Inhoud

12 jQuery en Ajax 255

1 Hoofdstuk

Kennismaken met JavaScript

HTML is al ruim twintig jaar de standaard voor het opmaken van pagina's op het web. HTML kan echter niet alles. Het is vooral een taal waarmee de structuur van pagina's wordt beschreven. In de loop der jaren zijn allerlei uitbreidingen ontwikkeld om de mogelijkheden van HTML te verbreden, met JavaScript als belangrijkste product. JavaScript is zonder twijfel de populairste programmeertaal op internet. Elke browser heeft een JavaScript-motor, waardoor moderne webapps mogelijk worden. In dit inleidende hoofdstuk maakt u kennis met enkele algemene kenmerken van JavaScript en leert u welke tools u nodig hebt om succesvol met JavaScript aan de slag te kunnen gaan. Natuurlijk schrijft u alvast een eerste JavaScript voor snel resultaat.

U leert in dit hoofdstuk:

Een korte geschiedenis van JavaScript.

Waarvoor JavaScript wordt gebruikt.

Welke belangrijke begrippen u moet kennen bij het werken met JavaScript.

Welke tools u nodig hebt bij het programmeren.

Hoe JavaScript en HTML gecombineerd worden in webapps.

Een eerste script schrijven en de tags <script>…</script>.

Kennismaken met JavaScript-debugging.

Een korte geschiedenis van JavaScript

JavaScript is oorspronkelijk in 1995 ontwikkeld door Brendan Eich, die bij Netscape werkte. Netscape is het bedrijf dat een van de oerbrowsers voor internet maakte, Netscape Navigator. Aanvankelijk werd als naam Mocha ('mokka') gekozen, maar al snel daarna koos Netscape voor de naam LiveScript. Nog weer later werd dit veranderd in JavaScript. Ook toen al was JavaScript bedoeld als uitbreiding van HTML om meer interactiviteit op webpagina's mogelijk te maken. De combinatie van HTML en JavaScript stond destijds bekend onder de naam *Dynamic HTML* (DHTML).

De browsers uit die tijd (Internet Explorer 4 en Netscape 4) boden ondersteuning voor de combinatie van HTML en JavaScript in webpagina's. Ondertussen zijn we natuurlijk vele browserversies verder, maar nog steeds is JavaScript een erg belangrijke pijler binnen de browser. Alle huidige browsers (Internet Explorer, Firefox, Chrome, Safari enzovoort) kunnen JavaScript uitvoeren.

Versies van JavaScript

In de loop der jaren is het versienummer van JavaScript steeds licht opgehoogd. Eerst liep het versienummer omhoog met het verschijnen van een nieuwe browserversie, maar nu is de ontwikkeling van JavaScript losgekoppeld van nieuwe versies van de browser. We geven enkele belangrijke mijlpalen:

- **JavaScript 1.0** Uit 1996, was aanwezig in Netscape Navigator 2.0 en Internet Explorer 3.0;
- **JavaScript 1.1** Uit 1996, opgenomen in Netscape Navigator 3.0;
- **JavaScript 1.3** Uit 1998, opgenomen in Netscape Navigator 4.06 en Internet Explorer 4;
- **JavaScript 1.5** Uit 2000, dit was de eerste versie van ECMAScript (zie verderop), opgenomen in Netscape Navigator 6.0, Mozilla Firefox 1.0 en Internet Explorer 5.5 en hoger;
- **JavaScript 1.6** Uit november 2005, met extra mogelijkheden voor arrays en strings.

Op het moment van schrijven van dit boek is JavaScript 1.8.5 uit juli 2010 de nieuwste versie, maar vrijwel niemand let nog echt op het versienummer. Deze versie bevat alle mogelijkheden van ECMAScript 5 en maakt deel uit van Firefox 4 en hoger, Internet Explorer 9 en hoger en Chrome 10 en hoger.

ECMAScript

De term ECMAScript is al even gevallen. JavaScript is inmiddels omgevormd tot een officiële, genormeerde en gestandaardiseerde programmeertaal. Dit is gedaan door de structuur en inhoud van de taal te laten goedkeuren en standaardiseren door de organisatie European Computer Manufacturers Association (ECMA). Hiermee is de ontwikkeling van JavaScript nu in handen van een onafhankelijk instituut en niet meer van één browserfabrikant.

Wat doet ECMA?

ECMA heeft tot taak standaarden te publiceren en hierop toe te zien. ECMA houdt zich niet alleen bezig met JavaScript, maar ook met andere computergerelateerde standaarden, zoals het bestandsformaat voor cd-rom's, de specificaties van de programmeertaal C# en het bestandsformaat Office Open XML. Wilt u hier meer over weten, bezoek dan **www.ecma-international.org**.

De JavaScript-specificatie is vastgelegd in een document met het nummer 262 (en daarvan inmiddels de vijfde versie). Daarom heet JavaScript officieel *ECMAScript-262 5th Edition*. Maar voor het gemak heeft iedereen het altijd gewoon over JavaScript.

Alle bekende browsers ondersteunen JavaScript. In Internet Explorer heeft JavaScript de naam JScript (en kent het een iets afwijkende versienummering), maar de functionaliteit is verder grotendeels gelijk.

