WEBPROGRAMMEREN EN DATABASES

Robert G. Belleman

email R.G.Belleman@uva.nl
tel. 020 525 7272
adres Science Park 904
1098 XH Amsterdam

Assistenten:
Wendy Günther,
Sander Latour,
Camiel Verschoor,
Luuk Swartsenburg,
Sharon Gieske,
Daniël Karavolos

1. Inleiding

1.1. Tijdplan

Week	*onderwerp*	*techniek*	*resultaat*
9 - 13 jan.	Een website ontwerpen en vormgeven	HTML en CSS	Een geschikt vormgegeven site voor een bedrijf
16 - 20 jan.	Een database ontwerpen en beheren via het web	MySQL en PHP	1) Compleet ontwerp van je webtoepassing. 2) database + beheerdersdeel van de webtoepassing.
23 - 27 jan.	Het bezoekersdeel van de webtoepassing: inloggen en interactie.	PHP, javascript, AJAX	Het bezoekersdeel van de webtoepassing; vriendelijk reageren op fouten.
30 jan - 2 feb	Testen, documenteren, presenteren.	Testen en schrijven!	Webtoepassing als geheel getest en klaar, rapport, presentatie.

1.2. Werkwijze tijdens het project

De hele periode wordt in een groep van 4 of 5 personen gewerkt aan een project dat aan het eind af zal zijn. Het betreft een dynamische website, waarin interactie tussen de gebruiker en de aanbieder is verwerkt. Er zijn verschillende interactieve webtoepassingen waaruit een groep kan kiezen (zie sectie 2 van deze syllabus). Je maakt je keus voor woensdag van de eerste week.

Je mag zelf je groep samenstellen, maar elk lid van de groep dient een gelijkwaardige en significante bijdrage aan de totstandkoming van het project te bieden. Stel daarom uw groep samen uit collegas met een vergelijkbaar kennisnivo. De taakverdeling binnen een groep is altijd “horizontaal”: iedereen draagt gelijk bij aan het systeem als geheel; en dus niet “verticaal” als in: jij maakt de layout, jij schrijft de code, jij maakt de database. Elke werkgroep kiest een contactpersoon die bij problemen op tijd contact zoekt met de docent. Je maakt zelf onderlinge werkafspraken. Voor het project staat een studiebelasting van 40 uur per week.

Elke werkgroep krijgt een studentassistent toegewezen waarmee drie maal per week wordt gesproken. De studentassistent heeft twee rollen: assistent en opdrachtgever. Als assistent houdt de studentassistent de voortgang bij en kan hij/zij je helpen bij het vinden van een oplossing in het geval van problemen. Als opdrachtgever kan de studentassistent antwoord geven op inhoudelijke vragen omtrent de opdracht. Beperk vragen aan je studentassistent tot deze drie momenten per week. In het bijzonder: val ze niet lastig per email.

Tijdens de hoorcolleges worden de onderwerpen behandeld die in die week aan de orde zijn.

De rest van de tijd kun je werken in de computerzalen van het Science Park.

Behalve de website zelf lever je aan het eind een rapport op, waarin het ontwerp van de website wordt uitgelegd aan je klant. Dit rapport bestaat uit twee delen: 1) uitleg van het

ontwerp aan het management 2) technische uitleg aan de ICT-afdeling: een onderhoudshandleiding voor de site.

Aan het eind van week 4 presenteert elke werkgroep zijn werkstuk via een beamer. Stel je voor dat je de website echt voor een opdrachtgever hebt gemaakt. In deze presentatie leg je aan de opdrachtgever je ontwerp-beslissingen uit. De zaal speelt voor opdrachtgever en stelt kritische vragen. Vanuit Tutoraat/Vaardigheden wordt nog een afspraak gemaakt in week 3 om de presentatie voor te bereiden.

Aanvullende aanwijzingen worden gegeven tijdens de colleges en via Blackboard.

1.3. Webserver

Voor deze cursus is een webserver ingericht: http://websec.science.uva.nl/
Als je een browser opent naar dit adres wordt je gevraagd om een username en password: dat is respectievelijk "**webdb**" en "**webdb2012**". Na inloggen verschijnt een pagina met welkomstbericht en nuttige informatie.
Inloggen op de webserver doe je met je science account via SSH. Als je een Linux of Mac hebt, dan is een SSH client al aanwezig. Download voor Windows bijvoorbeeld PuTTY: zie de link op http://websec.science.uva.nl/.
Je kunt op deze server zelf experimenteren door in je home directory een directory met de naam `public_html` te maken. Deze komt beschikbaar als:

```
http://websec.science.uva.nl/~<jeinlognaam>
```

Zie sectie 1.3.1 voor meer informatie.

1.4. Voorbeeldcode

Er wordt in deze syllabus op een aantal plekken verwezen naar voorbeeldcode. Deze is op twee manieren te vinden:

1. Met een browser op http://websec.science.uva.nl/voorbeeldcode/
   Je krijgt dan een index te zien van de voorbeeldcode directory. Je kunt hier doorheen navigeren en het resultaat van de voorbeelden zien door er simpelweg op te klikken. Als je wilt zien *hoe* het resultaat is bereikt (de code), dan kan dat in het geval van HTML, CSS en Javascript voorbeelden meestal door de browser de "broncode" ("source") te laten zien. Dat kan echter niet in het geval van PHP aangezien die door de server wordt verwijderd. Gebruik dan de volgende manier.

2. In `/var/www/html/voorbeeldcode` als je bent ingelogd op websec.science.uva.nl. Uit deze directory kan je de voorbeelden bekijken en kopiëren naar je eigen `public_html` map als je verder wilt experimenteren.

1.5. Aanbevolen software

De meeste software die u nodig heeft voor het bouwen van een website staat op de server: websec.science.uva.nl. Deze machine is voorzien van de standaard “LAMP” software om een dynamische website te kunnen bouwen: Linux als besturingssysteem, Apache als webserver, MySQL als database management systeem, en PHP als back-end scripting taal.
Je doet er goed aan om je website te testen met meerdere verschillende browsers. Op de Windows PCs in het FNWI gebouw zijn in ieder geval Internet Explorer en Firefox geïnstalleerd, maar het kan geen kwaad om ook Chrome en Opera op je eigen laptop te installeren.

Als je lang onderweg bent tussen FNWI en thuis en onderweg wilt door kunnen werken, dan kan het handig zijn om op je eigen laptop een vergelijkbare omgeving te installeren. Kijk daarvoor eens naar EasyPHP (voor Windows - http://www.easyphp.org/), WampServer (voor Windows - http://www.wampserver.com/en/), XAMPP (voor Linux, Solaris, Windows and Mac - http://sourceforge.net/projects/xampp/) of MAMP (voor Mac - http://www.mamp.info/en/).
Gebruik voor het editen van HTML, CSS, PHP en Javascript files bij voorkeur een editor die "syntax highlighting" ondersteunt zodat syntax en typefouten snel zichtbaar zijn. Onder Linux: vim of Emacs. Onder Windows: Notepad++.
Deze cursus is bedoeld om de architectuur van dynamische websites te bestuderen. Het is daarom niet toegestaan je project te bouwen met behulp van geïntegreerde ontwikkelsoftware als DreamWeaver, FrontPage, Web Studio, Flash/AIR, etc, of een CMS ("Content Management Systems"). Maak ook geen gebruik van libraries als jQuery, Dojo, MooTools, YUI, etc. Het gebruik van ondersteunende software als HTML/CSS editors, phpMyAdmin, debuggers, etc. is natuurlijk wel toegestaan.
In deze syllabus en tijdens de colleges zul je verwijzingen tegenkomen naar andere software die nuttig kan zijn.

1.6. Literatuur

- *Deze syllabus*, bevat uitleg en oefeningen over: HTML, CSS, database ontwerp, MySQL, PHP, beveiliging en Javascript. Ook vind je hier aanwijzingen voor je project.
- *Blackboard*: er is een cursus site ingericht in Blackboard. Via "Content" komen alle noodzakelijke documenten beschikbaar.
- Er is enorme hoeveelheid materiaal beschikbaar op het web. Maak daar verstandig gebruik van! Over het algemeen kunnen antwoorden op alle vragen in dit materiaal (eventueel via Google) worden gevonden. Het is toegestaan kleine code fragmenten van het internet te gebruiken zonder bronvermelding, maar vermeld bij grote stukken altijd de bron.

  Hieronder een lijst met websites die van pas zullen komen:

  - W3C (World Wide Web Consortium) is de organisatie die de coordinatie van de webstandaarden bijhoudt. Op hun website is gedetailleerde informatie te vinden over alle standaarden:
    - http://www.w3.org/

  - W3schools is een website met een grote hoeveelheid tutorials en referentiemateriaal over alles dat je nodig hebt om een website te maken:
    - http://w3schools.com/

    Als je goed voorbereid aan deze cursus wilt beginnen, dan volg je voorafgaand aan de colleges. en zelfs voordat je de respectievelijke hoofdstukken uit deze syllabus leest, de volgende tutorials:
    - http://www.w3schools.com/html/
    - http://www.w3schools.com/xhtml/
    - http://www.w3schools.com/css/
    - http://www.w3schools.com/sql/
    - http://www.w3schools.com/php/
    - http://www.w3schools.com/js/
    - http://www.w3schools.com/htmldom/
    - http://www.w3schools.com/ajax/

- (X)HTML
  - XHTML 1.0 Strict Cheat Sheet: http://www.w3.org/2010/04/xhtml10-strict.html
  - XHTML Basic 1.1 Cheat Sheet: http://www.w3.org/2007/07/xhtml-basic-ref.html
  - HTML 4.01 / XHTML 1.0 Tag Reference http://w3schools.com/tags/default.asp
  - HTML5 Tag Reference http://w3schools.com/html5/html5_reference.asp
  - Markup Validation Service, een service die controleert of je HTML documenten goed zijn opgemaakt: http://validator.w3.org/
  - http://validator.w3.org/unicorn/

- CSS
  - CSS Property Reference: http://w3schools.com/cssref/default.asp
  - CSS Selector Reference: http://www.w3schools.com/cssref/css_selectors.asp

- PHP
  - http://www.php.net/
  - http://www.php.net/manual/en/

- MySQL
  - http://dev.mysql.com/doc/refman/5.1/en/
    Met name hoofdstukken 10 t/m 12.

- Javascript
  - http://www.devguru.com/Technologies/ecmascript/quickref/javascript_index.html
  - https://developer.mozilla.org/en/Gecko_DOM_Reference
  - https://developer.mozilla.org/en/DOM/element

2. Webtoepassingen

Bij het maken van de website kun je kiezen uit de hierna omschreven specificaties voor een interactieve webtoepassing. De keuze moet gemaakt zijn op de woensdag van week 1.

Er zijn twee categorieën specificaties: normale en moeilijke. Normale opdrachten zijn bedoeld voor groepen met een minimale voorkennis. De tweede zijn voorbehouden voor groepen die al ruime ervaring hebben met het bouwen van dynamische websites. Deze opdrachten zijn enerzijds moeilijker omdat ze vaak voortbouwen op bestaande systemen maar anderzijds uitdagender omdat het echte opdrachten zijn, opgesteld door echte opdrachtgevers. Voor deze opdrachten wordt van de groep een hoge mate van zelfstandigheid verwacht.

Bij hoge uitzondering mag je ook zelf met een specificatie komen als je met de keuzemogelijkheden niet tevreden bent. De deadline voor eigen specificaties is ook de eerste woensdag van week 1. Je voorstel wordt op de eerstvolgende voortgangsbespreking met ons besproken. We vullen deze dan eventueel aan met extra eisen en het resultaat wordt vastgelegd als afspraak.

Voor alle webtoepassingen geldt dat creativiteit op prijs worden gesteld, en dat je met een minimumaanpak (simpelweg voldoen aan de opdracht) dus niet hoog scoort. Alle werkstukken worden behalve op functionaliteit ook beoordeeld op "look and feel", gebruiksvriendelijkheid, duidelijkheid en een representatief of mooi uiterlijk. Daarnaast wordt gelet op de structuur van het programmeerwerk.

Elke site bevat, behalve de dynamische pagina's van de webtoepassing, ook een aantal statische pagina's waar informatie te vinden is over het bedrijf of de organisatie. Het statische deel van de website wordt gemaakt in week 1. Scheid vormgeving, navigatie en inhoud zoveel mogelijk (o.a. met CSS) zodat je de vormgeving ook nog kunt gebruiken voor het dynamische deel van de site.

Hieronder volgen de webtoepassingen waaruit je kunt kiezen.

2.1. Openbare agenda (normale opdracht)

Een groot bedrijf wil op het web een openbare "agenda" bijhouden van evenementen die voor werknemers, klanten en aandeelhouders van belang zijn.
De agenda is zonder inloggen te bekijken. Mensen die daarvoor rechten hebben kunnen inloggen om berichten voor de agenda klaar te maken. Daarbij kan ook worden aangegeven voor welke doelgroep(en) het bericht is bestemd. Er is een kleine redactie die alle ingezonden berichten screent en die beslist over het al dan niet openbaar maken. Er zijn dus twee groepen gebruikers met ieder een eigen interface: één voor het maken van berichten en één voor het goedkeuren.
Dan is er nog de openbare interface. Standaard verschijnt de agenda voor alle doelgroepen en voor de huidige week. Gebruikers kunnen dit veranderen door aan te geven tot welke doelgroep(en) zij gerekend willen worden. Ook kunnen zij vooruit of achteruit bladeren in de agenda. Bij klikken op een aankondiging verschijnen details (een stuk tekst over het evenement).

2.2. Vakkenpakket database (normale opdracht)

Een faculteit biedt in het derde, vierde en vijfde studiejaar een groot aantal keuzevakken. Elk vak loopt gedurende een heel trimester, ofwel 's ochtends ofwel 's middags. De bedoeling is dat studenten online kunnen inschrijven op deze vakken. Na een poging tot inschrijving krijgen zij te horen: (i) gelukt (ii) niet gelukt: maximaal aantal deelnemers is bereikt, (iii) niet gelukt: strijdig qua rooster met een eerder gekozen vak. Na inschrijving op één of meer vakken kan een student zijn rooster te zien krijgen waarbij ook aangegeven wordt welke ruimte nog in te vullen is. Om misbruik te voorkomen is inloggen met naam en wachtwoord nodig.
Na inloggen verschijnt allereerst een persoonlijke pagina, met hyperlinks naar de homepagina's van de vakken waarvoor men zich eerder heeft ingeschreven (en die nu lopen, of, van het lopende jaar). Vanaf deze pagina kan ook het rooster zichtbaar gemaakt en gewijzigd worden. Andere persoonlijke voorzieningen kunnen naar smaak worden toegevoegd.

2.3. Bestellen en faktureren (normale opdracht)

Een bedrijf stelt vaste klanten in staat om via het web bestellingen te doen. In het openbare deel van de site is een catalogus met produkten (met klikbare afbeeldingen). Vaste klanten kunnen inloggen. Zij zien dan bij elk produkt een extra link, waarmee ze het produkt in hun winkelwagentje kunnen stoppen, waarbij ook het aantal exemplaren kan worden opgegeven. Er is een "check out" link. Hierna ziet de klant de inhoud van zijn bestellingen. Het is dan nog mogelijk om naar keuze bepaalde items uit het karretje te verwijderen. Daarna kan men doorgaan met shoppen of de bestelling definitief maken.
Fakturen gaan ook via het web. Elke klant kan een overzicht zien van eerdere bestellingen. Bij elke bestelling staat vermeld of deze al is geleverd, en of de klant al heeft betaald. Beheerders van de site hebben een aparte interface waar zij geplaatste bestellingen kunnen zien. Zij kunnen per bestelling aangeven of deze bestelling geleverd is, en of deze betaald is.

2.4. Discussieforum (normale opdracht)

Een softwarebedrijf wil een serie web-supportforums opzetten zodat gebruikers van hun produkten elkaar kunnen helpen, in plaats van steeds de dure supportafdeling te moeten bellen (bijna alle bedrijven hebben dergelijke forums). Ze vragen jullie de bedrijfswebsite te maken met daarin verwerkt de forumsoftware.
Er zullen drie soorten gebruikers zijn voor deze forums:

1. *Beheerders* (mensen van het softwarebedrijf zelf) moeten nieuwe forums – elk met een eigen thema – kunnen maken en opheffen. Per forum kan worden gekozen of ingezonden berichten direct worden geplaatst, danwel eerst gescreend worden door een beheerder. Beheerders kunnen een lijst zien van ingezonden berichten die zij nog moeten goed of afkeuren. Na goedkeuring wordt het bericht zichtbaar voor iedereen, binnen het forum waarvoor het bestemd was.
2. *Ingeschreven gebruikers* kunnen goedgekeurde berichten lezen en kunnen zelf berichten insturen, via een webformulier, en zij kunnen antwoorden op eerdere berichten.
3. *Iedereen* kan (zonder inloggen) goedgekeurde berichten lezen. Ook kan men zich aanmelden als gebruiker. Er wordt dan automatisch een wachtwoord toegestuurd, waardoor de gebruiker verandert in een ingeschreven gebruiker.

Een goed forum werkt met "draden" (threads), zodat de relatie tussen berichten en antwoorden zichtbaar wordt. Wanneer iemand antwoordt op een eerder geplaatst bericht, dan

ontstaat een "draad" (thead). Bij openen van een forum zie je eerst alleen de berichten die de beginpunten zijn van alle draden. Elk bericht bevat hyperlinks naar de bijbehorende antwoorden, en ieder antwoord bevat een hyperlink naar het bericht waarop werd geantwoord.

2.5. Community-driven platform voor oriënterende informatie over master opleidingen (moeilijke opdracht)

De huidige informatiepagina's over master opleidingen van de verschillende universiteiten in Nederland zijn allemaal gemaakt door de communicatie afdeling van de desbetreffende universiteit. Dit heeft meestal tot gevolg dat deze websites erg statisch zijn en vaak nogal saai. Dat zorgt er voor dat toekomstige studenten eigenlijk helemaal niet zo'n goed beeld krijgen van de studie of hoe het zal zijn om die opleiding te volgen.
De insteek van dit project is dat er nog zoveel meer te delen is aan informatie over de opleidingen dan wat er nu op die websites staat. Tenslotte zijn er onderzoekers en studenten die misschien wel een veel beter beeld hebben van de sterke punten van de opleiding. Ook de ori□entatie van de standaardwebsites op tekst is wat ouderwets, je kunt veel zeggen met anekdotes, voorbeelden van scripties, en foto's en video's van leuke projecten.
Dit project speelt in op deze observaties door een platform te bieden voor iedereen die bij de opleiding betrokken is om in allerei vormen bij te dragen aan het beeld van de opleiding.

2.5.1. Invulling

Wij vragen jou om een website te maken waar de betrokkenen van de opleidingen deze informatie zo simpel mogelijk kunnen delen. Alles is erop gericht de drempel zo laag mogelijk te houden om iets bij te dragen. Teksten en foto's moeten zonder veel klikken toegevoegd kunnen worden.

- Er is één pagina per opleiding. De structuur moet zo eenvoudig mogelijk, maar het mag wel mooi zijn!
- Alles is gericht op de bijdragen en niet zozeer op profielen en namen van mensen.
- Alle bijdragen kunnen op chronologische volgorde getoond worden, maar het moet mogelijk zijn de volgorde te beïnvloeden door bijdragen te "liken" en te "disliken".
- Er moet een homepage komen om de opleidingen te linken, maar dat wordt niet de belangrijkste pagina. De opleidingspagina's zullen uiteindelijk vooral via Google gevonden worden, denken wij, dus focus vooral op een perfecte implementatie van de opleidingspagina.
- Video's hoef je niet per se te verwerken, maar je mag links accepteren naar externe videosites en deze video's automatisch embedden.
- Maak optioneel een suggestiesysteem: link vanaf een opleiding naar gerelateerde opleidingen (bijvoorbeeld door gebruik te maken van gemeenschappelijke ingangseisen of categorie).
- Voorbeelden van bijdragen: video's, foto's van projecten, pdf's van scripties, korte teksten (tweet-size 160 tekens), langere teksten (tot zo'n 500 woorden?)

2.5.2. Gebruik van HODEX

HODEX is een open standaard voor opleiding informatie waar alle universiteiten en een aantal hogescholen aan mee doen. Naast algemene informatie bevat het ook informatie over ingangseisen en een agenda van open dagen. Er is een centraal XML-bestand waarin alle individuele instellingen gelinkt worden; die publiceren op hun beurt een XML-bestand met alle relevante opleidingsinformatie.

- HODEX Index http://www.hodex.nl/hodexDirectory.xml
- HODEX Wiki http://nl.wikipedia.org/wiki/Hodex
- HODEX Website http://www.hodex.nl
  (Hier staat onder andere ook de technische specificatie)

Dit is een perfect startpunt voor deze website omdat je niet hoeft te wachten tot je gebruikers met de hand hun eigen opleidingen hebben aangemaakt. Ook staat er wat korte informatie in de HODEX waarmee je de pagina van een opleiding al wat invulling kan geven. Je moet de HODEX zo gebruiken dat de beheerder van de website de opleidingspagina's kan aanmaken met een druk op de knop. De beheerder moet ook later een update kunnen doen waarmee nieuwe opleidingen in de HODEX worden toegevoegd. Met opleidingen die verdwijnen hoef je nu geen rekening te houden.

2.5.3. Authenticatie: UvaNetID

Het is altijd lastig om mensen zover te krijgen te registreren voor jouw website. Uiteindelijk willen we bij deze website bijdragen toestaan van alle medewerkers en studenten van alle universiteiten, maar voor de UvA hebben we een mooi middel om de drempel te verlagen. Via het CAS-protocol kunnen mensen van de UvA inloggen op jouw website zonder zelf een account aan te moeten maken. Ze worden tijdens het inloggen doorgestuurd naar de UvA-authenticatie waar ze inloggen met hun UvAnetID.

- CAS http://en.wikipedia.org/wiki/Central_Authentication_Service
- Een document met specifieke instructies voor het gebruik van de UvA CAS faciliteit wordt op verzoek beschikbaar gemaakt.
- Voor het gebruik van de UvA CAS faciliteit is het noodzakelijk dat alle groepsleden hun UvANetID aanleveren bij het Informatiserings Centrum (IC).

Als de website goed werkend wordt opgeleverd kan deze direct in productie worden genomen. We raden dan ook aan de site zo snel mogelijk door andere studenten van het vak te laten testen. Omdat we deze website graag echt willen gebruiken en doorontwikkelen moet de code onder de MIT-licentie worden opgeleverd.

