het rentepercentage en een veld voor het aantal jaren. Na een druk op de knop moet er met JavaScript berekend worden hoe groot het bedrag wordt uitgaande van de rente en het aantal jaren die ingevuld zijn. Laat dit eindbedrag met een alert op het scherm verschijnen.

Voor je de JavaScript code schrijft moet je er eerst een PSD van maken!

Opdracht 33

Maak een website om valuta om te rekenen. Geef de keuze van de valuta aan door middel van radiobuttons. Bedenk zelf hoe het scherm eruit moet gaan zien. Maak weer eerst een PSD voordat je de JavaScript code schrijft.

8 DOM

Een HTML-pagina bestaat uit objecten. Eerder (zie bladzijde 12) spraken we al over het DOM. DOM betekent Document Object Model. De body van een HTML-pagina wordt het document genoemd. Alle tags vormen de objecten in dat document. Met JavaScript kunnen we de objecten manipuleren.

DOM
document

In opdracht 11 hebben we een menu ontwikkeld. Het wordt nu tijd daar nog wat meer mee te gaan doen.

Dat menu had de keuzes Appels, Peren, Bananen. De bedoeling is dat bij het klikken op die links plaatjes, eventueel aangevuld met tekst, van appels, peren en bananen zichtbaar worden. Zoals in afbeelding 29.



Afbeelding 29: Opdracht 32

Opdracht 34-1

Roep de bestanden opdracht11.html en opdracht11.css op en bewaar deze onder de namen opdracht34.html en opdracht34.css.

Opdracht 34-2

We gaan nu eerst opdracht34.html aanpassen. Daar moeten immers verwijzingen naar de afbeeldingen in komen? Maar ook de link naar het CSS-bestand moet aangepast worden.

Verander de link in de head sectie als volgt:

```
<link rel="stylesheet" type="text/css" href="opdracht34.css" />
```

En nu de plaatjes in de tekst. In opdracht 12 heb je die plaatjes van internet gehaald. De links daarnaar komen in het `section` deel.

```
<section id="inhoud">
  <p>Hier komt de inhoud van de webpagina</p>
  <div id="Appels">
    
  </div>
  <div id="Peren">
    
  </div>
  <div id="Bananen">
    
  </div>
</section>
```

Test deze code maar eens uit. De kans is groot dat jouw afbeeldingen niet allemaal even groot zijn. Dat is natuurlijk niet fraai dus daar gaan we eerst iets aan doen.

Opdracht 34-3

Voeg onderaan het bestand opdracht34.css de volgende regels toe:

```
#inhoud img {
  width: 300px;
}
```

Hiermee geeft je aan dat alle `img` referenties in `section` met het `id = inhoud` een breedte moeten krijgen van 300 pixels.

Nu zouden alle afbeeldingen even breed moeten zijn. Maar ze zijn wel allemaal zichtbaar en we willen eigenlijk pas dat ze zichtbaar worden als er op de betreffende link wordt geklikt. We beginnen daarom met eerst de afbeeldingen te verstoppen en ze dan door het klikken op de link zichtbaar te maken.

Opdracht 34-4

We beginnen weer met de HTML code. Verander het section gedeelte als volgt:

```
<section id="inhoud">
  <p>Hier komt de inhoud van de webpagina</p>
  <div id="Appels" class="verstoep">
    
  </div>
  <div id="Peren" class="verstoep">
    
  </div>
```

```

</div>
<div id="Bananen" class="verstoep">
  
</div>
</section>

```

We geven ze hier een zogenaamde class selector mee. Een class selector wordt gebruikt om de stijl van een groep elementen te bepalen. Binnen CSS kunnen we hier dus iets mee doen. Het gaat immers om de stijl! Kijk maar eens hoe we dat in CSS gaan toepassen. Voeg de volgende code onderaan jouw CSS bestand toe.

```

#Appels.verstoep, #Peren.verstoep, #Bananen.verstoep {
  display: none;
}

#Appels.zichtbaar, #Peren.zichtbaar, #Bananen.zichtbaar {
  display: block;
}

```

Voor zowel Appels, Peren als Bananen wordt hier de **verstoep** class gedefinieerd. **display: none** betekent niets anders dan dat hij dan de betreffende **div** niet mag afbeelden. Tegelijk is ook maar de zichtbaar class gedefinieerd. Met **display: block** wordt de betreffende **div** wel zichtbaar.

Test jouw code uit. Als het goed is, zijn nu de afbeeldingen onzichtbaar geworden. Dat was ook de bedoeling. Maar nu moeten we ze zichtbaar maken als we op de betreffende link klikken. Als we naar de laatste CSS code kijken dan is het dus kennelijk mogelijk de class selector van een bepaald element te wijzigen. En dat is precies wat in de volgende code gebeurt.

Opdracht 34-5

Laten we eerst de links gaan wijzigen. Het enige wat we daar doen is een JavaScript functie aanroepen.

```

<nav id="menu">
  <ul>
    <li><a href="javascript:laatZien('Appels');">Appels</a></li>
    <li><a href="javascript:laatZien('Peren');">Peren</a></li>
    <li><a href="javascript:laatZien('Bananen');">
      Bananen</a></li>
    </ul>
  </nav>

```

Tenslotte komt in de **head** sectie van `opdracht34.html` de volgende code:

```

<title>De groentenman</title>
<script type="text/javascript">
  function laatZien(divID) {
    var item = document.getElementById(divID);
    if (item.className == "verstoep") {
      item.className = "zichtbaar"
    } else {
      item.className = "verstoep"
    }
  }
</script>

```

In dit laatste gedeelte zien we dat de functie **laatZien** wordt gedefinieerd. De functie wordt aangeroepen met als argument het **divID**. In ons voorbeeld wordt de functie dus aangeroepen met als argument: **'Appels'**, **'Peren'** of **'Bananen'**.

In de volgende regel wordt de variabele **item** aangemaakt en daarin wordt een referentie opgeslagen van het element met de **divID** die meegegeven werd. Kortom, met **item** weten we, dankzij **document.getElementById** om welk element het gaat. Dit is één van de belangrijkste opdrachten voor de moderne websitebouwer.

If (item) is eigenlijk niets anders dan het bepalen of **item** daadwerkelijk wel bestaat. Daarna wordt naar de class-naam van **item** gekeken. Die verkorte schrijfwijze van een keuzeblok kwam al eerder aan de orde op bladzijde 52. Hier staat dus: Als de class-naam gelijk is aan **"verstoep"**, maak er dan **"zichtbaar"** van; zo niet, maak er dan **"verstoep"** van.

En dat is precies wat er gebeurt. Test het programma maar.

Opdracht 35

Voeg in `opdracht34.css` de volgende regel toe binnen de selector **#inhoud img**

```
#inhoud img {  
  width: 300px;  
  float: left;  
}
```

- Verklaar wat er gebeurt.
- De afbeeldingen zijn 300 pixels breed, de body is 940 pixels breed. Waarom lijkt het dan toch niet te passen?
- Bedenk twee manieren om ze wel passend te maken.

Opdracht 36

Pas de JavaScript code van opdracht 34 dusdanig aan dat er steeds maar één plaatje zichtbaar is. Dus als appels zichtbaar is en je klikt op peren, verdwijnt appels en komen de peren tevoorschijn.