2.6. Een interactieve leeromgeving voor videocolleges (moeilijke opdracht)

Voor de minor “Programmeren” gaan we gebruik maken van onder andere de cursus CS50 van Harvard (http://cs50.tv/2011/fall/). We gaan studenten ter plaatste begeleiding bieden, maar ze zullen ongetwijfeld een eigen tempo oppikken, omdat ze meer of minder ervaring hebben. Het doel van deze leeromgeving is om studenten de mogelijkheid te geven zelfstandig een serie videocolleges en extra materiaal door te werken, maar om toch ook bij te kunnen dragen aan het leerproces van de groep: alle studenten kunnen vragen stellen en materiaal toevoegen zoals filmpjes van YouTube, PDFs, links enzovoort.
Jouw doel is het maken van een web app waarmee de docent razendsnel een leeromgeving voor een nieuw vak kan opzetten. De omgeving is gebaseerd op secties met daarin een reeks *afleveringen* of andere onderdelen (voor de site hierboven zijn dat bijvoorbeeld Lectures, Sections, Problem Sets enz). Elk los onderdeel heeft een eigen pagina. Daarop staat het materiaal dat geïmporteerd wordt. Daarnaast is dat de plek waar studenten materiaal en vragen kunnen toevoegen.

Importeren

- importfunctie voor XML http://cs50.tv/2011/fall/?output=xbel

- dit formaat is geen standaard, maar de importmodule mag hier wel op afgestemd worden.

Navigatie
- navigatie om snel bij alle onderdelen van de cursus te komen,
- alle onderdelen hebben afleveringen: dat kunnen “weken” zijn of “tentamens”,
- elke aflevering heeft een eigen pagina met bijbehorend materiaal,
- optioneel: per onderdeel onthouden bij welke aflevering de ingelogde student is gebleven.

Materialen en vragen toevoegen
- toevoegen van mededelingen door docent bij een aflevering,
- toevoegen door studenten van links, comments, youtube-video’s (embedden!), PDFs,
- mogelijkheid tot beantwoorden van vragen door docenten en studenten,
- mogelijkheid tot verwijderen van bijdragen door docenten.

Algemeen
- niet-ingelogde mensen krijgen niet de toegevoegde materialen te zien ivm privacy,
- inloggen implementeren door middel van CAS, zodat registratie niet nodig is,
- elke installatie van de web app is geschikt voor 1 cursus,
- mogelijkheid voor exporteren van zo’n zelfde XML-bestand,
- code inleveren onder licentie CC-BY-SA 3.0 NL.

2.7. Groepskalender (moeilijke opdracht)

Het doel van deze groepskalender is om groepen een uitwisselingsmogelijkheid voor evenementen te geven. Een belangrijke mogelijkheid is dat groepen elkaars evenementen kunnen bekijken via de eigen kalender. Voor het Instituut voor Informatica willen we graag een site die twee ideeën combineert: extern uitdragen welke evenementen door het instituut worden georganiseerd, maar ook intern laten zien naar welke evenementen collega’s gaan. Zo kunnen onderzoekers aangeven naar welke evenementen ze gaan, en of ze presenteren, organiseren, of deelnemen. Vaak zijn dit externe evenementen, die niet door het instituut zelf worden georganiseerd.
Maar ook het uitdragen van de eigen evenementen is belangrijk. Zie bijvoorbeeld http://www.meetup.com/Appsterdam/ waar alle evenementen van Appsterdam staan.
Jouw doel is een website te maken waarop deze informatie verzameld wordt. Leden van de groep kunnen met heel weinig moeite nieuwe evenementen toevoegen of aangeven waar ze heen gaan. De evenementen worden gepubliceerd via een publieke site, via een widget en via een iCal-feed. Daarnaast kan de groep zich abonneren op iCal-feeds van andere groepen.

Functionaliteit voor de leden van de groep
- bekijken van evenementen die door de groep georganiseerd worden,
- bekijken van evenementen waar leden van de groep naartoe gaan,
- toevoegen van een nieuw evenement (naam, startdatum en url),
- aangeven deelnemer, organisatie of bezoeker,
- bekijken van evenementen van andere groepen,
- toevoegen van een feed van een andere groep (iCal),
- bekijken van geabonneerde feeds,
- kopiëren van een evenement uit iCal-feed naar groepskalender.

Functionaliteit voor buitenstaanders

- bekijken van evenementen die door de groep georganiseerd worden,
- bekijken van evenementen waar leden van de groep naartoe gaan,
- iCal-feed en RSS-feed voor beide lijsten,
- widget voor beide lijsten.

Technische eisen

- inloggen implementeren door middel van CAS, registratie dan niet nodig,
- systeem direct al geschikt voor meerdere groepen (IvI + bijvoorbeeld studievereniging),
- iCal-feeds moeten gecached worden en regelmatig gecheckt op nieuwe evenementen,
- geen rechtensysteem: iedereen die kan inloggen wordt geacht lid te zijn van alle groepen (kan optionele feature zijn),
- code inleveren onder licentie CC-BY-SA 3.0 NL.

3. Inleveren

Door het gecompliceerde samenspel van twee vakken, Webprogramming en Databases en Tutoraat/Academische Vaardigheden, valt er in de laatste week van Webprogramming en Databases heel wat in te leveren. Alles dat moet worden ingeleverd gebeurt via Blackboard door middel van één zipfile met alle bestanden. Hier een overzicht:

3.1. De webtoepassing

De webtoepassing is een map in de hoofd-directory van de zipfile met daarin alle benodigde bestanden, zodanig dat ik een werkende applicatie kan krijgen op een webserver. Let daarbij op het volgende:

- Vergeet niet de database bij te voegen; struktuur én inhoud! Een database kan naar een tekstbestand geschreven worden met phpMyAdmin of de MySQL tool "mysqldump".
- Zet in de hoofd-directory een tekstbestand "README.txt" met daarin:
    - een URL naar de werkende applicatie op websec.science.uva.nl (of een URL dat redirect naar websec.science.uva.nl),
    - (indien nodig:) gebruikersnaam en wachtwoord om met zo hoog mogelijke rechten te kunnen inloggen op de werkende applicatie,
    - (indien van toepassing:) bronvermeldingen van niet-eigen code.

De map in de zipfile is nodig om jullie code te kunnen nakijken. De URL is nodig om de toepassing te kunnen testen op de server waar jullie deze zelf getest hebben.

3.2. Verslag

Zet in de hoofd-directory van de zipfile een PDF versie van het verslag. Het verslag bestaat uit twee delen. In het eerste deel worden ontwerpkeuzen toegelicht voor de klant (vertel waarom je produkt zo goed is). Het tweede deel is geschreven voor de ICT-afdeling van de klant. Dit vertelt hoe de site technisch is gerealiseerd, en dient als installatie- en onderhoudshandleiding.
Documentatie op details (afzonderlijke functies) mag in de code. Dat hoeft dus niet in het verslag! Spreek af hoe het schrijfwerk verdeeld wordt en wie de credits ervoor krijgt (krijgen).

3.3. Presentatie

Zet in de hoofd-directory van de zipfile een PDF versie van de presentatie. De presentatie wordt gehouden door *alle* groepsleden: elk lid presenteert een deel. Tijdens de presentatie worden ontwerpkeuzen toegelicht voor de klant. Het gaat om keuzen m.b.t. de functionaliteit en de look-and-feel. Plaats en tijd: zie de mededelingen op Blackboard. Vanuit Tutoraat/Vaardigheden wordt nog een afspraak gemaakt in week 3 om de presentatie voor te bereiden.

3.4. Bewijs van samenwerking

Zet in de README.txt die je in de hoofd-directory van de zipfile zet de samenstelling (namen en studentnummers) van je projectgroep. Dit telt als bewijs dat je hebt samengewerkt.

Wij sturen een beoordeling van de samenwerking, op basis van onze indruk of er gewerkt is, en of dat werk eerlijk verdeeld was.
Als het een weekje niet goed gaat met de samenwerking: vertel het ons! Dit wordt nooit negatief aangerekend. Wat telt is of aan het eind het werk goed verdeeld is geweest.

3.5. Beoordeling

De eindbeoordeling van dit vak wordt individueel bepaald door middel van het gemiddelde cijfer van de volgende beoordelingen:

1. Het eindproduct: de website en het verslag. Daarbij letten we op: werkt de website volgens de specificatie, is er zorg besteed aan uiterlijk en interactie, is er voldoende bescherming tegen misbruik, is het verslag goed gestructureerd, duidelijk geformuleerd, is de code goed gestructureerd en gedocumenteerd. Voor het verslag worden twee cijfers gegeven: een voor de inhoud (telt bij Webprogramming en Databases) en een voor de vorm (telt bij Vaardigheden).
2. Je functioneren in teamverband; daarbij letten we op ondernemingszin (eigen initiatief, aanpakken van problemen, sturen van het project), organisatie (taakverdeling, planning), werkhouding (aanwezigheid, punctualiteit, inzet) en samenwerking (verantwoordelijkheid, werksfeer).
3. Je presentatie in de laatste week. Daarbij letten we op: inhoud (correct en afgestemd op publiek), structuur (opbouw en tijdsverdeling) en vorm (houding, gebruik van beschikbare techniek, taalgebruik). Ook voor dit werk worden twee cijfers gegeven, net als bij het verslag.

De beoordeling wordt gedaan door een docent en een studentassistent.

Inhoudsopgave

Oefeningen

De hoofdstukken in deze syllabus bestaan uit tekst afgewisseld met oefeningen. De oefeningen zijn bedoeld om kennis te maken met de stof. Ze zijn niet verplicht en worden niet ingeleverd.

- Oefeningen zijn te herkennen aan de vette zwarte punt.

1. HTML en XHTML

Om het world wide web tot een succes te laten worden waren drie dingen nodig:
1) Een standaard protocol voor overdracht van teksten: HTTP;
2) Een standaard taal voor beschrijving van documentstructuur: HTML;
3) Een methode om documenten vorm te geven: tegenwoordig is dit CSS (Cascading Style Sheets); een taal voor stylesheets.

Van overdracht van teksten hoef je weinig te weten als je webtoepassingen maakt. HTTP is meer een onderwerp voor mensen die zich verdiepen in netwerktechniek. Je kunt de HTTP-specificatie vinden op:

ftp://ftp.rfc-editor.org/in-notes/rfc2068.txt

HTML (HyperText Markup Language) is de taal waarin webdocumenten hun eigen structuur beschrijven. Met HTML geef je structuur aan een tekst die bestemd is voor het web. Je geeft bijvoorbeeld aan wat een kopje is en wat een alinea is. Of je geeft aan dat iets een tabel is. Dit gebeurt met markeringen of in het engels *tags*. Zulke tags kunnen in paren voorkomen: een begintag en een eindtag, bijvoorbeeld zo:

```
<h1>Hoofdstuk 3: de motor van de auto</h1>
```

Een paar tags met de inhoud die er tussen staat heet tezamen een *element*, dus hierboven zie je een `h1`-element. Met het `h1`-element geef je aan dat de tekst die ertussen staat een kop (“heading”) is van het eerste (hoogste) niveau.
Hieronder zie je een gecompliceerder element. Dit stelt een hyperlink voor.

```
<a href="http://www.science.uva.nl/" title="klik hier">
homepagina van de faculteit</a>
```

We zien de *begintag*: `<a href="..." title="...">`,
en de *eindtag:* `</a>`.
Bij elkaar is dit een a-element (a voor "anchor", anker).
De begintag heeft twee *attributen*: `href` en `title`.
`"http://www.science.uva.nl/"` is de waarde die aan het href-attribuut wordt toegekend. Waarden staan tussen aanhalingstekens.

De inhoud van dit element, "homepagina van de faculteit" is de tekst waar de webbezoeker op klikt. Hij krijgt dan de webpagina te zien, die zich bevindt op het adres "http://www.science.uva.nl/". Als zijn muis boven de link zweeft ziet hij een geel vlekje verschijnen met de tekst "klik hier", een zogeheten tooltip.

Enige tijd geleden vervulde HTML twee functies: het diende zowel om documenten vorm te geven als om de structuur vast te leggen. Sinds HTML 4.0 zijn deze functies gescheiden.
- Structuur wil zeggen: voor elk stukje tekst de functie vastleggen: is het een kopje, een alinea, een tabelcel, een hyperlink, ...?
- Vormgeving wil zeggen: hoe groot moet het kopje zijn, welk lettertype, welke kleur, moet het inspringen of niet?

Voor veelvoorkomende functies zijn elementen gedefinieerd, met elk een naam (zoals `a` of `h1`) en met geschikte attributen die verplicht (`href`) of optioneel kunnen zijn (`title`).
Als je in je tekst structuurelementen hebt waarvoor nog geen element is voorgedefinieerd, dan kun je het generieke `div` element nemen en dit voorzien van een *class*-attribuut. Hiermee maak je in wezen zelf je eigen tags.

Vormgeving doe je door in CSS enige stijlregels te schrijven. Hiermee vertel je voor elke soort tekst (elke tekstfunctie) hoe deze soort tekst er uit moet zien. Je vertelt dus hoe kopjes, hyperinks en tabellen er uit moeten zien. Met CSS is het weinig werk om een consistente opmaak te krijgen! Als er voor elke soort tekst maar één regel is, dan is de opmaak vanzelf consistent.

In dit hoofdstuk bekijken we HTML, de taal waarmee je je document indeelt. In het volgende hoofdstuk bekijken we CSS, waarmee je het ingedeelde document vormgeeft.

1.1. Ontwikkelingen in de HTML-taal

XHTML 1.1 (Extensible Hypertext Markup Language) is de huidige versie van de HTML-taal. Daarvoor hadden we XHTML 1.0 en daarvoor HTML 4.01. De nieuwste versie, die op het moment van schrijven nog niet tot standaard is verheven, gaat HTML5 heten. In de nummering en naamgeving van de versies zit een merkwaardige sprong, zoals je ziet.

HTML behoort tot een grote familie van talen die SGML (“Standard Generalized Markup Language”) wordt genoemd. Alle SGML talen werken met *tags* maar deze hoeven niet per sé in paren voor te komen. In HTML 3 kon je bijvoorbeeld typen:

```
<h1>Dit is een hoofdstuktitel
<p>Dit is een nieuwe alinea en hoort niet meer bij de titel
```

En dat was toen goed. In XHTML is dit echter geheel fout!
Omdat de SGML taalfamilie allerlei complicaties heeft en bekend staat als erg moeilijk en foutgevoelig, heeft men sinds ongeveer 1990 een strengere familie gevormd, de XML-taalfamilie. Dit is een deel van de SGML familie.
In XML hoort bij elke begintag een eindtag, *behalve* voor elementen die van nature leeg zijn, zoals bv. het linebreak element. Voor zulke lege elementen bestaat een aparte syntax in XML: in het geval van de linebreak: "
". Dus dit zijn de twee mogelijkheden in XML:

Element met inhoud:

```
<naam attribuut1 attribuut2 ...>inhoud</naam>
```

Element van een type dat nooit inhoud heeft:

```
<naam attribuut1 attribuut2 ... />
```

Let op de geniepige slash die aan het einde staat. Hiermee wordt het element in wezen toch nog beëindigd, hoewel er geen aparte eindtag is.
Tot en met versie HTML 4.01 behoorde HTML tot de SGML-familie en daarna zijn we overgestapt op de strengere XML-familie. Deze stap vond men zo belangrijk, dat men opnieuw begonnen is met de nummering. Echter de inhoud van de taal is met die stap

nauwelijks veranderd. Vóór de overstap had je al `h1` en `a` en `p`-elementen en na de overstap heb je die nog steeds. De XHTML 1.0 specificatie is dan ook heel kort, omdat voor de omschrijving van de elementen verwezen wordt naar de HTML 4.01 spec.

Momenteel wordt hard gewerkt aan een voorstel voor HTML5. HTML5 omvat niet alleen een nieuwe HTML-standaard maar ook een scala van technieken die allemaal meegenomen worden in de definitie, waaronder CSS3, Javascript, XML, JSON, SVG en ondersteuning voor video en audio. De voornaamste reden om al deze technieken deel te laten uitmaken van de definitie is om de implementatie van interactieve webapplicaties makkelijker te maken. Vooralsnog is het de bedoeling HTML5 pas in 2014 als “aanbevolen standaard” te lanceren. Het is nu al echter goed mogelijk met HTML5 te experimenteren met diverse browser implementaties. Binnen dit document beperken wij ons echter tot XHTML 1.0 en 1.1.

Hier is de complete lijst eisen waaraan een XHTML document voldoet. Sommige worden hierna nog verklaard:

1) Een XHTML document begint met de xml-versie declaratie:

   ```
   <?xml version="1.0"?>
   ```

   Dan volgt een verwijzing naar de XHTML Document-type-definitions (DTD's). Je kunt in XHTML 1.1 kiezen uit de volledige ("strict") of een "basic" definitie. De laatste wordt vooral gebruikt voor mobiele toepassingen.
   Voorbeeld:

   ```
   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
        "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
   ```

   Of voor de "basic" versie:

   ```
   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.1//EN"
       "http://www.w3.org/TR/xhtml-basic/xhtml-basic11.dtd">
   ```

   In HTML5 gaat dit veranderen in iets overzichtelijks:

   ```
   <!DOCTYPE HTML>
   ```

2) In elk document is er een hoogste element, het `html`-element; dit is de ouder van alle andere. Het `html`-element volgt direct na de <!DOCTYPE...> declaratie.
3) Een `html`-element bevat een `head`-element en een `body`-element. Binnen het `head`-element zit, indien nodig, meta-informatie over de pagina: de titel van het document, keywords, verwijzingen naar stylesheets. Het `body`-element bevat de inhoud van de pagina.
4) In XHTML zijn alle tag-namen en attribuutnamen in *lowercase*. Hoewel compatibel met HTML 4.0 betekende dit een breuk met bestaande gewoonten.
5) In XML moet ieder element expliciet worden beëindigd. Dit gebeurt ofwel met een eindtag zoals `</p>`, ofwel – als een element geen inhoud heeft – door de begintag te voorzien van een sluitsymbool. `<img>` wordt bijvoorbeeld: `<img />` omdat het `img`-element nu eenmaal nooit tekstuele inhoud heeft. Hetzelfde recept geldt voor alle lege elementen. (N.B.: Voor compatibiliteit met oudere browsers is een spatie vóór de / essentieel!)
6) In XML heeft ieder element ondubbelzinnig één ouder.
   Dus constructies zoals

```
<em><strong>deze zin is erg nadrukkelijk</em></strong>
```

zijn niet toegestaan. Het hele `strong`-element moet zich binnen het `em`-element bevinden, of andersom! Oftewel, het document moet in strikte zin een boomstructuur hebben. Dit is nodig om elementen goed te kunnen adresseren in stylesheets of scripts.

7) Attribuut-waarden staan in XML tussen dubbele aanhalingstekens:

```
<base target="_blank">
```

8) Attributen hebben altijd een waarde. In HTML 4 was dit nog toegestaan:

```
<option value=1 checked>
```

In XHTML wordt dit:

```
<option value="1" checked="checked">
```

9) Er is een lijst met beschikbare elementsoorten: `h1`, `h2`, `h3`,... `a`, `p`, `div`, ...
10) Bij elke elementsoort is bekend welke attributen er kunnen voorkomen.
11) Bij elke elementsoort is bekend welke elementen daarbinnen kunnen voorkomen.

Voor de laatste drie eisen verwijzen we naar de naslagwerken op onze website. De voornaamste elementsoorten en hun attrbuten zul je echter gauw tegenkomen.

1.2. De boomstructuur van HTML

HTML-elementen kunnen genest worden. Dit betekent dat je in de *inhoud* van een HTML-element tekst kunt opnemen, die zelf ook weer HTML-elementen kan bevatten. Voorbeeld:

```
<em>Dit is nadrukkelijk <strong>en dit is erg
nadrukkelijk</strong></em>
```

Sommige van deze nestings zijn vrij, andere zijn verplicht. Bijvoorbeeld een tabel wordt altijd opgebouwd uit `table`, `tr` en `td`-elementen in een vaste volgorde van nesting:

```
<table>
      <tr>
            <td>Naam:</td><td>Jan</td><td>Piet</td>
      </tr>
      <tr>
            <td>Cijfer:</td><td>9</td><td>10</td>
      </tr>
</table>
```

Spaties, regeleindes en inspringen hebben geen betekenis in HTML. Je mag ze vrij toevoegen om de leesbaarheid te vergroten. Dat heb ik hier dan ook gedaan.
Hierboven zie je een `table` element, dat twee `tr`-elementen bevat. Elk `tr`-element bevat weer enkele `td`-elementen. De `tr`-elementen stellen de *rijen* van de tabel voor, de `td`-

elementen zijn de afzonderlijke *datacellen* in elke rij. De getoonde volgorde van nesten is verplicht, dus:

- `table` bevat `tr`
- `tr` bevat `td`

Er zijn zeer veel van zulke regels, de meeste zijn zo logisch dat je ze niet hoeft te onthouden, andere zoek je op zodra je ze nodig hebt. Bij twijfel is de W3C specificatie van XHTML 1.0 of 1.1 doorslaggevend: zie de literatuurlijst in sectie 1.6.

Door elementen te nesten ontstaat een boomstructuur. Ieder element heeft één *ouder*, dat is het kleinste element waar het zich geheel binnen bevindt. Bijvoorbeeld alle `td`-elementen hierboven hebben een `tr` als ouder. Daarentegen kan een element meerdere *kinderen* hebben, dat zijn de elementen die zich er direct binnen bevinden. De terminologie van ouders, kinderen (en zelfs zusjes, grootouders, etc.) kom je tegen als je elementen gaat selecteren via CSS of in het Document Object Model (DOM).

Een boomstructuur is pas goed als er één hoogste knoop is,en dat is bij XHTML het `html`-element. Direct daaronder hangt ook een verplichte structuur die als volgt in elkaar zit:

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html>
      <head>
            <title>Titel die verschijnt in de titelbalk van je
                  browser</title>
            <meta http-equiv="Content-Type" content="text/html;
                  charset=iso-8859-1" />
            <meta name="keyword" content="Amsterdam" />
            <link rel="stylesheet" type="text/css"
                  href="mijnstylesheet.css" />
      </head>
      <body>
            ...Hier de inhoud van je pagina...
      </body>
</html>
```

Alle hier getoonde elementen, behalve `meta name` en `link` zijn verplicht.
Het document wordt meteen in twee delen verdeeld:

- de header of `head`: hier komt meta-informatie die op het hele document slaat, bijvoorbeeld de titel (verplicht), trefwoorden die het document beschrijven, en een stylesheet voor het document.
- de `body` van het document: Hier komt de tekst van je pagina, gemarkeerd met elementen zoals `p`, `a`, `table`, `div`, ...

De `xml`- en !DOCTYPE-declaraties vallen buiten de boomstructuur van het document. Je ziet dat deze declaraties ook een afwijkende syntax hebben, het zijn geen gewone HTML-elementen.

1.3. Oefeningen

De opdrachten in deze tekst zijn bedoeld om tijdens de colleges te doen. Deze zijn meestal kort en bedoeld om je op weg te helpen met de techniek. We beginnen met wat standaardinstellingen die handig zijn tijdens de colleges.

1.3.1. Standaardinstellingen maken.

- Voor deze cursus is een webserver ingericht: websec.science.uva.nl
  Open een browser naar dit adres. Er wordt je gevraagd om een username en password: dat is respectievelijk "webdb" en "webdb2012". Na inloggen verschijnt een pagina met welkombericht en nuttige informatie.
- Inloggen op de webserver doe je met SSH. Als je een Linux of Mac hebt, dan is een SSH client al aanwezig. Download voor Windows bijvoorbeeld PuTTY: zie de link op websec.science.uva.nl.
- Log met een SSH client in op websec.science.uva.nl met je science account inlognaam en password.
- Maak in je home directory een nieuwe map:
  `mkdir public_html`
- Typ het volgende commando om de rechten goed te zetten:
  `chmod o+rx public_html`
- Als je in de map "`public_html`" een bestand "`index.html`" plaatst dan wordt dit zichtbaar voor de buitenwereld op het adres:
  `http://websec.science.uva.nl/~jeinlognaam`
  Typ bijvoorbeeld:
  `echo "Hallo" > ~/public_html/index.html`
  Het is dus een persoonlijke, openbare homepagina. Op deze pagina kun je hyperlinks maken naar andere pagina's die je openbaar wilt maken (niet verplicht).
- Maak eventueel werkmappen voor deze cursus onder "`public_html`".
  Werkstukken die nog niet openbaar zijn kun je hier rustig laten staan, zolang je er geen hyperlink naartoe maakt.

Met een "`.htaccess`" bestand kun je een map grondig beschermen tegen inkijk vanuit internet. Dit heeft overigens geen haast! Dit bestand kan bijvoorbeeld de volgende inhoud hebben:

```
AuthUserFile /home/jeinlognaam/filenaam
AuthName sitenaam
AuthType Basic
require valid-user
```

De `AuthUserFile` regel wijst de weg naar een bestand met inlognamen en bijbehorende wachtwoorden. Met het commando "`htpasswd`" kun je zo'n wachtwoordenbestand maken en aanvullen:

```
htpasswd –c /home/jeinlognaam/filenaam inlognaam
```

Met "-c" maak je een nieuw wachtwoordenbestand. Daarna, als je een tweede gebruiker (inlognaam) aan het bestand wilt toevoegen, laat je de "-c" weg.

Met `AuthName` geef je een naam aan het beveiligde gebied. Als je meerdere mappen met dezelfde *sitenaam* beveiligt, hoeft de gebruiker maar éénmaal in te loggen om toegang te krijgen tot al deze mappen.

We komen later nog terug op beveiliging van websites met wachtwoorden.

1.3.2. HTML maken en het effect uitproberen.

- Kopieer "`leeg.html`" uit de map `/var/www/html/voorbeeldcode/HTML+CSS` naar je `public_html` map. Dit is een leeg html document, het bevat alleen het verplichte geraamte.
- Probeer het effect van een paar tags uit in de "body" van het document.
    - Maak bijvoorbeeld een hyperlink naar www.science.uva.nl met hulp van een `a`-element
- Bekijk het document in een browser om het effect te zien.
- Probeer nog een paar dingen en bekijk steeds het effect:
    - Voeg een `title`-attribuut toe aan de hyperlink of aan een `p`-element (dus vb. `title="iets"`). Wat is het effect als je met de muis boven de link of de alinea zweeft?
    - Wat is het effect van `target="_blank"` in een `a`-element?
    - Hoe maak je een lijstje met bullets (vette punten) voor elk van de items?
    - Hoe maak je een tabel met 2 rijen en 3 kolommen?
    - Wat denk je met `<meta name="keywords" ... >` te kunnen bereiken?

Meer ideeën vind je op http://www.w3schools.com/tags/

1.3.3. De boomstructuur van HTML benutten – voorbeeld: stylesheets.

- Open de eerder gemaakte pagina in een editor. Gebruik bij voorkeur een editor die "syntax highlighting" ondersteunt zodat deze als spell-checker kan fungeren. Een voorbeeld is "vim": deze editor kent behalve HTML ook CSS, Javascript en nog veel meer.
- Een stylesheet voor één document maak je door in de `head` van dat document een paar tags `<style>` en `</style>` te plaatsen, met daartussen een of meer CSS-regels. Zo:

```
<style type"text/css">
CSS-regels
</style>
```

- Een CSS-regel die alle `table`-elementen adresseert en de bijbehorende tabellen een rode achtergrondkleur geeft ziet er zo uit:

```
table { background : Red; }
```

- Check het resultaat in verschillende browsers, zoals Internet explorer en/of Mozilla Firefox.
- Net als `table` kun je ook alle andere HTML-elementen (zoals `body`, `p`, `tr`, `td`, `a`) adresseren. Probeer het effect.

- Zo kun je selectief elementen adresseren die zich "binnen" een genoemd ander element bevinden:

```
table  td  { background : Green }
```

  (Niet erg nuttig! Immers álle `td` elementen zitten binnen een `table` element).
- Geef zelf de hyperlink die zich in de tabel bevindt een groene achtergrondkleur. Doe het zo dat hyperlinks buiten de tabel niet beinvloedt worden. (Verifieer dit)
- Als in de stylesheet twee regels staan, één voor het `body`-element en één voor een element dat zich binnen het `body` element bevindt, welke regel heeft dan voorrang in geval van een conflict? (De uitkomst geldt algemeen voor elementen binnen elementen)

De boomstructuur van HTML wordt nog interessanter als je in stylesheets met "class" selectors werkt of als je via javascript de tekstuele inhoud van geselecteerde elementen gaat wijzigen via het HTML Document Object Model (HTML-DOM). Meer hierover later.

1.3.4. Finetuning – een tabel die zich (al dan niet) aanpast aan het venster.

Om een webpagina in kolommen te verdelen worden soms tabellen gebruikt. Dit is volgens XHTML toelaatbaar als het om een "structurele" indeling van de pagina gaat, zoals in het voorbeeld hieronder. Tabellen mogen dus niet meer gebruikt worden om plaatjes of tekst alleen maar op hun plaats te zetten.

Hieronder zie je twee tabellen:
1) Bij de eerste is de bedoeling dat de tabel het beschikbare venster volledig vult, waarbij de verhouding tussen de kolommen 50/50 moet blijven.
2) Bij de tweede is het gewenst dat de eerste kolom een vaste breedte krijgt en de tweede zich aanpast om het venster maximaal te benutten.

Enerzijds	Anderzijds
Blabla bla bla bla bla bla bla bla bla bla blabla bla bla bla blabla bla bla bla blabla bla bla bla blabla bla bla bla blabla bla bla bla bla. Blabla bla bla bla bla bla bla bla bla bla blabla bla bla bla blabla bla bla bla blabla bla bla bla blabla bla bla bla blabla bla bla bla bla. Blabla bla bla bla bla bla bla bla bla bla blabla bla bla bla blabla bla bla bla blabla bla bla bla blabla bla bla bla blabla bla bla bla bla.	Blabla bla bla bla bla bla bla bla bla bla blabla bla bla bla blabla bla bla bla blabla bla bla bla blabla bla bla bla blabla bla bla bla bla. Blabla bla bla bla bla bla bla bla bla bla blabla bla bla bla blabla bla bla bla blabla bla bla bla blabla bla bla bla blabla bla bla bla bla. Blabla bla bla bla bla bla bla bla bla bla blabla bla bla bla blabla bla bla bla blabla bla bla bla blabla bla bla bla blabla bla bla bla bla.

Tabel 1: gelijke breedte, venstervullend

Jan	Blabla bla bla bla bla bla bla bla bla bla blabla bla bla bla blabla bla bla bla blabla bla bla bla blabla bla bla.
Piet	Blabla bla bla bla bla bla bla bla bla bla blabla bla bla bla blabla bla bla bla blabla bla bla bla blabla bla bla.
Klaas	Blabla bla bla bla bla bla bla bla bla bla blabla bla bla bla blabla bla bla bla blabla bla bla bla blabla bla bla.

Tabel 2: eerste kolom vaste breedte, venstervullend

De elementen `table` en `col` kunnen een `width` attribuut hebben.[1]
Als waarde kun je opgeven:

– een getal (`width="600"`), dit stelt een aantal pixels voor,
– een percentage (`width="20%"`) van de breedte van het venster,
– of vrij (`width="*"`), deze laatste werkt alleen bij `col`. In plaats van deze waarde kun je het width-attribuut ook weglaten[2].

Als zowel bij `table` als bij alle kolommen getallen zijn opgegeven, moet de optelling kloppen, anders is het resultaat onvoorspelbaar. Hetzelfde geldt voor percentages. In één tabel kun je "*" combineren met vaste waarden of percentages. Vaste waarden en percentages door elkaar gebruiken voor kolommen in één tabel leidt tot fouten (waarom?).

- Kopieer de bestandjes tabel1.html en tabel2.html,
- Corrigeer tabel1.htm zo dat het bestand zich aanpast aan het beschikbare venster.
- Corrigeer tabel2.htm zo dat de linkerkolom een geschikte vaste breedte krijgt terwijl de rechterkolom zich aanpast (lees eventueel eerst de tekst hieronder).

Het `col` element kun je gebruiken om te voorkomen dat je voor alle cellen in de tabel apart een breedte moet instellen. Het `col` element bestaat sinds HTML 4.0 en wordt niet herkend door oudere browsers (Netscape 4.7). Het werkt zo:

```
<table attributen>
      <col width="waarde" /><col width="waarde" />...
      <tr>
            <td>celinhoud</td><td>celinhoud</td>...
      </tr>
      ...
</table>
```

Volgens de HTML 4.0 norm mag "*" gecombineerd worden met "2*". In dat geval wordt de overblijvende vrije ruimte in de tabel verdeeld in porties die zich verhouden as 1 : 2, of algemener *n* : *m* als je "*n**" en "*m**" opgeeft. Dit werkt echter niet in alle browsers (zoals Internet Explorer 6).

[1] Het gebruik van het `width` attribuut bij een `td` element is afgekeurd in HTML 4.0, ten gunste van `width` bij het `col` element (zie volgende blz).
[2] Vrij ("*") wordt door Internet Explorer anders opgevat dan door Firefox. Firefox vult de hele beschikbare ruimte (alles wat overblijft na de andere kolommen). Internet Explorer neemt de ruimte die de inhoud nodig heeft. Alleen bij een tekst van meer regels neemt hij de hele beschikbare ruimte.

1.4. Frames en waarom ze niet deugen...

Frames zijn delen van het Browser-venster waarin verschillende documenten geladen worden die onafhankelijk van elkaar gescrold kunnen worden door de gebruiker.
De gewenste indeling van het scherm wordt vastgelegd in een frameset-document, waarin verwijzingen staan naar de documenten die in de diverse frames moeten verschijnen. (Voor details, zie bijv. de XHTML-referentie van w3schools.com, onder `<frameset>`) .

Frames hebben een paar voordelen en veel nadelen. Het voordeel is dat je een banner of navigatie-elementen permanent in beeld kunt houden, terwijl de gebruiker scrollt. Bij gewone software zijn gebruikers gewend dat de menubalk altijd in beeld blijft, en daar zijn goede redenen voor. Het is nuttig om datzelfde effect ook bij webtoepassingen te kunnen bereiken. Maar: frames zijn destijds te vroeg geïntroduceerd. Ze pasten niet binnen de toenmalige opzet van het web en hebben daardoor nog steeds een paar belangrijke nadelen:

- Als iemand over het web surft, ziet hij in zijn adresbalk steeds de huidige locatie. Mensen zijn gewend deze locatie te kunnen kopieren en bijvoorbeeld in een email aan iemand toesturen. Op een frames-site geeft de adresbalk echter het adres van het frameset-document aan. Als je binnen de frameset op hyperlinks klikt, verandert het adres in de adresbalk meestal niet. Als je dit adres aan iemand stuurt, komt hij waarschijnlijk naar de homepagina, niet naar de pagina die je wilde sturen.
- Om dezelfde reden kun je een toestand van een frameset niet bookmarken. Als je op een frames-site surft, en je maakt dan een bookmark, dan brengt die bookmark je niet naar de pagina die je op dat moment bekeek, maar naar de begintoestand van de frameset.
- Via zoekmachines, zoals Google, komen bezoekers altijd op gewone pagina's terecht. De frameset pagina bevat geen tekst en wordt dus via zoekmachines niet gevonden. Dit betekent dat een bezoeker de eerste pagina zal zien zonder de frameset eromheen. Maar de site is niet ontworpen om zo gezien te worden!
  Om dit op te lossen zou je op elke afzonderlijke pagina een "home" hyperlink naar de frameset moeten plaatsen, zodat een bezoeker die via een zoekmachine binnenkomt, snel de nodige navigatie-hulpmiddelen krijgt.
- Als je een framespagina resizable maakt, kunnen er bij een resize extra scrollbars verschijnen op plaatsen waar dit niet fraai staat. In je code kun je die scrollbars verbieden, maar dan kan er dus tekst wegvallen zonder dat de gebruiker dit opmerkt.

Er zijn tegenwoordig goede alternatieven voor HTML-frames. Een paar demootjes van alternatieven vind je in “/var/www/html/voorbeeldcode/HTML+CSS/CSS-frames. De beste oplossing gebruikt behalve CSS ook PHP, waar we in hoofdstuk 4 op terugkomen. Als je voor je website gebruik wilt maken van frames dan kun je de code van deze voorbeeldjes alvast inzien en gebruiken.

1.5. Validatie

Je kunt controleren of je document voldoet aan een XHTML specificatie met de validatieservice van het Worldwide Web Consortium (W3C):

http://validator.w3.org/

Absolute voorwaarde voor validatie is de aanwezigheid van een <!DOCTYPE …> declaratie, waarin je aangeeft aan welke specificatie je wilt voldoen, zie hierboven de eisen voor XHTML.

Een andere eis is het specificeren van een “character-encoding” voor het document. Voor gebruik in westeuropa kun je deze tag opnemen in de HEAD van het document:

```
<meta http-equiv="Content-Type" content="text/html;
charset=ISO-8859-1" />
```

Om te voorkomen dat je telkens deze elementen moet intypen kun je het best steeds gebruik maken van “leeg.html” dat je eerder ophaalde van websec, of van een zelfgemaakt leeg document.

2. CSS stylesheets

Een stijlregel in een CSS stylesheet ziet er bijvoorbeeld zo uit:

```
p     { text-align: right; font-variant: small-caps; }
```

Het linkerdeel "p" heet de *selector*. De eenvoudigste selectors zijn gelijk aan elementtype namen. Deze selector selecteert alle `p`-elementen, dus alle alinea's.
Het rechterdeel is de *stijldeclaratie*. Deze staat tussen {}. In deze stijldeclaratie worden twee *eigenschappen* toegekend aan de geselecteerde elementen. Er wordt gezegd dat alle `p`-elementen rechts uitgelijnd zullen worden en dat ze in "small-caps" lettertype gezet worden. "text-align" en "font-variant" zijn dus namen van eigenschappen, "right" en "small-caps" zijn de toegekende waarden.

We gebruiken Topstyle Lite om niet alle eigenschappen uit ons hoofd te hoeven leren dan wel voortdurend te bladeren in een manual.

2.1. Een stylesheet maken met TopStyle Lite.

We gaan een documentje voorzien van een stylesheet en daar wat mee experimenteren. Het documentje is al klaargezet.

- Haal "csstestdoc.html" van /var/www/html/voorbeeldcode/HTML+CSS en open dit bestand in een teksteditor (bv. vim of de HTML-view van Mozilla composer). Bewaar het in je eigen public_html.
- Download en installeer TopStyle Lite: `http://www.akker-huis.nl/topstyle-lite.php`
- Open TopStyle Lite.
- Instellingen: onder View, controleer of "Preview" en "Style inspector" aan staan.
- Op de werkbalk vind je een knop om een nieuwe selector te maken (vierde knop van links, tooltip "New selector"). Druk erop. Kies een HTML-element uit de lijst, bijvoorbeeld `p`. Klik op OK. In de "Style Inspector" verschijnen nu de eigenschappen die van toepassing zijn op `p`-elementen.
- In de "Style Inspector" vind je een keuzelijst waar je "CSS Level 2" kunt kiezen. Resultaat: nu zie je alleen nog maar de eigenschappen van `p` elementen volgens de CSS Level 2 norm[3].
- Kies in de Style Inspector een eigenschap, bijvoorbeeld `background-color` en klik in de kolom rechts van deze eigenschap. Er zijn nu twee mogelijkheden. Als er een pijltje ▼ verschijnt: klik op het pijltje om de mogelijke waarden te zien die deze eigenschap kan krijgen. Maak een keuze. Als er geen pijltje ▼ verschijnt: typ zelf een waarde, of klik op het plusje links van de eigenschap-naam om meer keuzen te krijgen.
- Let op wat er in de linkerhelft van het venster gebeurt: hier is een stijlregel ontstaan. In het Preview venster onderaan zie je een voorbeeld van hoe het resultaat van je stijlregel eruit ziet.
- Je kunt nog meer eigenschappen toevoegen aan het eerder geselecteerde element.

---

[3] Met de andere opties in het menu kan je controleren of je CSS ook op andere versies en oudere browsers ondersteund wordt.

- In de linkerhelft van het venster kun je, als je wilt, zelf typen en editen. Je kunt fouten selecteren en met delete of backspace verwijderen.
- Maak nu de stijlregel die bovenaan deze bladzijde werd geanalyseerd.
- Plaats de cursor onder de zojuist gemaakte stijlregel. Nu kun je een tweede stijlregel gaan opbouwen. Doe dit naar eigen smaak.
- We gaan de opgebouwde stylesheet (die uit twee stijlregels bestaat) kopieren en inplakken in het testdocument: selecteer beide regels en kies Edit > Copy.
- Activeer het venster van je teksteditor (vim bijvoorbeeld). Maak in de `head` van het document ruimte voor de stijlregels, zo:

```
<style type="text/css">
Plak hier de gemaakte stijlregels
</style>
```

- Test het resultaat met verschillende browsers.

Nu we de tools paraat hebben kunnen we experimenteren met verschillende soorten selectors. Daarbij is het soms ook nodig om de structuur van het testdocument wat aan te passen, dus houd je editor paraat met het testdocument geladen. Houd ook de vensters van TopStyle en van je browser(s) open.

We kijken eerst eens naar de verschillende manieren om elementen te selecteren.

2.2. Elementen onderscheiden met "class".

Hierboven kregen alle `p`-elementen dezelfde stijlregel opgelegd. Als dat niet gewenst is kun je zelf elementen indelen door ze een `class` attribuut mee te geven. Dit doe je in de `body` van het document zelf, zo:

```
<p class="zelfbedachtenaam">
```

Geef dezelfde klassenaam aan `p` elementen die een overeenkomstige rol hebben in je document. Geef verschillende namen aan `p` elementen die je wilt onderscheiden.

Tip: Geef namen die de rol van het element omschrijven. Bijvoorbeeld: in een programmeurshandleiding worden vaak "tips" onderscheiden van "waarschuwingen", en van "normale tekst". Het onderscheiden op basis van structuur (niet op basis van vormgeving) is in overeenstemming met de filosofie van HTML, immers een structuurgeoriënteerde taal. Dus dit `p` element zou ik als `class`-naam "tip" geven en niet "grijs". De naam blijft dan ook nog juist als ik voor een andere vormgeving kies. En dat is juist het doel van CSS: dat je makkelijk van vormgeving kunt wisselen.

Hierna kun je in de stylesheet gebruik maken van de gemaakte onderscheidingen. De regel uit het begin van dit hoofdstuk kan dan bijvoorbeeld zo worden:

```
p.classnaam  { text-align: right; font-variant: small-caps; }
```

De klassenaam wordt voorafgegaan door een punt. Het is ook mogelijk om het elementtype weg te laten, als in:

```
.classnaam  { color: blue; }
```

In dat geval geldt de regel voor *alle* elementen in deze class. Als beide bovenstaande stijlregels bestaan, dan gelden de regels in cascade: alle elementen in deze class worden blauw maar een p-element in deze class wordt ook nog small-caps en rechts uitgelijnd.

Je kunt klassenamen geven aan alle soorten elementen die in de boom onder `body` voorkomen, dus ook aan `table`, `img`, `td`, enzovoorts (niet aan die in de `head`).

- Probeer het uit.

2.3. Pseudo-element en pseudo-class selectors

CSS biedt selectors die een deel van een document selecteren waarvoor geen HTML element bestaat. Omdat deze selectors als het ware een nieuw element omschrijven worden ze ook wel *pseudo-elements* genoemd. Hier volgen een aantal voorbeelden:

p:first-letter	de eerste letter in een paragraaf
p:first-line	de eerste regel van een paragraaf
p:first-child	de eerste paragraaf binnen elk element
p:before	element voor elk p-element
p:after	element na elk p-element
p:lang(nl)	elk p-element met lang attribuut waarde "nl"
input:focus	input element dat nu de focus heeft

Let op de dubbele punt[4].

Aan een HTML element kunnen ook kenmerken kleven die de "staat" van het element weergeven. Een link bijvoorbeeld (een `a`-element) kan eerder bezocht zijn ("visited") of de muispointer kan erboven hangen ("hover") of hij kan aangelikt worden ("active"). Deze staat wordt in CSS weergeven door een *pseudo-class*:

a:link	niet-bezochte hyperlink
a:visited	bezochte hyperlink
a:active	link waarop nu geklikt wordt
a:hover	link waar de muis boven zweeft

- Maak een regel voor een `p`-element die de eerste letter van een paragraaf groot maakt en de rest van de eerste regel in small-caps.
- Zorg dat de hyperlinks in het testdocument rood (of groen of pimpelpaars) oplichten als je met de muis erover heen beweegt. Ook aardig: de lettergrootte veranderen (trekt zeer de aandacht, de toeschouwer verwacht het namelijk niet).
- Als je zowel a:hover als a:visited definieert, dan is de volgorde van belang: in geval van een conflict (de één zegt "rood", de ander "blauw") heeft de laatste regel voorrang! Wat is dus de goede volgorde? Beredeneer en test.
- En hoe zou dat zijn voor a:link en a:active?

[4] In CSS3 moet je voor pseudo-elements *twee* dubbele punten gebruiken en voor pseudo-classes één dubbele punt. Hier gebruiken we nog de CSS2 syntax.

Volgens CSS Level 2 zullen de "hover" en "active" toestanden voor meer elementen dan alleen het `a`-element gebruikt kunnen worden. In CSS3 zijn de pseudo-elements en pseudo-classes flink uitgebreid.

2.4. Elementen selecteren op attribuut.

Volgens de CSS-specificatie kun je elementen ook selecteren op hun attributen. Stel je wilt alle hyperlinks benadrukken die naar "www.science.uva.nl" verwijzen. Dan selecteer je de A-elementen op basis van hun HREF attribuut, op één van deze manieren:

```
a[href="www.science.uva.nl"]
a[href~="www.science.uva.nl"]
```

De ~= operator betekent "bevat", de = staat voor exacte gelijkheid.

2.5. Contextuele selectors.

Een voorbeeld hiervan zag je eerder in sectie 1.3.3. In de boomstructuur van een HTML-document zijn allerlei relaties mogelijk tussen elementen. Je kunt elementen selecteren op basis van hun relatie tot een gegeven ander element.

notatie	hiermee selecteer je:
E F	alle elementen F die zich in de boomstructuur onder E bevinden (zie sectie 1.3.3)
E > F	alle elementen F die directe kinderen zijn van E
E + F	alle elementen F die direct na een element E komen (broertjes en direct er op volgend)

In deze tabel zijn E en F "simpele selectors". Dat kunnen elementnamen zijn zoals `a`, maar er mogen ook klasse, pseudoklasse of attribuut-selecties aan hangen. Dus voorbeeldje:

```
h1.jan  em.piet
```

Dit selecteert alle `em` elementen (nadruk) met klassenaam "piet" die zich bevinden binnen een `h1` element (een kopje) met klassenaam "jan".

- Een praktijkvoorbeeldje. Stel je hebt een regel die alle `h1` kopjes met klassenaam "jan" rood maakt, en een regel die alle `em` elementen ook rood maakt. Dan gaat de nadruk verloren als het `em` element zich in een "`h1.jan`" kopje bevindt, want rood in rood valt niet op. Maak een stylesheet van drie regels die dit probleem illustreert en oplost – kortom verzin (en test) een derde regel zodat de nadruk toch zichtbaar wordt in het kopje.

2.6. Economisch bouwen.

Als een serie eigenschappen moet gelden voor elementen e, f, g, h, i, j en k dan kan je het zo doen:

```
e, f, g, h, i, j, k   { eigenschappenlijst }
```

Hier mogen `e`, `f`... etc. zowel "simpele" als "gecompliceerde" selectors zijn, dus niet alleen met klassen maar ook met ieder een context eraan.

2.7. `div` en `span`.

Dit zijn twee elementen met als voornaamste doel als “container” voor `class` of `id` attributen te dienen[5], waar je dan vervolgens in stylesheets gebruik van maakt.
Met `span` kun je delen van een alinea (`p`-element) markeren zonder dat dit op zichzelf invloed heeft op de presentatie. Dan hang je er een `class`-attribuut aan en kun je met de vormgeving gaan spelen. Met `div` kun je meerdere alinea's omspannen om ze in één keer van een `class` labeltje te voorzien.

2.8. Een website harmoniseren.

We zijn klaar met het verkennen van alle selectiemogelijkheden. Als je meerdere webpagina’s wilt vormgeven volgens dezelfde stylesheet dan plaats je deze niet in maar buiten het document: een “extern stylesheet”.

Tip: bouw een stylesheet op binnen een document – dat is makkelijk bij het testen (je hebt alles bij elkaar). Als hij klaar is knip je hem eruit en bewaart hem in een apart bestand. Je kunt de stylesheet dan in meerdere documenten gebruiken door ernaar te verwijzen met een <link>.

Een gelinkte externe stylesheet bevat alleen stijlregels. (geen <style ...> en </style> tags).
Om een document te linken aan een externe stylesheet plaats je het volgende element in de `head`:

```
<link rel="stylesheet" type="text/css" href="pad naar stylesheet"
title="naam">
```

Het `href`-attribuut bevat een relatief of absoluut pad naar de stylesheet, op dezelfde manier als het `href`-attribuut in hyperlinks. Het `title`-attribuut maakt de gelinkte stylesheet toegankelijk om er via Javascript iets mee te doen (met Javascript kun je stylesheets wijzigen in respons op acties van de gebruiker).

Tip: De volgende oefening is te gebruiken als een aanloop naar je website.

2.9. Oefening

- Maak een website met 4 of meer pagina's. De inhoud mag je "jatten" want die doet er nu niet toe. Maak een eigen layout met hulp van stylesheets – verander de layout compleet.

Tip : Verwijder uit de gejatte inhoud eventuele lay-out elementen en attributen als die zich in de `body` bevinden. Zulke elementen en attributen zijn verboden in XHTML, en niet voor

[5] Voor `div` is er nog een ander doel dat we later tegenkomen in verband met Javascript.

niets, want ze maken het werken met stylesheets onmogelijk (zoals je zult merken als je ze laat staan).

- Zorg voor een gemeenschapelijke "look" door een extern stylesheet.
- Ervaar het gemak waarmee je vele pagina's in weinig tijd van uiterlijk kunt laten veranderen! (denk aan de kopieer-acties die nodig zouden zijn als je de styles in de `head` had laten staan).
- Stel je wilt één pagina onderscheiden van de rest (de homepagina?). Dit kun je bereiken door in de `head` behalve een `link` ook een `style` element op te nemen met extra regels.
- Onderzoek welke regels voorrang hebben in geval van een conflict: die in de externe sheet of die in het `style` element. Is dit logisch en gewenst in verband met het doel: één bladzijde een eigen look te geven?
- Onderzoek het nut van `div` en `span` (zie sectie 2.7 hierboven). Is `div` handig? Wat kun je met `span` wat zonder niet lukt?

2.10. Een pagina indelen met `div`'s.

Met `div`-elementen kun je een pagina in tekstvakken verdelen die elk een eigen opmaak moeten krijgen. Je kunt deze tekstvakken ook elk op hun plaats zetten en geschikte afmetingen eraan geven.

- Bedenk een pagina die in verschillende delen verdeeld moet worden, bijvoorbeeld een kolom aan de linkerkant voor navigatie met hyperlinks en een groot tekstvak daarnaast. Het tekstvak moet flexibel zijn, zodat, als de redacteur er tekst bij typt, de rand vanzelf meerekt en hij dus niet meer naar de CSS hoeft te kijken. Elk moet een eigen opmaak krijgen, bijvoorbeeld het tekstvak moet een rand eromheen krijgen (of verzin zelf iets wat beter bij je webtoepassing past...). Gebruik de aanwijzingen en verwijzingen hieronder.

Je hebt misschien gemerkt dat er twee soorten elementen zijn.

– Met elementen zoals `a`, `span`, `em` en `strong` kun je gedeelten van alinea's markeren. Dit worden *inline* elementen genoemd, zij worden altijd zo uitgelijnd dat de regels mooi doorlopen, zelfs als je letters van verschillende grootte op een regel hebt.
– Elementen zoals `p`, `div`, `h1`, `h2` of `table` slaan altijd op een of meer hele alinea's. Zij kunnen niet voorkomen midden in een doorlopende regel. Deze elementen heten *block-level* elementen.

Elk van de block-level elementen maakt zijn eigen blokvormige ruimte, die standaard net zo breed is als de pagina. Je kunt deze breedte echter instellen, en je kunt zulke blokken ook naast elkaar zetten. Deze blokvormige ruimten kun je vormgeven met CSS. Je kunt ze bijvoorbeeld ieder een eigen kader of achtergrondkleur geven, en een apart lettertype voor de tekst die erin voorkomt, als je dat aardig vindt.
Hieronder is een schema van hoe een CSS-box er in het algemeen uitziet:

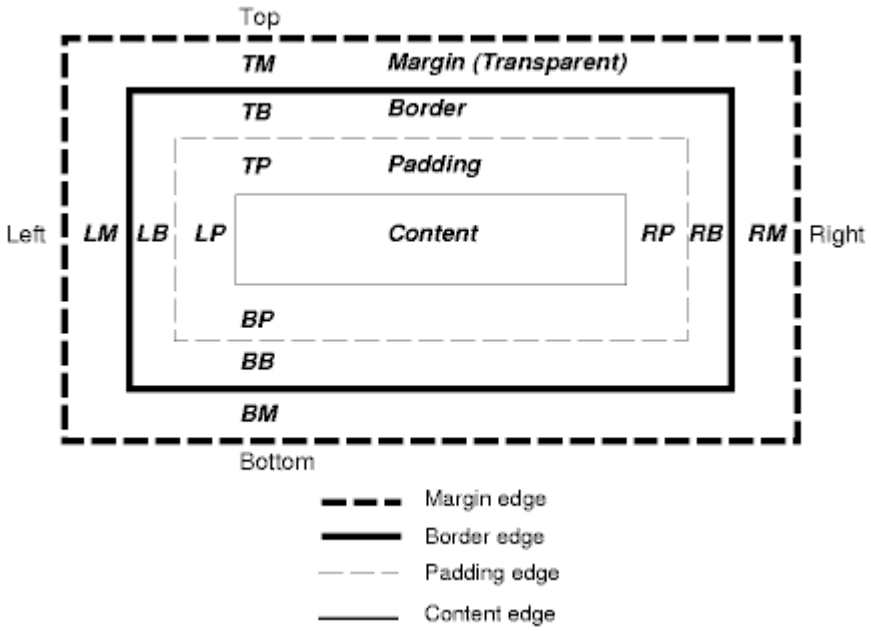


Figure 1: een CSS-box (bron: http://www.w3.org/TR/REC-CSS2/box.html)

Elke box kan een achtergrondkleur en een rand hebben. De *margin* is de ruimte buiten de rand. Deze marge zorgt voor een minimale afstand tot andere boxen. De *border* is de rand zelf, die een bepaalde dikte, kleur en lijn-stijl kan hebben. De *padding* is de ruimte die vrijgehouden wordt tussen de rand en de inhoud. Deze padding krijgt dezelfde achtergrondkleur als de inhoud. De inhoud zelf is tekst, of een plaatje, of andere boxen, want `div`-boxen kunnen eindeloos genest worden.

Verder kun je een blok op zijn plaats zetten met de eigenschappen *top, left, right* en *bottom* en je kunt de afmetingen instellen met *width* en *height*.
Let op, er zijn twee positionerings-schema's. Deze bepalen hoe de eigenschappen top, left, etc. worden opgevat. Je kiest het schema door de eigenschap *position* in te stellen. Dit zijn de twee mogelijkheden:

- *relatieve positionering*. Dit werkt zo: eerst wordt de plaats uitgerekend waar de `div` normaal terecht zou komen, als je geen positie zou kiezen. Dan wordt de `div` verschoven over de afstanden gegeven door *top* en / of *left*. *Top* zorgt voor een verschuiving van de bovenkant (omhoog/omlaag) en *left* geeft een verschuiving van de linkerkant.
- *absolute positionering.* Hier doet het er niet toe waar de box normaal terecht zou komen. De box krijgt nu een plek toegewezen ten opzichte van de hogere omvattende box. De *top* is nu de afstand van de bovenkant van de box tot de bovenkant van de omvattende box, en *left* de afstand van de linkerkant van de box tot de linkerkant van de omvattende box. De hoogste box is altijd die van het `body` element, dat de hele pagina omvat.

Grappig is dat je een box absoluut kunt positioneren, ten opzichte van een box die zelf weer relatief gepositioneerd is! Dus het woordje "absoluut" moet je hier niet te absoluut nemen. Details kun je nazoeken bij w3schools "css-reference".

3. Een database met MySQL

Wil er iets van twee-weg communicatie tussen gebruiker en website-eigenaar mogelijk worden, dan is meestal een database gewenst om de acties, wensen of berichten van gebruikers te registeren. De site-beheerder kan hier dan op reageren door bijvoorbeeld bestellingen uit te voeren, hulpvragen te beantwoorden, of bugs op te lossen. Het alternatief, bijvoorbeeld een email of formulier, is niet erg aantrekkelijk omdat het ordenen van de binnenkomende informatie dan met de hand moet gebeuren. Een database doet dat beter.

3.1. Een relationele database

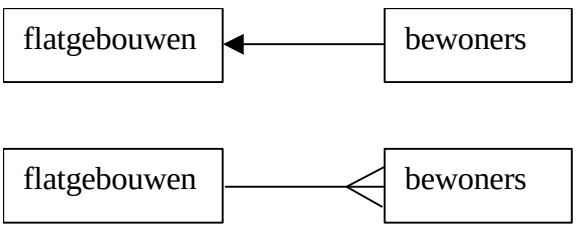
MySQL is een systeem voor het beheren van relationele databases. Een relationele database bewaart gegevens in verschillende *tabellen*. Elk soort object waarover je gegevens wilt bewaren krijgt een eigen tabel.

> Voorbeeld: een verhuurder beheert een aantal flatgebouwen. Hij houdt in een database bij welke bewoners er in die gebouwen wonen. Ook informatie over het onderhoud van de gebouwen komt erin. Doel is om snel brieven met klachten van bewoners te kunnen afhandelen.
> In dit voorbeeld zijn minstens twee tabellen nodig: een tabel voor de flatgebouwen (adres, bouwjaar, onderhoudsgegevens), en een tabel voor de bewoners (naam, flatgebouw, huisnummer, datum laatste brief).

In zo'n tabel staan op elke regel (*rij*) de gegevens van één object (één flatgebouw, of één bewoner). In de *kolommen* staan verschillende eigenschappen van dat object. Zo'n eigenschap kan een stukje tekst zijn, of een getal, of een verwijzing naar een object uit een andere tabel.

> Het is een eigenschap van bewoner 'Jansen' dat hij woont in flatgebouw 'Zonnestein'. Dus in de 'bewoners'-tabel staat in de rij van meneer Jansen een verwijzing naar de rij uit de 'flatgebouwen'-tabel met alle gegevens over 'Zonnestein'.
> In de bewonerstabel is dus een kolom gereserveerd voor deze verwijzingen: bij elke bewoner staat een verwijzing naar de bijbehorende rij uit de flatgebouwen-tabel.

Als een kolom uit tabel B verwijst naar rijen uit tabel A, dan zeggen we dat er een *relatie* is tussen de tabellen A en B. Zulke relaties worden gebruikt bij het opzoeken van gegevens (je zoekt bijvoorbeeld een adreslijst van alle bewoners uit 'Zonnestein').
Hieronder staan twee manieren om de relatie tussen twee tabellen aan te geven, in een ontwerptekening van een database:



> Het eerste schema geeft aan dat elke rij in de bewoners-tabel verwijst naar een rij uit de flatgebouwen-tabel. Zolang we geen specifieke beperkingen inbouwen betekent dit vanzelf, dat diverse bewoners-rijen naar hetzelfde flatgebouw kunnen verwijzen. De

relatie flatgebouw : bewoner is dus van het type "1 op meer" of "1:m" in schema's. De onderste tekening geeft dat duidelijk aan.
Beide schema's komen dus meestal op hetzelfde neer. Als je een schema maakt, kies dan voor één van beide notaties.

In bijzondere gevallen kun je behoefte hebben aan een 1:1 relatie tussen twee tabellen. We behandelen dit hier niet.
Een "meer : meer" relatie tussen twee tabellen kun je alleen realiseren met hulp van een derde tabel. Zie het voorbeeld hieronder.

In dit voorbeeld maken we een meer : meer relatie tussen boeken en auteurs. (Want: Een boek kan meer dan één auteur hebben, en een auteur kan ook meer dan één boek schrijven).
Alle eigenschappen van een boek (isbn, titel, uitgever, jaartal) komen in de "boeken" tabel, alle eigenschappen van een auteur (naam, geboortejaar) komen in de "auteurs" tabel. De "boek_auteur_combinaties" tabel hoeft maar twee kolommen te hebben. Elke rij uit deze tabel geeft één boek-auteur combinatie, in de vorm van referenties naar de twee andere tabellen.
Via deze tabel kun je altijd alle auteurs vinden bij een bepaald boek. En je kunt ook alle boeken vinden van een bepaalde auteur.

Je realiseert een "meer : meer" relatie dus door een extra "combinaties" tabel en twee "1 : meer" relaties te maken.

3.2. Ontwerpproblemen

Bij het ontwerpen van een database doen zich allerlei keuzen voor. Ik behandel hier twee belangrijke, en een ontwerpfout.

3.2.1. Eén kolom of meer kolommen?

Moet bijvoorbeeld iemands naam in één kolom, of neem je aparte kolommen voor voornaam, voorletters, achternaam en eventueel tussenvoegsel?
Het antwoord op zo'n vraag hangt ervan af wat je met de gegevens wilt doen. Als je de gegevens als aparte stukjes wilt kunnen combineren en hergroeperen (bijvoorbeeld sorteren op achternaam), sla ze dan als aparte stukjes op, in verschillende kolommen.

3.2.2. Nieuwe kolom of nieuwe tabel?

Stel: ik heb een tabel met mijn CD-verzameling. Hierin is een kolom "artiest" met de naam van de uitvoerende groep of persoon. Is dat handig of kun je beter een aparte tabel voor de artiesten hebben?
Hierover zijn twee meningen mogelijk. Als je maar één eigenschap van de artiest (zijn naam) wilt bewaren en je hebt geen behoefte die naam op te splitsen in verschillende velden

(voornaam, achternaam) dan kan een aparte tabel wat overkill zijn. Hoewel je redenen kunt hebben om het tóch te doen.
Zodra je van een artiest meer dan één gegeven (dus meer dan één kolom) wilt bijhouden, is een aparte tabel beslist aan te bevelen. Je merkt het voordeel als een eigenschap van een artiest moet worden gewijzigd: je hoeft dan maar één rij te wijzigen. Bovendien is je database beschermd tegen inconsistenties (dezelfde artiest met verschillende geboortejaren).

3.2.3. Kolomnamen met nummers erin

Je kunt wel eens in de verleiding komen om een tabel te maken die er zo uit ziet:

CDverzameling

CDtitel	jaar	artiest1	artiest2	artiest3

In de namen van sommige kolommen komen nummers voor (artiest1 t/m artiest3). Dit is bijna altijd een teken dat je fout bezig bent. Bij een zoekactie op artiest moet je nu gaan programmeren: zoek de CD's waar de gezochte naam staat in kolom`artiest1` OF in kolom `artiest2` OF in kolom `artiest3`. Dat is al niet leuk.
Nu krijg je een CD met 4 artiesten. Gevolg: het tabelontwerp *en* de bijbehorende scripts moeten worden aangepast.
Het is duidelijk: als je meer artiesten wilt bij dezelfde CD, dan moeten de artiesten in een aparte tabel. De relatie is dan van het type "1 : meer", in plaats van "1 : max. 3".

- Je kunt nu je database ontwerpen in de vorm van een schema: tabellen met relaties ertussen. In het rechthoekje dat een tabel voorstelt kun je ook al kolomnamen aangeven, of je kunt dat uitstellen tot later.

3.3. Tabellen in detail ontwerpen

Bij het ontwerpen van de verschillende tabellen wordt voor elke kolom een gegevenstype aangegeven. De belangrijkste gegevenstypen zie je hieronder met hun trefwoorden.

Type	omschrijving	lengte
INT	geheel getal	-
DECIMAL	decimale breuk	aantal cijfers voor en achter (n,m)
DATE	datum	-
DATETIME	tijdstip, inclusief datum	-
CHAR	tekst, starre kolom	aantal letters
VARCHAR	tekst, variabele kolom, sorteerbaar	aantal letters
TEXT	tekst, variabele kolom, niet sorteerbaar	-

-: invullen meestal niet nodig

Bij sommige gegevenstypen moet een lengte worden aangegeven. De meeste kolomtypen hebben 'vaste lengte'. Dit wil zeggen dat MySQL tevoren weet hoeveel bytes er gereserveerd zijn voor elke tabelcel.

Een tabel die geheel uit kolommen met vaste lengte bestaat, is sneller bij zoekacties. Bij lange tekstkolommen neemt zo'n tabel echter onredelijk veel ruimte in. Daarom definieert men langere teksten als VARCHAR of als TEXT. Zodra in een tabel één VARCHAR of TEXT kolom voorkomt, worden alle CHAR kolommen door MySQL opgevat als VARCHAR omdat het voordeel van vaste lengte dan toch al weg is.

Meer informatie over kolomtypen vind je in de MySQL handleiding.

In elke tabel is het nuttig om een kolom te hebben met waarden die gegarandeerd uniek zijn binnen de tabel. Zo'n kolom noem je de *primary key*. Een standaard-methode om een primary key te maken in MySQL is als volgt:

- Geef de tabel een kolom (vaak "id" genoemd) met automatisch oplopende gehele getallen (AUTO_INCREMENT optie) en maak een PRIMARY index bij deze kolom.

Een 1 : meer *relatie* tussen twee tabellen A en B implementeer je nu als volgt:

- Geef tabel B een kolom met waarden die overeenkomen met waarden uit de primary key kolom van tabel A. Deze kolom noemt men een *foreign key* omdat hij verwijst naar een andere tabel. Deze kolom moet ook een index krijgen (zie volgende paragraaf)

- Maak het tabelontwerp af: bij elke tabel een lijstje velden met hun kolomtype

3.4. Indexen plannen

Bij de meeste tabellen heb je enkele indexen nodig; die kunnen het zoekwerk spectaculair versnellen. Als je later snelheidsproblemen hebt met de database, is het nuttig eerst te kijken of alle gewenste indexen gemaakt zijn. Overwegingen:

- Voor elk veld, of elke combinatie van velden, waarop je regelmatig wilt zoeken of sorteren, heb je een index nodig.
- Voor elk veld (bijv. iemands emailadres) waar je dubbele waarden wilt voorkomen heb je een index nodig van het type UNIQUE.
- Voor elke foreign key kolom heb je een index nodig.
- Elke tabel waarnaar wordt verwezen heeft een primary-key kolom nodig, dit wil zeggen een kolom met een index van het type PRIMARY.
  (Veel MySQL-gebruikers geven elke tabel standaard een "id" kolom met AUTO_INCREMENT aan, als primary key).

Indexen vertragen het invoeren, updaten en verwijderen van gegevens. Daarentegen versnellen zij het zoeken en sorteren.

Voorkomen van dubbele invoer is belangrijk bij het op orde houden van een database. Bijvoorbeeld in een gebruikerstabel wil je voorkomen dat dezelfde persoon tweemaal wordt ingevoerd (met mogelijkheid van inconsistenties). Een gegeven dat gegarandeerd uniek is op je website, zoals een emailadres, is ook geschikt om de gebruiker tegen dubbele invoer te beschermen met een UNIQUE index.

- Plan je indexen. Noteer in het lijstje velden dat je eerder maakte welke velden een index krijgen, en evt. wat voor soort index.

3.5. Een database aanvragen

Op websec is voor elke projectgroep één database beschikbaar. Binnen deze database mag je zoveel tabellen maken als nodig is.

Tip: Experimenteer met de database door voor jezelf een tabel (of meerdere tabellen) te maken. Maak binnen je groep afspraken over de naamgeving van de tabellen zodat duidelijk is van wie ze zijn, bijvoorbeel `robert_tabel1`. Gebruik geen punt in de naam van een tabel!

De database is beschermd door een loginnaam (: de groepnaam) en password die aan iedereen in de groep beschikbaar wordt gesteld.

3.6. Een MySQL database beheren met phpMyAdmin

We gaan nu de ontworpen tabellen in de database neerzetten. We noemen hier een paar tools om met MySQL te communiceren.

1) *de mysql command-line client op websec*; deze is alleen toegankelijk als je via SSH ingelogd bent op websec[6]. Met deze client kan je direct commando's intypen die door de MySQL server uitgevoerd worden waarna het resultaat wordt getoond. Over het algemeen wordt deze methode alleen door gevorderden gebruikt. De client wordt gestart door het volgende commando:

   `mysql –u groepnaam –p`

   Verander *`groepnaam`* in de naam van je groep. Als wordt gevraagd om een password geef je het bijbehorende groep password.
2) *phpMyAdmin*; Dit is een web-interface naar de database, zoals je die zelf ook gaat maken. Met dit verschil dat deze webinterface bedoeld is voor database-beheerders en niet voor eindgebruikers. Er zijn vriendelijke "wijs en klik" methodes voor het creëeren en wijzigen van tabellen. Als je iets bijzonders wilt kun je terugvallen op het invoeren van een SQL-query in het daarvoor bestemde query-venster. Het adres van de phpMyAdmin pagina is:

   `https://websec.science.uva.nl/phpmyadmin/`

   Log in met je groepnaam en bijbehorend groep password.

Hieronder wordt *phpMyAdmin* geïntroduceerd zodat je deze kunt gebruiken om je eerste tabellen te maken.

- Ga naar dit adres:

  `https://websec.science.uva.nl/phpmyadmin/`

- Log in met je groepnaam en bijbehorend groep password.

[6] Remote toegang is om security redenen niet beschikbaar.

Je krijgt een pagina met aan de linkerkant een navigatieframe met daarin een aantal iconen en de naam van twee databases: `information_schema` en de naam van jouw database. Laat de eerste met rust.

- Klik op de naam van je database. Een pagina verschijnt met normaliter een lijstje tabellen, maar die is nu nog leeg. Je kunt nu je eerste tabel gaan maken.

3.6.1. Tabel maken

- Als dat al niet het geval is: selecteer aan de bovenkant tab “Structure”.
- Typ *een naam* voor je tabel en het gewenste *aantal velden* (=kolommen).
- Klik "Go".
  Je krijgt een nieuw scherm waarin je je velden kunt definieren.
- Voor elk veld:
  - Typ een naam voor het veld.
  - Kies het type uit de lijst.
  - Vul eventueel een lengte in (of dit nodig is; hangt af van het type).
  - Over een eventuele “Default” gaan we het zo dadelijk hebben.
  - Over het algemeen is het verstandig “Collation” te laten zoals die is.
  - Als attribuut is UNSIGNED aan te bevelen bij getallen die niet negatief kunnen worden.
  - Kies uit "Null" of "Not null": het laatste betekent dat het veld verplicht is om in te vullen (door de gebruiker of door jouw applicatie, straks).
  - "Not Null" velden krijgen een standaardwaarde (“Default”). Deze wordt gebruikt als er geen gegevens zijn. Kies zelf een standaardwaarde of laat het aan MySQL over...
  - “A_I” staat voor "auto_increment". Dit is zinvol als je een automatisch oplopende primary-key kolom wilt.
  - Stel het maken van indexen nog even uit. (suggestie)
- Klik "Save" om de nieuwe tabel te bewaren.

3.6.2. Tabel wijzigen

Je kunt het ontwerp van een tabel achteraf nog wijzigen of uitbreiden. Wijzigen is lastig als je al php-scripts hebt gemaakt die iets doen met je tabel: deze moeten dan ook gewijzigd. Een tabel uitbreiden, dus nieuwe kolommen erbij maken, kan nooit kwaad.

- Klik op de naam van de tabel in het linkerframe
  Rechts verschijnt een overzicht van de velden en hun type, lengte, en dergelijke. De lengte wordt achter het type tussen haakjes weergegeven.
- Klik op de checkboxen bij de velden die je wilt wijzigen
- Klik op "Change" (of op "Drop" als je deze velden geheel wilt verwijderen – gegevens in de gekozen kolommen gaan verloren)
- Je krijgt een venster dat lijkt op het venster toen je de tabel maakte, maar nu met alleen de geselecteerde velden. Je kunt het type, de lengte, en andere opties wijzigen.
- Klik "Save".

3.6.3. Een index maken op één kolom

Een index maken op een enkele kolom gaat snel.

- Klik op de naam van de tabel in het linkerframe, om naar de tabel te gaan waarbij je de index wilt maken.

- In het overzicht van kolommen kies je de kolom waarbij de index moet komen en klik je op één van de volgende iconen:
    - "Primary", om een primary key te maken,
    - "Index", om een gewone index te maken,
    - "Unique", om een index te maken die afdwingt dat elke waarde maar éénmaal voorkomt in de kolom,
    - "Fulltext", om een index te maken die volledige-tekst-zoekacties ondersteunt - dit is nuttig voor velden met langere teksten, als je middenin de tekst iets wilt kunnen vinden.

Klik op “Yes”. De index is dan gemaakt. Je ziet hem verschijnen in het lijstje “Indexes”.

3.6.4. Een index maken op een combinatie van kolommen

Als je een index wilt maken op een combinatie van twee of meer kolommen dan:

- Klik op de naam van de tabel in het linkerframe om naar de tabel te gaan waarbij je de index wilt maken.
- Zoek het zinnetje "Create an index on [1] columns”.
- Typ in plaats van 1 het aantal kolommen waarop je wilt indexeren
- Klik "Go".
  Je krijgt een nieuw scherm, met net zoveel "velden" als je net koos.
- Typ een naam voor de index, of laat dit leeg.
- Kies een Index type uit de lijst (keuze uit: PRIMARY, INDEX, UNIQUE, FULLTEXT).
- Voor elk veld waarop je wilt indexeren,
  in volgorde van 1^{e} sorteersleutel, 2^{e} sorteersleutel, etc.:
    - kies het gewenste veld uit de lijst
- Klik "Save".

De nieuwe index verschijnt in het lijstje met indexen.

Als de structuur van alle lege tabellen gemaakt is, dan kun je beginnen om met PHP een gebruikersinterface te bouwen bij de database. Dat wil zeggen: webformulieren maken waarmee de database gevuld en gewijzigd kan worden. Hiervoor stuurt PHP opdrachten naar de MySQL server. Die opdrachten zijn gesteld in de SQL opdrachtentaal.

3.7. MySQL bedienen met de SQL opdrachtentaal

Systemen voor het beheren van relationele database (zoals MySQL) verstaan allemaal de SQL taal. In deze taal kun je zoekopdrachten geven, of opdrachten voor het wijzigen van de inhoud van de database.

Bij het bouwen en vullen van de database kun je phpMyAdmin gebruiken en dan heb je de SQL-taal dus niet nodig (phpMyAdmin gebruikt deze voor je). Straks als je zelf PHP-scripts gaat schrijven, moeten daarin ook opdrachten aan de database worden gegeven. Daarbij zul je SQL nodig hebben.

We vertellen hier de eerste beginselen van SQL. De volledige handleiding is beschikbaar op de website van MySQL. SQL heeft commando's om gegevens in te voeren, te wijzigen, te wissen en om de database te doorzoeken[7]. Hieronder volgt een overzicht met voorbeelden.

[7] Er zijn ook commando's om tabellen te creëeren of te wijzigen. Deze hebben we niet nodig zolang we phpMyAdmin bij de hand hebben.

keyword	omschrijving
INSERT	rij toevoegen
UPDATE	rijen wijzigen
DELETE	rijen wissen
SELECT	rijen opvragen

We geven van deze commando's de eenvoudigste syntax, die je het vaakst nodig hebt. Voor meer opties wordt verwezen naar de MySQL handleiding.

INSERT syntax[8]:

INSERT INTO *tabelnaam*
SET *veldnaam* = *waarde*[, *veldnaam* = *waarde*, ...]

voorbeeld:

INSERT INTO users SET username="robbel", password="geheim"

In een tabel met kolommen 'username' en 'password' wordt één rij ingevoerd, waarbij deze kolommen een waarde krijgen toegekend. Als er in deze tabel meer kolommen zijn, dan krijgen deze vanzelf de standaardwaarde. Aan een "auto-increment" kolom moet je bijvoorbeeld nooit een waarde toeschrijven.

UPDATE syntax:

UPDATE *tabelnaam*
SET *veldnaam* = *waarde*[, *veldnaam* = *waarde*, ...]
WHERE *voorwaarde*
[LIMIT *n*]

voorbeeld:

UPDATE users SET password="geheimxx" WHERE username ="robbel"

Hier worden alle rijen opgezocht die voldoen aan de voorwaarde dat username="robbel". In al deze rijen wordt de password-kolom gelijk gemaakt aan "geheimxx". Pas op, als je de WHERE voorwaarde weglaat worden alle rijen in je tabel gewijzigd! In de voorwaarde kun je logische operatoren (AND, OR en NOT) gebruiken. Met de optionele toevoeging "LIMIT 1" kun je zeker stellen dat niet meer dan één record (of het aantal dat je opgeeft) wordt gewijzigd.

DELETE syntax:

DELETE FROM *tabelnaam*
WHERE *voorwaarde*

---

[8] INSERT is er in drie varianten, allemaal met hun voor- en tegens. Deze is het eenvoudigst. Zie de MySQL-handeiding.

[LIMIT *n*]

voorbeeld:

DELETE FROM users WHERE username="robbel"

De voorwaarde werkt hetzelfde als bij de update. Er kunnen dus vele records worden gewist. Als je de WHERE voorwaarde vergeet, wordt je hele tabel leeg gemaakt.
Ook hier kun je een beveiliging tegen teveel wijzigingen inbouwen met LIMIT.

SELECT syntax:

SELECT *kolommenlijst*
FROM *tabel_met_joins*
[WHERE *voorwaarde*]
[ORDER BY *kolomnamenlijst*]

Over ieder van de onderdelen valt nu heel wat te zeggen.
Voorbeelden van een goede *kolommenlijst*, direct na SELECT:

```
username, password
user.username, user.password
user.username AS naam, user.password AS wachtwoord
```

In het eerste voorbeeld noem ik alleen de kolomnamen. In het tweede zet ik de naam van de tabel ervoor. Dit is bijna steeds nodig als we achter FROM meerdere tabelnamen gaan vermelden. In het derde voorbeeld heb ik elk veld een alias-naam gegeven. Als ik dat gedaan heb, dan kan ik in de *kolomnamenlijst* bij ORDER BY deze alias-namen gebruiken (dus niet bij WHERE). Deze alias-namen verschijnen bovendien als kolomnamen in het array dat PHP later krijgt toegespeeld.

Voorbeelden van een goede FROM *tabel_met_joins*:

```
FROM mycds
```

```
FROM mycds
INNER JOIN artists ON mycds.artists_id = artists.id
```

```
FROM mycds
INNER JOIN artists ON mycds.artists_id = artists.id
INNER JOIN labels ON artists.label_id = labels.id
```

In het eerste voorbeeld is er alleen een tabelnaam genoemd.
In het tweede voorbeeld wordt gebruik gemaakt van een bestaande relatie tussen twee tabellen (mycds en artists), om een gecombineerde zoekactie te doen in beide tabellen tegelijk.

- In dit voorbeeld is 'artists_id' een *foreign key*-kolom, die in de 'mycds' tabel voorkomt en die verwijst naar de artists tabel.
- 'id' is hier de *primary key* van de artists-tabel

De rijen uit beide tabellen worden geselecteerd en gekoppeld waar beide *key* velden aan elkaar gelijk zijn.

Je kunt net zoveel tabellen JOIN-en als je wilt. In het derde voorbeeld zijn er drie tabellen en twee relaties gebruikt: een relatie van 'mycds' met 'artists' en een relatie van 'artists' met 'labels'.

Voorbeelden van goede WHERE *voorwaarden*

```
WHERE mycds.title="Oh Suzy Q"

WHERE mycds.title="Oh Suzy Q" AND mycds.year="1970"

WHERE mycds.year="1970"
AND (artists.name="Armin van Buuren" OR artists.name="Buffalo Springfield")
```

Je kunt voorwaarden combineren met AND, OR en NOT. Daarbij kun je haakjes gebruiken, zoals hierboven voorgedaan.
Als je verschillende tabellen hebt ge-JOIN-ed dan kun je voorwaarden stellen aan kolommen uit verschillende tabellen.
De tabelaanduiding 'mycds' kun je weglaten, als alle kolommen uit dezelfde tabel komen.

Voorbeeld van een goede ORDER BY *kolomnamenlijst*

```
ORDER BY mycds.title, artists.name
ORDER BY naam, wachtwoord
```

Als je direct na SELECT aliassen hebt gemaakt, dan kun je die hier gebruiken.

Tot zover deze summiere beschrijving van de voornaamste mogelijkheden. Voor meer informatie, raadpleeg de MySQL-handleiding op de website van dit vak.

Oefening

- Als je een paar SQL commando's wilt uitproberen: dat kan met *phpMyAdmin*. Selecteer je database en druk ofwel op de SQL tab bovenaan het scherm, of op het “Query window” icoon in de navigatiekolom. In het laatste geval krijg je een nieuw venster waarin je een SQL commando kunt typen. Klik op “Go” om de query uit te voeren.
    - Na een SELECT query krijg je een lijst te zien met het selectieresultaat.
    - Na een INSERT, UPDATE of DELETE query krijg je te zien hoeveel records er zijn gewijzigd. Om het resultaat precies te bekijken klik je op de naam van de tabel en dan "Browse" om de gewijzigde inhoud van de tabel te inspecteren.

Tip: Het is vaak handig om een query eerst uit te proberen in phpMyAdmin voordat je de query in je php-code verwerkt.

4. PHP uit voorbeeldjes

In dit hoofdstuk worden de basisideëen van PHP en van server-side scripting besproken aan de hand van voorbeeldjes en oefeningen. Dit is bedoeld om snel aan de slag te kunnen in week 2. Een meer systematische inleiding in PHP als taal vind je op

`http://w3schools.com/php/`

Een complete naslaghandleiding van PHP staat op de website van PHP:

`http://www.php.net/manual/en/`

Om daar je weg te kunnen vinden, moet je de taal al een beetje kennen!

4.1. Redenen voor server-side scripting.

Met HTML, CSS en Javascript kunnen gebruikers interacteren met een webpagina, maar niet met de aanbieder van de webpagina. De communicatie van webserver met browser is nog éénrichtingverkeer. Om dat te veranderen moet er aan de server-kant iets gebeuren.
Bekende redenen voor server-side scripting zijn:

- Tweerichtingverkeer: je wilt iets terughoren van de bezoekers van je pagina,
- Een website makkelijker hanteerbaar maken ('content-management') en daardoor actueler,
- Je wilt je programmeerwerk geheimhouden.

Er zijn ook redenen om zuinig te zijn met server-scripts: ten eerste belasten ze de webserver en ten tweede gaat elke interactie via het netwerk en kost dus tijd.

4.2. CGI versus HTML-embedded scripts

CGI (Common Gateway Interface) is een algemene standaard om web-gebruikers te laten interacteren met een programma op de server. Het werkt zo: de gebuiker vult een formulier in waarbij een GET of POST actie-attribuut is ingesteld. De browser stuurt een HTTP-GET of POST verzoek naar de server. Dit bevat de URL van het gevraagde programma en alle nodige input parameters (de velden op het formulier, elk gevolgd door zijn waarde). De webserver-software start het gevraagde programma en geeft de input parameters door. Het programma doet met de input wat het wil en heeft vervolgens de plicht om ofwel output te produceren[9] die door de webserver wordt teruggestuurd naar de browser, ofwel een verwijzing naar een bladzijde die moet worden getoond. Daarna wordt het programma beëindigd.
Elk programma dat aan de CGI-voorwaarden voldoet kan als server-side script gebruikt worden, ongeacht wat het met de data doet en ongeacht de taal waarin het is geschreven. In combinatie met CGI worden vaak algemene programmeertalen gebruikt, zoals C/C++, Perl en Python.

HTML-embedded scripts zijn scripts die, net als Javascript, in HTML-code ingebed worden met gebruik van speciale tags. Het document krijgt een afwijkende extensie (bijvoorbeeld .php) waardoor de webserver weet dat het document aangeboden moet worden aan de PHP-

[9] Minstens een HTTP-header die het content-type van de output beschrijft (vb. "text/html", "image/gif"), dan een lege regel en dan de gegenereerde data.

(of ASP- of Coldfusion-) interpreter. Deze leest en verwijdert vervolgens de script-code. Als een script-fragment output produceert, komt dit op de plek te staan waar het script eerst stond. Het resultaat wordt naar de browser gestuurd.

Hoewel de interpreter van een HTML-embedded script een CGI-programma kan zijn, hoef je van CGI niets te weten als je embedded scripts schrijft. Een script-programmeur maakt dus een keuze: hij maakt óf een CGI-script (dan moet je meer van CGI weten dan hier verteld wordt) of een embedded script. Voors en tegens staan hieronder.

CGI	HTML-embedded scripts
voor:	**voor:**
– vrije keuze programmeertaal – gebruik van een algemene taal mogelijk	– efficienter en makkelijker bij specifieke webklussen
tegen:	**tegen:**
– geen speciale voorzieningen voor http en html	– apart taaltje leren

Wij gaan gebruik maken van HTML-embedded scripts.

4.3. Systemen voor embedded scripts

Drie ongeveer gelijkwaardige systemen voor embedded scripts zijn:

– PHP (vroeger afkorting van “Personal Home Page”, nu van “PHP: Hypertext Preprocessor”).
– ASP (Active Server Pages) van Microsoft.
– Coldfusion van Macromedia.

PHP en Coldfusion hebben elk hun eigen taal. Bij ASP kun je kiezen tussen Javascript of VBScript als taal. Een vierde systeem dat hierop lijkt is JSP (Java Server Pages) van Sun, met Java als programmeertaal. Hier is er echter een striktere scheiding tussen HTML en Java, deze komen niet voor in hetzelfde document. In plaats hiervan worden in het HTML-document speciale JSP-tags neergezet die verwijzen naar java-klassen die elders in een jar-file staan.

In deze cursus leer je PHP gebruiken. De manier van ontwerpen en structureren van scripts is in alle drie systemen gelijk, maar zoals gewoonlijk staan alle punten en komma's op een andere plaats.

4.4. Mijn eerste PHP pagina: een formulier

Als simpel voorbeeld demonstreren we hoe een PHP-pagina reageert op een HTML-formulier dat wordt gesubmit. We hebben dus twee files: een HTML-formulier en een PHP-pagina.

Dit is de code voor het HTML-formulier. Deze is beschikbaar op "/var/www/html/voorbeeldcode/PHP/mijnformulier.html":

```
<!DOCTYPE html
      PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
```

```
      <title>Een HTML formulier</title>
</head>
<body>
      <form method="post" action="mijnpagina.php">
            <p>Hoe heet je?
                  <input type="text" name="jenaam" />
                  <input type="submit" />
            </p>
      </form>
</body>
</html>
```

Hier volgt de PHP-pagina. Deze is beschikbaar op "/var/www/html/voorbeeldcode/PHP/mijnpagina.php":

```
<!DOCTYPE html
      PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
      <title>De eerste php pagina</title>
</head>
<body>
      <?php
            print "<p>Hallo, beste ". $_POST['jenaam'] . ".</p>" ;
      ?>
</body>
</html>
```

Als je beide files kopieert naar je public_html map op websec dan kun je het formulier oproepen via http://websec.science.uva.nl/~*jeaccountnaam*/mijnformulier.html.
Als je daarna het formulier invult (“Jan”) en op submit klikt, wordt de php-pagina geactiveerd en deze geeft als output:

Hallo beste Jan.

Dit voorbeeld illustreert het principe van PHP, en van alle embedded scripts: je maakt een gewone webpagina. Daarbinnen kun je eilandjes met programmocode maken. Deze eilandjes beginnen met <?php ... en eindigen met ...?>. Bij elkaar is dit een geldige XML-*processing instruction*. De instructies worden uitgevoerd door de applicatie die na het vraagteken staat aangeduid, in dit geval dus door php. Dit gebeurt op de webserver. De output van het script komt op de plaats te staan van de processing instruction.
Een processing instruction hoeft niet zo kort te zijn als hierboven. Je kunt complete programma's van vele pagina's schrijven in PHP.

4.5. PHP Uitproberen.

Deze oefening is wel erg basic. We schrijven een PHP-programma dat de opbrengst berekent van een éénmalige inleg van geld op een rekening met vaste rente.
Dit is de formule:

opbrengst = inleg * (1 + rentepc / 100)^jaren

Het rentepercentage is 3% als de looptijd (“jaren”) kleiner of gelijk is aan 3. Het is 5% als de looptijd 4 jaar is of langer. Alleen hele aantallen jaren zijn toegestaan.

Volg de aanwijzingen hieronder om het gewenste script te maken:

- Een begin van een php-pagina vindt je in "/var/www/html/voorbeeldcode/PHP/begin.php”
- Input van de gebruiker krijg je via een webformulier. Een geschikt formulier vind je in "/var/www/html/voorbeeldcode/PHP/form.html”. Bekijk de HTML-code van het formulier en let op de "action" en "method" attributen van het FORM-element. Kopieer het formulier naar je eigen "public_html" map en wijzig indien nodig het "action"-attribuut zodat het naar jouw PHP-script verwijst.
- Omdat het “method”-attribuut in het FORM-element de waarde “get” heeft, zijn de waarden van de velden uit het formulier beschikbaar in het $_GET array, zo[10]:

  $*mijnvariabele* = $_ GET[‘*formulierveldnaam*’] ;

- Namen van variabelen worden steeds voorafgegaan door $, statements eindigen allemaal met ;
- Een array in PHP kan een getal of een string als index hebben. $_ GET is een voorbeeld van een (automatisch beschikbaar) array met strings als indices. Een array met een string-index noemt men een “associatief array” of “hash”.
- Beschikbare controle-structuren vind je in de online referentiehandleiding, onder het kopje “control structures”. Je hebt er minstens één nodig.
- Een variabele declareren doe je door die variabele voor 't eerst een waarde te geven. Er is geen speciaal commando om variabelen te declareren.
- Een globale variabele maak je door een element toe te voegen aan het (automatisch beschikbare) $GLOBALS array. Zo:

  $GLOBALS[‘*globalevariabele*’] = *waarde*

  Je kunt dit ook zo zien: het $GLOBALS array is de enige globale variabele die bestaat. Telkens als je een globale variabele wilt gebruiken moet je expliciet dit array aanroepen.
- Output van PHP is (meestal) een complete webpagina[11]. Het hele PHP-script levert één HTML-bladzijde op waar alle output in staat. Output genereer je zo:

  ```
  print "letterlijke tekst" . $var . "vervolg tekst";
  ```

  of zo:

  ```
  print "letterlijke tekst $var vervolg tekst";
  ```

  waar $var een variabele is die eerder een waarde heeft gekregen. De output komt op de plek waar het script heeft gestaan.
- Een punt, . is dé operator om strings aan elkaar te rijgen. PHP is makkelijk: als $var toevallig een getal voorstelt wordt het automatisch omgezet in een string.

---

[10] Over het verschil tussen “GET” en “POST” meer in sectie 4.9.
[11] De nieuwe AJAX techniek brengt hier verandering in! Meer hierover in week 3.

- In strings tussen *dubbele* aanhalingstekens worden variabelen vervangen door hun waarde. Een string kan ook tussen *enkele* aanhalingstekens staan maar dan worden variabelen-namen daarbinnen niet vervangen. (dus, syntax 2 alleen met dubbele).
- De geprinte tekst moet valide HTML opleveren.
- Test het script dat je gemaakt hebt door het via http://websec... op te vragen in een browser.

4.6. De PHP – MySQL connectie

PHP-webpagina's worden de "voordeur" waardoor het publiek straks bij de database kan komen. Wij gebruiken intussen phpMyAdmin als "achterdeur" voor de webmaster, om tijdens het bouwen en testen te controleren wat er gebeurt.

Voor de communicatie tussen PHP en MySQL kun je tegenwoordig kiezen uit vele stijlen, waarvan we er hier eerst één uitlichten, het gebruik van *prepared statements* via de "mysql improved" bibliotheek van php (kortweg mysqli). We kiezen hier bovendien de object-geörienteerde syntax.[12] Volgens deze stijl verloopt de communicatie tussen PHP en de MySQL server in de volgende stappen:

1) Creëer een *mysqli*-object. Dit object stelt een verbinding met de database voor. Bij de constructie geef je het adres van de server, je username, wachtwoord en de naam van de database mee.
2) Maak een SQL-opdracht als string. Deze string kan vraagtekens bevatten, die parameters voorstellen om later in te vullen. Roep nu de *prepare* methode van je mysqli-object en geef de string mee. Je krijgt als resultaat een *prepared statement* object.
3) Roep *bind_param* om de parameters van het statement-object te binden aan variabelen in je PHP-script.
4) Roep *bind-result* om kolommen uit het queryresultaat alvast te binden aan variabelen in je script - als het een statement is die resultaten oplevert, bij INSERT e.d. hoeft dit dus niet.
5) Roep *execute* om het prepared statement uit te voeren.
6) Hierna zijn de resultaten van de query beschikbaar. Met de *fetch* methode kun je naar de eerste rij toegaan. (Als er geen rijen zijn krijg je *false* terug). Hierna merk je dat de gebonden "result" variabelen nu waarden hebben gekregen: je kijkt naar de eerste rij van de resultaatset. Met de *fetch* methode kun je de pointer verschuiven naar volgende rijen uit het resultaat, net zolang totdat je *false* terugkrijgt.
7) Stappen 5 en 6 kun je meermalen herhalen voor dezelfde prepared statement, wat zinnig is als je intussen ook de waarden van de gebonden parameters verandert (de vraagtekens).
8) Tenslotte gooi je de resultaten weg met met $statement→*free_result*() of $statement→*close*().
9) En nog later sluit je de connectie met $mysqli→*close*()

Hierna kunnen de objecten worden weggegooid.

Probeer het nu zelf:

- Voorbeelden van deze stappen vind je in het "claimwachtw2.php" script in "/var/www/html/voorbeeldcode/PHP/claimwachtw". Dit script wordt gestart door het formulier "claimwachtw2.html". Let op de foutafhandeling bij de stappen 1 t/m 6.
- Als je een 'user' tabel hebt om gebruikersnaam en wachtwoord combinaties te bewaren, dan kun je nu een php-script bouwen dat de wachtwoord-controle uitvoert en het resultaat

[12] In de PHP-handleiding vind je de alternatieven, waaronder ook de "klassieke" procedurele syntax.

meldt. Zie het vorige hoofdstuk voor een overzicht van SQL-commando's en zie de MySQL-handleiding.

- De MySQL username en wachtwoord die je van ons kreeg kun je het best goed beveiligen. Plaats deze in een php-bestand buiten de root-directory van de webserver en “include” deze file. Je bent dan beveiligd tegen aanvallen van buiten (Waarom?).

Omdat boven beschreven stappen zo vaak voorkomen, kan het vervelend worden om ze telkens opnieuw te typen, met de fout-afhandeling functies erbij. Het opbouwen van een functiebibliotheek of, nog beter, een stel *klassen* ligt voor de hand. In de volgende paragraaf krijg je een beginnetje van zo'n stel klassen te zien.

4.7. Standaard beveiliging van PHP scripts

Een paar goede gewoonten kun je het best meteen aanleren:

- Krakers zijn dol op foutmeldingen. Ze leveren informatie over de inbraakgevoelige plekken in jouw webtoepassing. Het is daarom een goede gewoonte *foutmeldingen display uit* te zetten als je applicatie klaar is, maar in plaats daarvan *foutmeldingen te loggen*, zodat de eigenaar toch zicht houdt op het functioneren van de site.
  Op "/var/www/html/voorbeeldcode/dblibrary” vind je "fatalerror.inc.php" een simpel bibliotheekje waarmee je de display of de logging van foutmeldingen kunt regelen. Lees de code nauwkeurig, wijzig waar nodig, en probeer uit!
- Je vindt daar ook twee php-klassen "mymysqli" en "mystatement". Dit zijn twee varianten van de klassen die we net bespraken, met ingebouwde foutafhandeling. Ze werken samen met "fatalerror.inc.php" om mysql-foutmeldingen te vertonen of te loggen.
  Door deze klassen te gebruiken in plaats van de standaard php mysqli-klassen, bespaar je je het herhaalde typwerk van de foutafhandeling!
- Om "SQL-injecties" te voorkomen is het nodig alle input die van gebruikers komt *via parameters* aan prepared statements toe te voegen. Je moet invoer van gebruikers dus niet direct in je querystring verwerken.
- Pas op met gebruikers-input op plekken waar vreemde files in het programmaverloop betrokken kunnen worden. Een bekende is:

  include $var

  Als $var een door een gebruiker ingevoerde waarde bevat, dan zou deze gebruiker willekeurige php-code kunnen opstarten. De websec-server is in voorgaande jaren via zo'n truc gekraakt.
- Kijk uit waar je de include file neerzet waar de loginnaam en wachtwoord staan waarmee je contact opneemt met je database.
  In claimwachtw2.php ben ik zelf expres slordig geweest, maar kijk eens naar claimwachtw1.php (d.i. de serieuze versie): dat wachtwoord is via het web niet te vinden.

4.8. Verschillende stijlen in de communicatie met MySQL

Hierboven is één stijl uit de doeken gedaan. We behandelen nog kort twee andere, omdat je die waarschijnlijk vroeg of laat ook tegenkomt, zeker als je in de PHP-handleiding gaat bladeren:

1) De klassieke stijl: *directe queries*, zonder parameters. Voor MySQL 4.1 waren er geen prepared statements en voor PHP 5 was er geen mysqli-bibliotheek die daar gebruik van kon maken. Destijds moest je invoer van gebruikers wel rechtstreeks in je query-string

verwerken. Het gevaar van SQL-injecties werd destijds bestreden door systematisch alle invoer van gebruikers te "escapen", d.i. door de functie mysql_real_escape_string() heen te halen.
Deze stijl van werken is nog steeds mogelijk en populair.

2) *Opgeslagen procedures*. Vanaf MySQL 5.0 is het mogelijk om in SQL procedures te schrijven die op de MySQL-server als onderdeel van je database worden bewaard. Zulke procedures hebben ook parameters. Als je werkt volgend deze stijl dan hoeft je PHP-script niets meer van SQL te weten. Het hoeft alleen maar de namen van opgeslagen procedures te kennen. Het aanroepen van een opgeslagen procedure kan zelf weer via een prepared statement gebeuren.

De stijl die we hiervoor behandelden zit in zekere zin tussen deze twee uitersten in.

- Wijzig je php-scripts zodat je makkelijk kunt overschakelen van foutmelding *display* naar *logging* via een centrale DEBUG constante.
- Test mijn "mymysqli" en "mystatement" klassen. Kijk of dit je typewerk bespaart.
  Je mag de klassen nog uitbreiden of wijzigen.
- Probeer stored procedures. Kijk of dit voordelen heeft. Blijven "mymysqli" en "mystatement" nu bruikbaar? Zijn er wijzigingen die je zou willen doen?

4.9. Variaties in de relatie van script en formulier

Bij het FORM-element kun je kiezen tussen `method="get"` en `method="post"`. Hierbij behoren ook twee verschillende arrays in PHP: $_GET en $_POST.

- Probeer het verschil (kijk naar de location-balk in je browser). Welke van de twee zou je kiezen als...
  i. je wilt dat bezoekers bookmarks en hyperlinks kunnen maken naar specifieke zoekresultaten in je database?
  ii. je dit juist wilt voorkomen?
- Stel dat iemand het script direct oproept zonder eerst het formulier te gebruiken. Hoe reageert het script?

Tip. Formulier en script kunnen tot één bestand gecombineerd worden (zie "claimwachtw3.php"). Het script kán dan niet zonder formulier worden opgeroepen. Het idee is als volgt: de eerste keer dat het script/formulier wordt opgeroepen mag het script niet worden uitgevoerd, dit heeft immers nog geen zin. Via een `if`-statement kun je dit voorkomen.
De gebruiker vult nu het formulier in, drukt op submit en het formulier/script roept nu *zichzelf* op. Nu mag het script wel gaan werken.

- Probeer de tip en maak een gebruiksvriendelijke fout-afhandeling.
- Bij een invoer-formulier ligt het voor de hand dat je meer dan één record achter elkaar wilt invoeren. Lukt dit snel en handig?

4.10. Een toevoegen/wijzigen formuliertje

Met de informatie hierboven kun je een formulier + php-code maken om informatie toe te voegen aan je database.

- Probeer het.

Het is vaak handig om voor updaten hetzelfde formulier + script te gebruiken als voor het invoeren van nieuwe gegevens.
Bij een update heb je een extra probleem: het formulier moet onthouden wélk record daar wordt weergegeven, zodat het update-script weet welke rij moet worden gewijzigd.

Tip: Geef het formulier een `input`-element met `type="hidden"` (een verborgen formulierveld) en bewaar hierin de primary-key waarde van de rij die wordt weergegeven[13]. Als het formulier gebruikt wordt voor invoer van nieuwe rijen, blijft dit verborgen veld dus leeg. Het script weet nu wat het doen moet door naar dit verborgen veld te kijken.

- Pas het eerder gemaakte formulier + script aan zodat het ook geschikt is voor updaten (als je dat wilt). Of maak een apart script voor de update!

4.11. Een site modulair opbouwen met 'templates'.

Veel sites hebben een banner en navigatie die op elke pagina van de site zichtbaar is. Dit effect is eenvoudig te bereiken met frames of met server-script 'templates'. Nadelen van frames zijn bekend: via zoekmachines worden alleen losse pagina's gevonden, nooit de bijbehorende frameset-pagina. Je kunt toestanden van een frameset niet bookmarken, etc.
Een alternatief voor frames zou zijn: op elke pagina de banner en navigatie-links simpelweg te herhalen. Dit zou het nadeel hebben dat er bij elke wijziging in de navigatie veel kopieerwerk nodig is om alle pagina's bij te werken.
Server-side scripts zoals PHP bieden hiervoor een oplossing: we kunnen de HTML-pagina opbouwen uit losse files, 'templates', die door de PHP-interpreter worden gecombineerd tot een complete bladzijde.

Table 1: PHP-functies voor het werken met templates

functie	conditioneel gebruik	eenmalig
include(file)	kan	nee
include_once(file)	kan	ja
require(file)	nee, altijd uitgevoerd	nee
require_once(file)	nee, altijd uitgevoerd	ja

De 'file' bevat een brok van de pagina. Dit brok hoeft op zich geen valide HTML te zijn, bijvoorbeeld het `head`-element kan ontbreken, of er kan een `<body>` tag voorkomen, zonder de bijbehorende sluittag `</body>`.
Op de URL waar de gebruiker naartoe gaat staat een php-script, dat met hulp van een serie include- of require-opdrachten de pagina in elkaar zet, uit de losse onderdelen. Bij elkaar moet dit een valide HTML-document opleveren.
In geval van "include" kun je if-statements gebruiken om bepaalde brokken al dan niet te gebruiken.

Tip: In de URL kun je variabelen meegeven aan het script, die bepalen welke brokken worden getoond. Bijvoorbeeld zo:

[13] Aangezien de primary key een betekenisloos getal bevat hoeft de gebruiker dit nooit te zien. Bovendien is het van belang dat hij dit getal niet per ongeluk wijzigt.

http://www.mijnserver.nl/mijnscript.php?pagina=mijnpagina

De waarde van de ‘pagina’ parameter kun je weer opvragen via $_GET:

$pag = $_GET[‘pagina’] ;

Het script kan nu zelf bepalen welke inhoud er getoond moet worden onder de banner en naast de hyperlinks, die bijvoorbeeld voor alle pagina’s gelijk blijven.

Pas op: laat de gebruiker niet volledig vrij om te bepalen welk php-script er wordt uitgevoerd! Bouw garanties in dat alleen scripts worden uitgevoerd die tot jouw applicatie behoren.

De opdrachten “include” en “require” zijn ook nuttig om bijvoorbeeld een bibliotheek van veelgebruikte PHP-functies in te voegen in elke script-pagina van je site.

4.12. Oefening: een site modulariseren met templates.

Stel je hebt een site met *n* pagina's die allemaal dezelfde banner en navigatiebalk aan de linkerkant moeten krijgen. Ook krijgen ze een footer met o.a. "laatst gewijzigd op..." die automatisch gegenereerd kan worden.

- Maak een testsite voor *n*=2. Zorg dat geen inhoud dubbel voorkomt[14]. Zie hieronder voor een overzicht van PHP-functies die handig zijn in headers en footers.
- Waar zet je de DOCTYPE-declaratie, <head>-tags, links naar stylesheets?
- Als je kiest om de hele "bovenkant" (inclusief `head`) in een include-file te zetten, wat betekent dat voor de titels van de pagina's, kunnen die nog verschillen?

Tabel 1: Overzicht van handige PHP-functies voor gebruik in headers en footers

functie	omschrijving
getdate( )	huidige datum en tijd als timestamp
filemtime(file-padnaam)	file-modified datum+tijd als timestamp
date (formatstring, timestamp)	formatteer een datum+tijd om te printen
$_SERVER[‘HTTP_USER_AGENT’]	browser-informatie*

* geen functie maar een door PHP gedefinieerde variabele.

Een voorbeeld van een formatstring voor datums is "d-m-Y". Een overzicht van alle mogelijkheden vind je in de PHP referentiehandleiding op de cursus website (kijk onder "Date and time functions").
Het is mogelijk verschillende stylesheets te linken, afhankelijk van browser en besturingssysteem van de gebruiker. In PHP is deze informatie beschikbaar via de variabele $_SERVER[‘HTTP_USER_AGENT’].

- Beoordeel voor je eigen webtoepassing of templates daar nuttig kunnen zijn, en modulariseer je site.

[14] Tags die met verplichte documentstructuur te maken hebben, zoals <head> mogen dubbel voorkomen – immers de kans dat je die ooit wilt wijzigen is gering. Het doel is om "dubbel wijzigen" te voorkomen!

4.13. Frames-navigatie zonder frames.

Eerder hebben we het over de nadelen van frames gehad. Ze zijn verboden in XHTML-strict. Het effect van frames dat diverse delen van het venster onafhankelijk van elkaar kunnen scrollen, kan ook zonder frames bereikt worden: met CSS-stylsheets.
Door PHP-templates en CSS te combineren kun je het effect van frames volledig nabootsen, zonder <frameset> te hoeven gebruiken. Zie voorbeeldcode/HTML+CSS/CSS-frames voor een voorbeeldje.

5. Sessies, Beveiliging, Invoercontrole

In dit hoofdstuk drie onderwerpen die alle drie nodig zijn bij het maken van een web-database.
1. Sessies zijn een methode om te zorgen dat gegevens getypt in het formulier op pagina 1 ook bekend zijn als de bezoeker is verder gesurft naar pagina *n*. Een toepassing van sessies is om na het inloggen te onthouden wie de gebruiker is.
2. Het is meestal nodig een database te beveiligen tegen onbevoegde toegang. Vaak zijn er verschillende groepen gebruikers met verschillende rechten.
3. Controle op valide invoer is enerzijds een service aan de gebruiker, maar heeft ook te maken met beveiliging tegen fraude of hacks. Bij deze twee kanten aan de invoercontrole horen ook twee verschillende technieken: met Javascript of met PHP?

We eindigen met een korte notitie over datums. Datums in PHP en in MySQL worden enigszins verschillend behandeld. We vatten kort samen hoe je in beide systemen met datums kunt werken en hoe je gemakkelijk van het ene naar het andere format komt. Gebruik het zodra je het nodig hebt.

5.1. PHP-Sessies.

Een gebruiker kan via een URL of via een formulier informatie meegeven aan een script. Het script verwerkt de informatie en geeft een bladzijde als resultaat. Wanneer de gebruiker achter elkaar twee script-pagina's bezoekt dan weet script B niet wat script A gedaan heeft. Dit is op den duur een vervelende beperking! Het kan bijvoorbeeld betekenen dat de gebruiker steeds dezelfde informatie opnieuw moet intypen. De oplossing hiervoor zijn "sessies". Met hulp van sessie-functies krijg je de beschikking over variabelen die gedurende de hele "sessie" blijven bestaan, en die door meerdere scripts op dezelfde site gebruikt kunnen worden.
Doel is om te zorgen dat script B kan weten wat er in script A is gebeurd tussen de server en gebruiker *x*. Hiervoor is nodig, dat de gebruiker *x* een tijdelijke ID krijgt zodat we hem kunnen herkennen als hij opnieuw contact opneemt.

PHP kan zulke ID's realiseren op twee manieren:
1) door een "cookie" te plaatsen op de machine van de gebruiker. Deze cookie krijgt standaard géén expiry date, dit wil zeggen dat de cookie direct gewist wordt bij afsluiten van de browser. We herkennen de gebruiker dus gedurende één browser-sessie, vandaar de naam "sessies".
2) door een string "?PHPSESSID=..." toe te voegen aan alle hyperlinks in elk script-document op de site. Dit lukt door deze hyperlinks te genereren met PHP, als volgt: `<a href="`*padnaam*`?<?php print SID ?>">`[15]
3) als je zelf een server beheert, kun je in het PHP-initialisatiebestand ("/etc/php.ini") de opdracht `session.use_trans_sid=1;` plaatsen. Daarna worden automatisch alle hyperlinks voorzien van de PHPSESSID parameter.

[15] Je vraagt je misschien af; waarom niet **$**SID? SID is een constante, geen variabele. Voor een constante wordt geen $ gezet.

Methode 1 heeft het voordeel dat het óók nog werkt als de gebruiker zelf een nieuwe URL intypt, in plaats van op onze hyperlinks te klikken. Cookies kunnen echter worden uitgezet en in dat geval bieden methode 2 of 3 uitkomst.

We illustreren nu het gebruik van sessies met een kort experiment.

- Stel je hebt een site met 3 (script-)pagina's. Twee gebruikers mogen toegang hebben tot de site. Elk van beide kan als "admin", of als "user" inloggen. Het doel van dit experiment is, dat na inloggen niet alleen script 1, maar ook script 2 en 3 weten wie er is ingelogd, en met welke rol (admin of user).

Binnen één script gaat het gebruik van sessie-variabelen als volgt:

– Bovenaan het script roep je de functie `session_start();`
  Gevolg: er wordt gecontroleerd of er al sessie-gegevens zijn voor de huidige gebruiker. Indien ja, dan worden deze automatisch opgehaald. Indien nee, dan wordt er een nieuwe ID aangemaakt voor de huidige gebruiker (dus een cookie verstuurd, etc.).
– Alle sessie-variabelen bevinden zich in het $_SESSION array. Je kunt als volgt een nieuwe sessie variabele maken:

  $_SESSION['*mijnvariabele*'] = *waarde* ;

  Verder kun je het $_SESSION array gebruiken zoals je normaal een array gebruikt[16].
– Je kunt ook variabelen uitlezen waarvan je hoopt dat ze al eerder in een ander script gecreeerd zijn. Eerst testen of de variabele inderdaad bestaat:

```
if ( isset($_SESSION['anderevariabele']) )
          { doe er iets mee }
```

– Als het script eindigt, worden de waarden die de sessie-variabelen op dat moment hebben vastgelegd, tot dezelfde gebruiker (herkenbaar aan zijn SESSID) weer terugkomt.

Hierna kan de gebruiker een nieuw script starten waar zich het verhaal herhaalt. Merk op dat de gebruiker in principe vrij is om de scripts in willekeurige volgorde op te roepen. Als site-ontwerper kun je kiezen om die vrijheid eventueel aan banden te leggen, als dit voor de applicatie nodig is.

- Test de overdracht van gegevens tussen de twee of drie scripts: weet de server nog wie er is ingelogd, en met welke rol?
- Wat doe je als iemand script 2 oproept zonder eerst te zijn ingelogd via script 1? Hoe zou je sessie-variabelen gebruiken om een verplichte volgorde af te dwingen tussen 3 of meer scripts/pagina's?

Een nuttige functie tenslotte is

    session_destroy() ;

---

[16] Als in het PHP-initialisatiebestand (vaak "/etc/php.ini") register_globals=true voorkomt, dan zijn alle sessie-variabelen beschikbaar als globale variabelen. Dus $_SESSION['mijnvar'] kun je dan óók opvragen als $mijnvar! Dit wordt sterk afgeraden, omdat krakers zulke scripts makkelijk kunnen foppen door via een formulier of URL variabelen in te voeren die dan functioneren alsof het sessie-variabelen zijn.

Hiermee wordt een sessie beëindigd. Een gebruiker die zijn computer verlaat zonder zijn browser af te sluiten kan op een 'logout' link klikken om te voorkomen dat een collega bijvoorbeeld persoonlijke gegevens kan opvragen als gevolg van een niet-beëindigde sessie.

5.2. Beveiliging met wachtwoorden – keuze van aanpak

Beveiliging van een site met wachtwoorden kan dienen om de toegang te beperken tot een selecte groep, of om de privacy van deelnemers aan de site te beschermen. Een site-beheerder kan daarom kiezen uit verschillende aanpakken:

1) Vrije aanmelding, dit betekent meestal ook: zelf een wachtwoord kiezen
2) Binnenkomst alleen op uitnodiging, dus beheerder kiest een initieel wachtwoord.
   a) dit is niet te wijzigen, óf
   b) het is wel te wijzigen, mits je al een wachtwoord hebt.

Mogelijkheid "b" wordt gekozen als de beheerder geen absolute macht mag hebben, of voor het comfort van gebruikers, die een zelfbedacht wachtwoord beter onthouden.

Al deze mogelijkheden zijn met PHP-scripts te realiseren. Maar je kunt ook gebruik maken van de beveiliging-mogelijkheden van de Apache webserver. Hieronder de voors- en tegens van beide aanpakken.

Beveiliging via PHP – MySQL zelf regelen
voordelen • MySQL is sneller in het doorzoeken van een grote gebruikersdatabase • Met PHP-MySQL kunnen gebruikers zelf hun wachtwoord wijzigen via het web • Gebruikersnamen en wachtwoorden in één tabel met andere gebruikersgegevens (die je meestal toch in MySQL wilt hebben). **beperkingen** • Bugs in het script kunnen de beveiliging in gevaar brengen • Elk script moet apart beveiligd (kan wel economisch met een "include")
Beveiliging via Apache webserver ".htaccess"
voordelen • Robuste toegangscontrole, ongeacht de kwaliteit van de scripts • Een hele directory (incl. subdirectories) in één keer beveiligen **beperkingen** • Alleen server-beheerder kan de wachtwoorden instellen

Op "websec" kun je allebei de technieken uitproberen. Aangezien de PHP-MySQL-aanpak het lastigst, maar ook het flexibelst is, vragen we jullie deze aanpak zeker in je site te verwerken. We bespreken deze aanpak eerst en laten daarna zien hoe het met .htaccess kan.

5.3. Wachtwoorden – de PHP/MySQL benadering

MySQL kent een "MD5"-functie die een getypt wachtwoord omzet in een code die veilig kan worden opgeslagen in een kolom van een database. Terugvertalen van de code naar het wachtwoord is niet mogelijk (en ook niet nodig). Een ingetypt wachtwoord kan met het opgeslagen wachtwoord worden vergeleken door de MD5-functie nog eens te gebruiken.

```
$SQL = "INSERT INTO tabelnaam ( veldnaam ) VALUES (MD5('$wachtw'))";
```

```
$result = mysql_query($SQL, $linkID)
```

Hier is $*wachtw* een PHP-variabele die het wachtwoord bevat dat de gebruiker heeft getypt in een formulier[17]. Pas op: gecodeerde wachtwoorden nemen soms meer ruimte in dan ongecodeerde. Het veld waarin het wachtwoord wordt opgeslagen moet een lengte hebben van 32 tekens.
Bij een volgend bezoek van deze gebruiker kan er een test gedaan worden als volgt:

```
$SQL = "SELECT * FROM tabelnaam WHERE
     gebruikersnaam = $gebruikersn AND
     wachtwoord = (MD5('$wachtw'))";
$result = mysql_query($SQL, $linkID)
if (mysql_num_rows($result) > 0)
     { ...gebruiker krijgt toegang ...}
```

Hier zijn $*gebruikersn* en $*wachtw* de gebruikersnaam en het wachtwoord zoals getypt door de gebruiker. Daarentegen zijn *gebruikersnaam* en *wachtwoord* twee velden uit de MySQL database waarin de juiste gegevens zich bevinden. Het getypte $*wachtw* wordt eerst met MD5 omgezet en wordt daarna pas vergeleken met het opgeslagen (gecodeerde) wachtwoord.

- In je webtoepassing is het nodig sommige gebruikers te registeren omdat deze bijzondere rechten hebben. Maak een tabel (in MySQL) om naam, wachtwoord en bijzondere rechten voor deze gebruikers vast te leggen.
- De Beheerder heeft een script nodig om gebruikers in te voeren, met o.a. een initieel wachtwoord[18]. Maak eerst een onveilig script. Later moet dit script zelf beveiligd worden met een wachtwoord – dat kan natuurlijk pas zodra de beheerder zelf een wachtwoord heeft.
- HTML-formulieren kennen een apart type `<input ...>` element voor wachtwoorden. Zie: "w3schools" HTML-reference, onder "HTML-Forms".
- Om het wachtwoord onderweg tussen browser en server te beveiligen is een "HTTPS" (Secure HTTP) verbinding nodig. De eigenaar van de server (websec) heeft daartoe een certificaat moeten aanvragen. De webpagina moet via "https: //" worden opgevraagd.
- Voor het geval de gebruiker dit vergeet, kun je de gebruiker een "redirect" geven als hij binnenkomt via het niet-beveiligde HTTP-protocol. Een voorbeeldje hiervan vind je op "/var/www/html/voorbeeldcode/PHP/https-omleiding".

5.4. Sessies gebruiken voor beveiliging

Stel we hebben drie scripts die alle drie slechts gerund mogen worden door bevoegde personen. Met behulp van sessies kunnen we zorgen dat de inlog maar éénmaal hoeft plaats te vinden. Het resultaat is een stel sessie-variabelen met bijvoorbeeld gebruikersnaam of ID erin. Volgende scripts hoeven alleen te testen of één van deze variabelen een geldige waarde heeft, het is niet meer nodig de gebruiker lastig te vallen.

- Maak een inlogscript. Als het inloggen lukt, krijgt de gebruiker toegang tot een menu dat aangepast is aan zijn rechten.

---

[17] We hebben gebruik gemaakt van de bijzonderheid dat PHP een variabele-naam binnen een string tussen " " zal vervangen door zijn waarde.
[18] Dit lukt niet met MS-Access of MySQL-front, omdat deze de MD5 encryptie-functie niet kennen.

- Probeer een script waarmee de gebruiker zelf zijn wachtwoord kan veranderen, mits hij correct is ingelogd.
- Het script van de beheerder (uit par. 2) kan nu ook gemakkelijk worden beveiligd (en toegevoegd aan het menu?).

5.5. Is het nu echt veilig?

De communicatie tussen een browser en server volgens het HTTP protocol is “clear-text”. Iedereen die toegang heeft tot een machine op hetzelfde netwerk is in staat om de transmissie van gegevens te volgen. Een persoon met toegang en kennis van de juiste tools is vervolgens in staat de gegevens te gebruiken om de communicatie naar zijn hand te zetten.
De transmissie van gegevens van browser naar server is goed beveiligd als het HTTPS protocol wordt gebruikt. Voor zo'n verbinding moet de eigenaar van de server over een certificaat beschikken waarmee hij aantoont dat hij is wie hij zegt te zijn. Dit garandeert de bezoeker dat de beveiliging geen fake is. Gegevens gaan vervolgens in gecodeerde vorm van de browser naar de server.
Daar aangekomen is het van belang dat webserver en MySQL-server op dezelfde computer draaien, want de communicatie tussen beiden gebeurt in 'clear-text', dus zou afgeluisterd kunnen worden als het via een netwerk gebeurde.
PHP-scripts bevatten de nodige wachtwoorden voor het inloggen op MySQL in clear text. We hebben gezien dat PHP-scripts niet via de webserver zichtbaar te maken zijn – zolang de webserver goed functioneert[19]! Als extra beveiliging kan men riskante scripts buiten de directory-boom van de webserver plaaten. Op de plek waar ze eerst stonden komt dan een simpel scriptje met een "include".

- Deze simpele maatregel kun je direct nemen als je denk dat 't nodig is.

Verder hangt de veiligheid nog af van het besturingssysteem van de server. Directories buiten de webserver-boom zijn niet toegankelijk via "http:" of “https:”. Of een inbreker er langs andere weg in kan komen hangt af van het besturingssysteem van de server en de services die er draaien.

5.6. Toegangscontrole met Apache “.htaccess”-bestanden

Hieronder worden de mogelijkheden van Apache “.htaccess” bestanden beschreven. Hier kunnen gebruikers niet alleen op hun naam en wachtwoord gescreend worden (situatie 2), het is ook mogelijk gebruikers te weigeren of toe te laten op basis van hun domein van herkomst.

situatie 1: gebruikers toelaten op basis van domein.
Plaats in de te beveiligen map een ".htaccess" bestand met bijvoorbeeld de volgende inhoud:

```
order deny,allow
deny from all
allow from uva.nl
```

Let op: tussen deny en allow staat wel een komma, maar géén spatie.

---

[19] Als de webserver vergeet dat .PHP een geregistreerd bestandstype is dat een bijzondere behandeling verdient kan men PHP-bestanden gewoon opvragen. Je kunt bijvoorbeeld PHP-bestanden direct opvragen van iedere webserver waar géén PHP-interpreter is geïnstalleerd.

Effect: iedereen wordt tegengehouden, behalve degenen met een DNS-naam in het uva.nl-domein.

situatie 2: toelaten op basis van naam en wachtwoord
Eerst moet door de beheerder van de site een wachtwoordenbestand worden gemaakt. Daarvoor gebruikt je het Apache-commando `htpasswd`:

htpasswd –c *wachtwoordfile gebruikersnaam*

De optie "-c" gebruik je alleen de eerste keer. Hiermee geef je opdracht een nieuw wachtwoordbestand te maken. *wachtwoordfile* is de padnaam van de gewenste wachtwoordfile. Het is goed gebruik deze file *buiten* de directory-boom van de webserver te plaatsen, dus in ons geval buiten "public_html" of “htdocs”. *gebruikersnaam* is de naam van de eerste gebruiker die je wilt toevoegen aan het bestand.

Er wordt om een wachtwoord gevraagd. Typ dit en de eerste gebruiker is toegevoegd. Herhaal nu het bovenstaande commando – *zonder* de -c optie! – voor alle volgende gebruikers die je wilt toevoegen. Hierna kun je elke gewenste map beveiligen met een ".htaccess" bestand zoals het volgende:

```
AuthUserFile wachtwoordfile
AuthName gebiedsnaam
AuthType Basic
require valid-user
```

Iemand die een bestand uit de map opvraagt krijgt een inlogscherm met twee invulvelden voor naam en wachtwoord, en de tekst:

Enter username for *gebiedsnaam*

Hier is *gebiedsnaam* de naam die je zelf aan het beveiligde gebied hebt gegeven. Je kunt meerdere mappen beveiligen met dezelfde gebiedsnaam. Het effect daarvan is dat de gebruiker maar éénmaal hoeft in te loggen voor dat hele gebied.

5.7. Integratie Apache beveiliging met PHP

Als een map beveiligd is met een “.htaccess” dan zullen gebruikers niet via PHP inloggen maar via Apache. Dat lijkt nadelig, want nu weet jouw script niet wie er is ingelogd... Daar bestaat een simpele oplossing voor: het script kan de gebruikersnaam opvragen.

$user = $REMOTE_USER;

Een grote beperking van beveiliging via Apache is, dat het wachtwoordbestand alleen handmatig kan worden bijgewerkt. Webgebruikers kunnen niet hun eigen wachtwoord kiezen of wijzigen.

Meer informatie over beveiliging met Apache vind je op
http://httpd.apache.org/docs/howto/auth.html

5.8. Datums in PHP en MySQL

Omdat MySQL en PHP ieder hun eigen systeem hebben voor datums besteden we er hier kort aandacht aan.

MySQL heeft een apart kolomtype voor datums. Dit kolomtype verwacht invoer in het voor ons ongewone formaat yyyy-mm-dd (jaartal-maandnummer-dagnummer).
In PHP is "timestamp" het centrale formaat voor datum-tijd informatie. Een timestamp is een getal, namelijk het aantal seconden sinds 1 januari 1970, 00:00:00. Timestamps zijn voor menselijke ogen weinig informatief, maar PHP rekent er graag mee. Er zijn dan ook diverse functies beschikbaar voor converteren van en naar timestamp.

Als invoer van de gebruiker krijg je ofwel een string van een vorm zoals "01-01-2012", maar je kunt het invoerformulier ook zo organiseren dat je drie getallen krijgt. Beide mogelijkheden zijn hieronder geïllustreerd.

Typ hier de datum: [ ] (dd-mm-yy)

Typ hier de datum: [ ]-[ ]-[ ]

De drie-getallen variant heeft het voordeel dat geen instructie over gewenste separators nodig is. De lengte van de drie INPUT-elementen kan met de attributen `size` en `maxlength` zo ingesteld worden dat niet meer dan het gewenste aantal cijfers wordt geaccepteerd (zie "w3schools" HTML-reference, onder HTML-Forms).

Hebben we een invoer van de gebruiker in deze vorm, dan wordt dit als drie aparte variabelen doorgegeven aan het PHP-script[20], bijvoorbeeld: $d, $mnd, $yr.

Als we geen ander doel hebben dan deze informatie door te geven aan MySQL voor opslag in een date-kolom, dan voldoet de volgende omzetting:

```
$datestrMysql = "$yr-$mnd-$d";
```

Willen we daarentegen eerst in PHP met de gegeven datum gaan rekenen, dan kan omzetting naar een timestamp nodig zijn:

```
$phptimestp = mktime(0, 0, 0, $mnd, $d, $yr);
```

De functie "mktime" verwacht 6 parameters, waarvan we de eerste drie op nul kunnen zetten als het tijdstip van de dag ons niet interesseert. Na de nodige bewerkingen kan een timestamp weer omgezet worden in een voor Nederlandse begrippen leesbaar formaat:

```
$datestrNed = date("d-m-Y", $phptimestp);
```

---

[20] Als je gekozen hebt om de gebruiker één string dd-mm-yyyy te laten invoeren dan kun je in PHP de "explode" functie gebruiken om de string in drie delen te splitsen – zie de PHP referentie onder “Function Reference → Text Processing → Strings”.

De "date" functie accepteert twee parameters, waarvan de eerste een string is die het gewenste format aangeeft. Met dezelfde functie kan het timestamp geschikt gemaakt worden voor opslag in MySQL:

```
$datestrMysql = date("Y-m-d", $phptimestp);
```

Er zijn in PHP nog meer functies beschikbaar voor het omgaan met datums en tijden. Steeds is timestamp hierbij het centrale format. Een overzicht van deze functies vind je in de online PHP-naslag, onder het kopje " Function Reference → Date and Time Related Extensions → Date/Time".

6. Javascript

Javascript en PHP zijn allebei programmeertalen die in webpagina's worden ingebed. Een belangrijk verschil is dat PHP wordt uitgevoerd op de webserver. De bezoeker krijgt de php-code nooit te zien, ook al zou hij het willen. Javascript wordt uitgevoerd door de browser van de bezoeker. De pluspunten en minpunten van javascript zijn een gevolg hiervan:

- De gebruiker kan Javascript uitzetten.
- Javascript code is niet geheim te houden.
- Javascript belast de server niet.
- Javascript kan sneller reageren op acties (muiskliks) van de gebruiker.
- Met Javascript kun je effecten bereiken die op een andere manier niet lukken.

Het feit dat Javascipt uitgeschakeld kan worden betekent vaak dat Javascipt alleen voor niet essentiële verbeteringen wordt gebruikt. De toepassing moet immers ook nog werken als Javascript uit staat. Alleen als je zeker weet dat je toepassing belangrijk is voor de gebruikers, kun je vereisen dat Javascript aanstaat. Dit is bijvoorbeeld het geval bij intranet toepassingen in een bedrijfsomgeving. Het is ook mogelijk alleen voor vertrouwde sites Javascript aan te zetten in de browser en voor alle andere sites uit.

Hier zijn vier simpele trucjes om een indruk te geven waar Javascript voor te gebruiken is.

6.1. vier simpele trucjes.

Voorbeeld-bestandjes staan op "/var/www/html/voorbeeldcode/Javascript". Als je Javascript al aardig kent is het niet de moeite waard om er naar te kijken.

1) NieuwVenster: via een link een nieuw venster openen waarvan de eigenschappen (hoogte, breedte, aankleding) door mij zijn gekozen.
2) VerwijsDoor: een gebruiker doorverwijzen na een verhuizing - al dan niet na een tijdvertraging. Direct doorverwijzen kan ook met PHP, maar met vertraging is meestal beter.
3) InputVerwerken : input vragen, deze verwerken en het resultaat tonen. Dezelfde renteberekening die we eerder met PHP deden lukt ook met Javascript. En nu doet de computer van de bezoeker het werk.
4) Muistrucjes: Reageren op muisbewegingen of kliks. Als de applicatie niet alleen op de submit knop moet reageren dan... Javascript!

Dit is een Javascript statement uit voorbeeld 1 (NieuwVenster):

```
window.open('tweede.html', 'mijnvenster',
'height=200,width=200,left=300,top=200,screenX=300,screenY=200,
location=no,menubar=no,resizable=no,scrollbars=no,status=no');
```

Deze statement is opgehangen aan het `onclick` attribuut van een hyperlink, er staat letterlijk:

```
<input type="button" value="Klik hier"
     onClick="window.open('tweede.html', 'mijnvenster',
     'height=200,width=200,left=300,top=200,screenX=300,screenY=200,
     location=no,menubar=no,resizable=no,scrollbars=no,status=no');"
     />
```

Het gedeelte na `onClick=,` tussen de de dubbele aanhalingstekens, is een Javascript statement. De rest is HTML-code.

6.2. Overzicht van manieren om Javascript in een HTML document te verwerken:

- Zet het script tussen `<script type="text/javascript">` en `</script>`
    - plaats het in de `head` als je wilt dat het geladen is zodra de gebruiker de pagina ziet (zie voorbeeld: "InputVerwerken").
    - plaats het onder aan de `body` als het script pas moet worden uitgevoerd als de pagina is geladen[21] (Voorbeeld: "VerwijsDoor").
- Bewaar het script als een extern bestand met de achternaam "js" (javascript). Plaats in de `head` of de `body` (keuze: zie vorige punt) de volgende tags:

  `<script type="text/javscript" src="`*`bestandsnaam`*`.js"></script>`

- Gebruik één van de *gebeurtenis (event)*-attributen: (zoals onClick, onMouseOver) van een HTML-element om daar het script aan te koppelen. Gevolg: het script wordt uitgevoerd als de gekozen gebeurtenis optreedt! (hierboven: voorbeelden 1, 3 en 4)
    - Net als elke waarde voor een atribuut staat het hele script nu tussen dubbele aanhalingstekens.
    - Gevolg: als je in het script aanhalingstekens wilt / moet gebuiken, dan moeten dit enkele aanhalingstekens zijn (zie hierboven rond 'tweede.html').

6.3. Wanneer uitgevoerd?

Scripts worden meteen uitgevoerd als ze worden geladen. Een uitzondering zijn "function" statements. Een functie wordt pas uitgevoerd als hij wordt aangeroepen. Een functie *aanroepen* doe je vaak elders in het document:

1) bij een gebeurtenis-attribuut van een HTML-element (zie voorbeeld InputVerwerken – de "onload" gebeurtenis van het `body` element),
2) via een "setTimeout" statement.

Een functie *definiëren* doe je altijd in de `head` om er zeker van te zijn dat ze zijn geladen als de gebruiker ze activeert (Voorbeeld: InputVerwerken).

6.4. Hoe zit Javascript als taal in elkaar?

Zie de diverse “tutorials” voor een vollediger beschrijving! Ik raad je dringend aan de Javascript tutorial op w3schools te volgen. Ik pik er hier wat dingen uit, zodat je snel met hulp van de referentie-handleidingen aan het werk kunt:

- Namen van variabelen en functies zijn hoofdlettergevoelig!
    - als je een variabele "leenBedrag" hebt genoemd, moet de B steeds groot.

---

[21] in dat geval kun je ook de "onLoad" gebeurtenis van het `body` element gebruiken (zie hieronder over gebeurtenis-attributen).

    - als een functie "berekenTijd" heet in de `head` van je document, moet je de functie ook zo aanroepen, met de T groot. (InputVerwerken)
- Letterlijke gegevens zijn er in 3 basistypen: getallen, logisch (*boolean*) en objecten. Twee veelgebruikte objecten zijn de *strings* en de *arrays*. In andere talen zijn dit dikwijls aparte datatypen maar hier zijn het soorten objecten.
- Variabelen hebben geen vast type, dat wil zeggen je kunt er elke soort gegevens in stoppen. Gebruik van getallen en variabelen zie je in voorbeeld "InputVerwerken". Het string objecttye zie je in alle voorbeelden, bijv. alle URL's zijn strings.
- Je merkt dat een string een object is aan de manier waarop je ermee werkt, bijvoorbeeld de lengte van een string is een eigenschap. Je krijgt hem zo:
  `stringvariabele``.length`
- Variabelen declareer je door er "var" voor te zetten de eerste keer dat je ze een waarde toekent. Dit is alleen nodig binnen functie-definities, als je de geldigheid van de variable wilt beperken tot de functie waarin je hem declareert.
- Elke statement eindigt met een ;
  Dit is niet verplicht, tenzij meerdere statements op één regel staan, dan moet het echt. Het is echter een goede gewoonte om het altijd te doen, ook al omdat het in andere talen (PHP) wel verplicht is.
- Je kunt soms 2 statements in elkaar nesten. Het volgende komt uit voorbeeld "VerwijsDoor":

  ```
  setTimeout("window.location.href='http://www.science.uva.nl';", 6000);
  ```

  De eerste statement is 'setTimeout'. Deze statement heeft tussen de haakjes twee parameters, waarvan de éérste een string is die zelf weer een complete Javascript-statement bevat.
- Aanhalingstekens: letterlijke strings staan, zoals in de meeste talen, tussen aanhalingstekens. Je mag kiezen uit enkele of dubbele. Doordat er twee soorten zijn kan je een string binnen een string maken – nuttig bij geneste statements (zie vorige punt) en ook nuttig als je Javascript en HTML moet mixen (HTML gebruikt ook aanhalingstekens).
- Een verzameling statements wordt tussen accolades { } geplaatst en kan dan gebruikt worden in controle-statements, zoals:
  if (*voorwaarde*) { *statements* }
  while (*voorwaarde*) { *statements* }
  function (*parameters*) { *statements* }
  Zie voorbeeld "InputVerwerken" voor een paar controle-statements. Een overzicht van alle beschikbare controle-statements staat op de DevGuru site (zie de naslagwerken).
- Je mag statements altijd over meer regels verdelen. Bij controle-statements is dat een goede gewoonte. Bij gewone statements geeft de ; het einde aan. Bij controle-statements geldt de laatste } als het einde. Een ; plaatst men niet bij controle-statements.

Dit was alleen een inleiding om je op gang te helpen! Als je iets wilt nazoeken, doe dat één van de naslag-handleidingen.

6.5. Het Document Objectmodel (DOM), werken met objecten in Javascript

In Javascript werk je voortdurend met objecten. In de map "IntelligentForms" vind je "VerplichteVeldenCheck.htm" en daarin vind je deze statement:

```
if (document.forms[this.formnaam].elements[this.veldenArray[i]].value=="")
{ ...code... }
```

Hier wordt een waarde ("value") op het formulier vergeleken met "", de nullstring. De vraag is nu: hoe vindt het script de verschillende onderdelen van de pagina waarmee het iets gaat doen? In dit geval: hoe vindt het script de formulierelementen waaruit de waarde gelezen moet worden?

- Het zoeken begint bij het "document". "Document" verwijst naar het document waarin het script staat.
- Dit "document" heeft als eigenschap een "forms" array.
- Uit dit array halen we een form-object, door het bij de naam te noemen.
- Dat form-object heeft als eigenschap een "elements" array.
- Uit dat array halen we het juiste element, door het bij de naam te noemen. (Deze arrays hebben strings als indices, net als in PHP).
- Het element-object stelt een individueel invoerveld voor op het formulier. Een van de eigenschappen is "value", d.i. de huidige waarde in het veld.

Met de punt "." kies je dus een eigenschap van een object.

Je ziet hoe het hele document wordt voorgesteld door een boom van objecten, die via eigenschappen en arrays met elkaar verbonden zijn. In de voorbeeldcode hierboven is ook nog een zelfgemaakt object op te merken. Dit wordt aangeduid met het woordje "this". De objecten "document", "form" en "element" daarentegen zijn niet zelfgemaakt, maar deze worden beschikbaar gesteld door de browser en zijn beschreven in het Document Object Model.
Een overzicht van alle objecten uit de DOM vind je in Mozilla's DOM-referentie.
Een deel van de DOM is gestandaardiseerd, het betreft het document-object, en alle element-objecten. De standaard vind je natuurlijk bij W3C. Mozilla's DOM referentie sluit goed aan bij de standaard.
Internet Explorer volgt de standaard ook behoorlijk. In geval van afwijkingen, die er nog steeds zijn, heb je de eigen "HTML en DHTML" referentie van Microsoft nodig.
W3Schools heeft zoals gewoonlijk een lekker korte referentie waarin je snel iets kunt vinden.

6.6. Eigenschappen en methoden verkennen.

Kijk nog eens naar het "muistrucjes" voorbeeld en dan naar onMouseOver2.html.

- Welke objecten spelen een rol in de twee functies gedefinieerd in de `head`? Welke methoden en eigenschappen van die objecten worden gebruikt? Begrijp je nu hoe deze functies werken? (Zoek op wat je niet snapt).
- Bekijk ook onMouseOver1.html. Dit is geen goede XHTML maar ouderwets. Kijk hoe het trefwoord "this" naar een HTML-element verwijst, zodat het gaat functioneren als object in javascript. Hierdoor kun je de eigenschappen van het element (zoals de kleur) met javascript veranderen.

6.7. Interactie met de gebruiker verbeteren.

In de muistrucjes map vind je "ClickToSelect.html". Dit zou je kunnen gebruiken als interface om de gebruiker selecties te laten maken.

- Verbeter dit script zo, dat gemaakte selecties weer ongedaan gemaakt kunnen worden door er nog een keer op te klikken.

- Als output wordt het aantal geselecteerde regels gegeven. De output verschijnt alleen als je regel 4 selecteert, dit is onlogisch. Maak een knop (zie hieronder "formulieren") om de output op te vragen.

Hint 1: doe je voordeel met het "keuze"-array uit het voorbeeld.
Hint 2: let op het verschil tussen 'Red' en 'red' en een eventuele malligheid hiermee.

6.8. Een formulier intelligenter maken

Formulieren zijn vaak bedoeld om te worden ge-submit naar de server om daar verwerkt te worden door een server script, zoals je dat met PHP (of ASP of JSP) kunt maken. Een paar dingen die je vaak wilt bij formulieren zijn:

- controleren of alle verplichte velden zijn ingevuld,
- velden grijs (of onzichtbaar) te maken die door een gemaakte keuze van de gebruiker (eerder op hetzelfde formulier) niet ingevuld dienen te worden,
- opties aangeboden in een keuzelijst B laten afhangen van een eerder gemaakte keuze in keuzelijst A (bijvoorbeeld in A kies je een provincie en in B een stad in die provincie).

Simpele voorbeeldjes voor deze drie gevallen zijn te vinden in de map "IntelligentForms". Ze worden op het college gedemonstreerd en besproken.

6.9. Invoercontrole: Javascript of PHP?

Een goede webapplicatie zorgt voor passende foutmeldingen als de invoer van de gebruiker onvolledig of ongeldig is. In geval van een server-side script komt de invoer van een HTML-formulier en de controle kan op twee manieren gebeuren:

1) vóór verzenden van het formulier kan een controle met Javascript worden gedaan,
2) na verzenden kan PHP een controle uitvoeren.

Argumenten vóór en tegen controle met Javascript of PHP:

- Met Javascript krijgt de gebruiker sneller een reactie. Hij is bovendien zijn ingevulde formulier nog niet kwijt.
- Met PHP moet je moeite doen om de gebruiker zijn ingevulde formulier weer terug te geven, of de "back" knop moet worden gebruikt, of de gebruiker begint weer met een leeg formulier.
- Controle met Javascript ontlast de server.
- Controle met Javascript kan controle met PHP dikwijls niet geheel vervangen, omdat elk script een eigen URL heeft en dus ook zonder het bijbehorende formulier kan worden opgeroepen. Ook dan moeten rampen worden voorkomen.
- Als invoercontrole nodig is om fraude te voorkomen, dan is controle via een server-side script vereist. Een Javascript kan worden gekopieerd en gewijzigd om de controle te omzeilen.

6.10. Invoercontrole met Javascript

Om een formulier vóór verzenden te controleren met Javascript is het handig de standaard submit-knop (`<input type="submit">`) te vervangen, het formulier moet immers niet meteen worden ingediend. Hier is het recept[22]:

- Plaats een gewone knop (`<input type="button" ...>`) en koppel daaraan op de gewone manier een Javascript.
- Test (`if`) of de verplichte input-elementen elk een waarde bevatten en of deze waarden in het goede bereik liggen. (Zie hieronder voor mogelijke tests)
- Foutboodschappen kun je geven via een "alert"-window of in een formulierveld of (via "innerHTML") in de HTML van het document zelf[23].
- Als de controle geen fouten oplevert: eindig met een Javascript-methode om het formulier (object) te submitten:

```
document.formnaam.submit();
```

Hieronder een lijstje met suggesties voor nuttige tests.

Tabel 2: invoercontrole tests met Javascript

Test	Gebruik voor
`(x == "")`	Test of een invoer-element leeg is.
`(isNaN(x))`	Test of x géén getal is (NaN="Not a Number")
`(x > 3 && x < 5)`	Test of x tussen twee waarden ligt

6.11. Invoercontrole met PHP

In PHP is het meestal nodig te controleren of een verwachte variabele gedefinieerd is, dit is namelijk niet het geval als het script illegaal (niet via het bijbehorende formulier) is opgeroepen. Dat gebeurt met de functie `isset()`.
Een lijstje met suggesties voor validatietests volgt hieronder.

Tabel 3: invoercontrole tests met PHP

Test	Gebruik voor
`(isset(x))`	Test of x een waarde heeft
`(is_numeric(x))`	Test of x een getal is
`(is_integer(x))`	Test of x een geheel getal is
`(is_float(x))`	Test of x een decimale breuk is
`(is_string(x))`	Test of x een string is
`(x == "")`	Test of een string leeg is.
`(x > 3 && x < 5)`	Test of een getal x tussen twee waarden ligt

Meer informatie over deze functies is te vinden in de online PHP-naslag. Kijk onder "Function Reference → Variable handling →Variable handling Functions ".

---

[22] Alternatieve methode: houd de gewone <input type=”submit” ... /> knop. Voeg hieraan een onClick-event to met waarde “return *mijnjavascriptfunctie*()” en laat *mijnjavscriptfunctie* false teruggeven als de submit niet mag doorgaan.
[23] Zie sectie 6.14 voor details.

6.12. Bewegingen veroorzaken met Javascript.

In de map "/var/www/html/voorbeeldcode/Beweging" vind je een pagina "Bewegen.html".

- Probeer 'm uit en bekijk hoe die gemaakt is.

De basis van bewegingen is het `div`-element. Van dit HTML-element kun je de positie vrij instellen met de attributen `top` en `left`. Een script kan deze posities na een gekozen tijdinterval wijzigen (hier: elke 0,1 seconde – zie de setTimeout opdracht) en zo ontstaat een soort filmpje: elke 0,1 seconde een ander beeld.
De basis van bewegen is dus het positioneren. Als meerdere objecten een samenhangend beeld moeten geven (hier een soort wiel) kun je 't best het positioneren van de afzonderlijke objecten scheiden van de beweging van het geheel.

- Probeer en bekijk "Positioneren.html". Hier is alleen het positioneren gedaan, er beweegt nog niets. Kijk hoe het gelukt is om de `div`-objecten in een cirkel te rangschikken.
- Rangschik nu zelf een serie tekstjes[24] langs een rechte lijn die van rechtboven naar linksbeneden op je scherm loopt (`top` is de afstand tot de bovenrand, `left` de afstand tot de linkerrand, in pixels). Doe het met een formule, zoals ik voor de cirkel ook deed, dat werkt vlotter als het aantal elementen groot is.
- Kun je de lijn ook van linksboven naar rechtsbeneden laten lopen? Is het veel werk om die aanpassing te doen?
- Laat nu de hele lijn met tekstjes langzaam naar rechts verschuiven. Probeer het effect van verschillende timeout-tijden. (Voor de setTimeout functie kun je spieken in "Bewegen.html" hoe ik die gebruikte).

Steeds zoeken we eerst de nodige `div`-elementen (met getElementsByTagName). Het resultaat is een array waar we mee verder werken.

- Als je wilt kun je verder experimenteren: hoe maak je een ellips? Hoe laat je een rij objecten 'golven' alsof ze op de zee drijven?

6.13. Twee DOM-methoden om elementen te vinden

Om iets met een element te kunnen doen moet je het element kunnen vinden. De methoden die we hiervoor gebruiken behoren tot het gestandaardiseerde deel van het DOM. Een element opzoeken kan op twee manieren:

```
obj = document.getElementsById("id")
objArray = document.getElementsByTagName("tagnaam")
```

In het eerste geval krijg je een enkel object, in het tweede een array van objecten.
In het array staan de gevonden elementen in document-volgorde. Je kunt het array doorlopen met een FOR-loop, zoals gebruikelijk. Elk gevonden element kan het vertrekpunt zijn voor een nieuwe zoekactie. Dus je kunt ook doen:

```
objArray = obj.getElementsByTagName("tagnaam")
```

---

[24] Je kan mijn `div`-jes kopieren om overbodig typewerk te voorkomen!

Hier is "obj" een eerder gevonden element-object. Er wordt dan alleen gezocht in de afstammelingen van dat element: contextueel zoeken.
Er zijn ook eigenschappen en methoden om de directe kinderen van een element op te vragen, en om kinderen toe te voegen of te verwijderen.
Zie onder naslag: de “DOM Element Reference".

6.14. Tekstinhoud van een document aanpassen via het DOM

In sectie 6.5 zag je al hoe je eenvoudig een formulierelement kunt selecteren om daar de output van je script naartoe te sturen. Met behulp van DOM-methodes kun je elk gewenst element vinden en eventueel de inhoud ervan wijzigen.
Om een element te wijzigen kun je de `innerHTML` eigenschap van het element opvragen en een nieuwe waarde ervoor opgeven.

- Open uit de map "Beweging" het bestand "inhoudwijzigen.html"
- Dit bestand bevat een tabel met 3 rijen en 3 kolommen, en in cel 1 van rij 1 staat een stukje tekst. We gaan dit woordje via DOM-methoden opzoeken en dan verplaatsen naar een andere cel. (lijkt nutteloos, maar stel je voor: het script staat in een ander venster en wordt toegepast op een ander "vreemd" document. Het zoekt daar iets op en wijzigt het vreemde document op basis van wat het vond – dat is het idee.)
- Selecteer eerst het eerste `tr`-element met DOM-methodes. Vraag de innerHTML-eigenschap van dit element en toon het resultaat in een "alert" berichtvenster (het "alert" venster is niet mooi, maar prima voor debuggen of om tussenresultaten te checken).
- Gebruik de gevonden `tr` en selecteer daaruit via DOM-methodes het eerste `td`-element. Vraag weer de innerHTML eigenschap – dit zou het gewenste tekstje zijn dat we willen verplaatsen. Voor dit doel bewaar je 'm natuurlijk in een variabele.
- Via soortgelijke methodes kun je nu de cel selecteren waar je het tekstje naartoe wilt verplaatsen. Ga je gang!
- Hoeveel DOM-opdrachten heb je uiteindelijk nodig gehad? Hoeveel arrays met objecten heb je gemaakt? Is alles efficient gedaan?

Het had sneller gekund als de cellen elk gemerkt waren met een "id" attribuut. Dat hebben we niet gedaan, omdat we ons voorstellen dat we in een vreemde tabel iets opzoeken (het script kan dan de id’s ook niet weten).

6.15. Scherm, venster en browser eigenschappen bepalen

Met Javascript kun je ook eigenschappen van de browser opvragen via het “Browser Object Model” (BOM). Dit is handig als je de weergave van een webpagina wilt aanpassen aan de situatie van de gebruiker.
Schermeigenschappen vraag je op met het "screen" object (in pixels), bijvoorbeeld:

```
screen.width
screen.height
```

Je kunt ook vragen naar `availWidth` en `availHeight`. Er wordt dan gecorrigeerd voor bijvoorbeeld de Windows taakbalk.
De grootte van het venster – de gebruiker kan dit immers resizen – kun je opvragen als eigenschappen van `document.body`:

```
document.body.clientWidth
document.body.clientHeight
```

Browser en besturingssysteem bepaal je met het "navigator" object:

`navigator.appName`	geeft de naam van de browser
`navigator.appVersion`	geeft versie-informatie, inclusief besturingssysteem
`navigator.userAgent`	geeft ook versie informatie, anders gepresenteerd
`navigator.vendorSub`	het versienummer
`navigator.platform`	geeft het besturingssysteem waarvoor de browser gecompileerd is

De "appName" eigenschap geeft als resultaat bijvoorbeeld "Netscape" (voor Firefox!), of "Microsoft Internet Explorer". De "platform" eigenschap geeft "Win32" voor diverse soorten Windows, "MacPPC" voor MacIntosh OS 9.0.4.

De "userAgent"-eigenschap geeft een string waarbij wat uitleg nodig is. De browser verstuurt deze string naar elke server die je bezoekt. Die informatie kan door de server gebruikt worden om een aangepaste pagina te genereren. Voor Firefox krijg je een resultaat dat er zo uit kan zien:

Mozilla/5.0 (Windows NT 6.1; WOW64; rv:8.0.1) Gecko/20100101 Firefox/8.0.1

Hier geeft Mozilla/5.0 aan wat de compatibiliteit is met de Mozilla rendering engine. Tussen haakjes volgen karakateristieken van het besturingssysteem en soms de taalversie van de browser. Aan het eind staan de versie van de layout engine en de werkelijke versie van de browser.

Ook andere browsers noemen allereerst het versienummer van Mozilla waar zij compatibel mee zijn. Dus Firefox kondigt zichzelf aan als versie 5.0 van Mozilla en Internet Explorer kondigt zich aan als versie 4.0 van Mozilla. De volgende stringfuncties zijn nuttig bij het ontcijferen van de appVersion of userAgent eigenschap:

parseFloat(*str*)	als de string *str* begint met een getal, dan wordt dit getal geretourneerd
str1.indexOf(*str2*)	*str1* wordt doorzocht op aanwezigheid van *str2*, indien gevonden wordt de beginpositie van *str2* geretourneerd (> 0) anders: –1.
str1.substring(*n, m)*	geeft een string met de tekens op posities *n* tot, maar niet inclusief, *m* uit de string *str1*
str1.length	geeft het aantal tekens in string *str1*

Meer informatie over deze functies vind je in de DevGuru referentie-handleiding die te vinden is onder "naslag" op de cursus website, en in de DOM-handleiding van Mozzilla.

6.16. Stylesheets en Javascript

Bij het aanpassen van de opmaak (o.a. lettertypen en lettergrootte) aan verschillende besturingssystemen en browsers zijn twee strategieën gangbaar:

1) verschillende stylesheets voor verschillende situaties (besturingssysteem / browser) maken, dan Javascript gebruiken om één van deze stylesheets aan het document te linken.

2) één CSS stylesheet maken en dan met Javascript enkele regels aan deze sheet toevoegen (voor het huidige document) dan wel bestaande regels overrulen, om deze aan te passen aan besturingssysteem en browser.

Elk van deze methoden wordt hierna toegelicht.

6.17. Stylesheets via Javascript linken

Met Javascript kunnen ook nieuwe stylesheets gelinkt worden door in het document tijdens het laden een `link`-element te schrijven met `document.writeln`, bijvoorbeeld zo:

```
<script type="text/javascript">
     if (...) {
          document.writeln('<link rel="stylesheet" type="text/css"
          href="mijnsheet" />');
          }
</script>
```

Javascript tussen <script> tags wordt uitgevoerd meteen als het wordt geladen. De opdrachten `write` en `writeln` schrijven hun output direct op de plaats waar deze opdrachten staan. Daarbij verdwijnt het script zelf uit de source-code. Dus als bovenstaande reeks voorkomt in een document dan verdwijnt het script en komt er een `link`-element voor in de plaats.
Dit werkt ook in oudere browsers. Als soms het <link> element niet wordt gehoorzaamd (heb ik wel eens meegemaakt...) kun je in plaats daarvan het <style> element proberen:

```
<style type="text/css" src="mijnsheet"></style>
```

6.18. Stylesheets via Javascript maken of aanpassen

In plaats van stylesheets met de hand te maken kun je ook javascript gebruiken om “on the fly” stijlregels toe te voegen aan een bestaande stylesheet van een document. Deze stijlregels kunnen bestaande stijlregels overrulen.

Volgens de DOM standaard gaat het zo:

```
aantal = document.styleSheets.length;
with (document.styleSheets[aantal-1]) {
     hoogsteregel = cssRules.length-1;
     insertRule("div.jan {color: Black; background-color: Red;}",
          hoogsteregel + 1);
     insertRule("h1 {font-family: Arial; }",
          hoogsteregel + 2);
}
```

Omdat een document aan meer dan één stylesheet gelinkt kan zijn, is het nuttig om eerst de stylesheet met het hoogste nummer te bepalen. Stijlregels in deze sheet hebben voorrang op regels in andere sheets. Het is meestal deze sheet die je wilt wijzigen.

Vervolgens kijken we hoeveel regels deze stylesheet al heeft. We plaatsen de twee nieuwe regels aan het eind, dus we geven ze volgnummers die hoger zijn dan bestaande regels in de sheet. Ook dat zorgt weer dat deze regels eventueel bestaande regels overrulen. Als je dat niet wilt kun je ze natuurlijk lager in de stylesheet zetten.

In het voorbeeld hierboven worden twee regels toegevoegd aan de stylesheet.

Je kunt ook bestaande regels wissen, mits je het regelnummer kent. Zie het voorbeeld hieronder:

```
aantal = document.styleSheets.length;
with (document.styleSheets[aantal-1]) {
     deleteRule(0);
     deleteRule(0);
}
```

Nadat een regel gewist is, schuiven alle hogere regels één naar onderen. Ik heb hier dus de regels gewist die in het begin op posities 0 en 1 stonden.

Dit alles werkt wel goed in Mozilla, maar niet in oude versies van Internet Explorer. IE heeft echter eigen commando's (niet volgens de standaard) waarmee je hetzelfde resultaat kunt bereiken. Kijk naar “/var/www/html/voorbeeldcode/Javascript/DynamicStyles" voor een voorbeeld hoe je zowel met IE als met Mozilla compatibel kunt zijn.

Er is nog een derde methode. Je kunt, buiten de stylesheet om, direct stijlen toekennen aan elementen. Dit is een wat bewerkelijke methode, maar werkt in IE en Mozilla precies hetzelfde:

1) Je gebruikt bijvoorbeeld `getElementsByTagName` om alle elementen van een bepaalde soort te selecteren.
2) Dat levert een array die je met een `for`-loop kunt doorlopen om elementen met een bepaalde `classname` eruit te halen.
3) Telkens als je er één vindt ken je de gewenste eigenschap eraan toe.

Een voorbeeld van deze methode vind je in dezelfde map "DynamicStyles".

Het gaat een stuk vlotter als je op id wilt selecteren in plaats van op een klasse. De for-loop kan dan weg en met "getElementById" heb je snel het gewenste element.