Waarvoor wordt JavaScript gebruikt?

We hebben al globaal aangegeven dat JavaScript wordt gebruikt om gedrag of interactie aan webpagina's toe te voegen. Meer specifiek kent JavaScript de volgende toepassingen binnen de browser:

- **Formuliervalidatie** JavaScript is erg geschikt om de ingevulde gegevens in een webformulier op een pagina te controleren voordat het formulier wordt verzonden. Omdat deze controle op de computer van de gebruiker plaatsvindt, gaat dit veel sneller dan controle op de webserver na het versturen. Ook wordt de server minder belast, waardoor er meer capaciteit is voor andere gebruikers. Door het gebruik van JavaScript is er geen *round-trip* nodig naar de server en kan via een lokale melding direct worden aangegeven dat iemand bijvoorbeeld een ongeldig e-mailadres heeft ingevuld.
- **Dynamische menu's en afbeeldingen** Met JavaScript kunnen menu's en afbeeldingen tijdens het gebruik van de pagina worden vervangen. Dit kan bijvoorbeeld van pas komen bij fotocarrousels of uitklapmenu's.

- **Aanpassingen van stijlen en animatie** Als een webpagina in de browser is geladen, kan met JavaScript de aanwezigheid, positie en inhoud van elk element op de pagina (teksten, afbeeldingen enzovoort) worden opgehaald en gemanipuleerd. Zo kunnen bijvoorbeeld kaders op een pagina vloeiend open- en dichtschuiven, menu's dynamisch worden uitgebreid, muisklikken van de rechtermuisknop worden afgevangen en aangepast en zo verder. Er zijn tal van kant-en-klare JavaScript-bibliotheken beschikbaar waarin al vele animatiefuncties zijn voorgeprogrammeerd. Deze kunt u op de pagina laden en (bijna) direct gebruiken.
- **Ajax-webapplicaties** U hebt misschien de term Ajax wel eens gehoord, het staat voor Asynchronous JavaScript And XML. Dit wil zeggen dat na het laden van de pagina asynchroon delen van de pagina ververst of aangepast kunnen worden door gebruik te maken van XML en JavaScript. Het hele idee van applicaties op het web zoals Facebook, Gmail, Twitter en Hotmail is gebaseerd op gegevensuitwisseling op de achtergrond (asynchroon) met JavaScript en XML. Zonder JavaScript zou het web in zijn huidige vorm niet bestaan!

Kernbegrip – JavaScript core

Het is belangrijk om te weten dat de *taal* JavaScript uit een relatief kleine set instructies bestaat. Er zijn opdrachten om te werken met variabelen, lussen, teksten, arrays en objecten, maar verder is er niet zo veel bijzonders. JavaScript bevat bijvoorbeeld geen opdrachten voor invoer en uitvoer, er zijn geen netwerkmogelijkheden of mogelijkheden voor het werken met bestanden. In andere programmeertalen zoals Java, PHP of C# zijn dergelijke zaken wel opgenomen.

Dergelijke uitgebreide handelingen zijn functies die worden overgelaten aan de zogenoemde *hosting environment*. In een internetomgeving is de host de webbrowser. JavaScript draait binnen de browser en maakt zodoende gebruik van het browservenster om teksten te tonen. Als JavaScript communiceert met een script op de webserver, maakt het gebruik van de browsermogelijkheden om netwerkverbindingen en (Ajax-)aanroepen op te zetten. De taal JavaScript zelf biedt hiervoor geen voorzieningen.

Indeling van dit boek

Dit boek bestaat uit twee delen:

- **Deel 1 – Kernmogelijkheden van JavaScript** De hoofdstukken 1 tot en met 6 gaan over de kernmogelijkheden van JavaScript. U leert de taal goed kennen door eenvoudige programma's te schrijven met de gereserveerde

JavaScript-woorden. U maakt kennis met variabelen, lussen en overige JavaScript-syntaxis. Dit is altijd de kern van elke programmeertaal. JavaScript is hierop geen uitzondering. Als u deze onderdelen goed beheerst, kunt u JavaScript in tal van omgevingen toepassen. In de browser, op webservers en misschien nog in andere omgevingen. JavaScript kan bijvoorbeeld ook gebruikt worden om interactieve PDF-documenten te scripten.

- **Deel 2 – Werken met jQuery** In de hoofdstukken 7 tot en met 14 gebruikt u de kennis uit deel 1 om met JavaScript het DOM in de browser te programmeren en jQuery te gebruiken. jQuery is een uitbreiding van JavaScript en biedt opties om met weinig code in alle browsers een goed resultaat te bereiken. jQuery is echter geen vervanging van JavaScript. Basiskennis van JavaScript zelf is beslist een vereiste. U leert jQuery zodat u snel onderdelen van de webpagina kunt tonen of verbergen of met uitgebreide onderdelen van de user interface kunt werken.

Alle hoofdstukken zijn zo veel mogelijk verduidelijkt met praktijkvoorbeelden en schermafbeeldingen.

Meer dan de browser

Webbrowsers zoals Chrome, Firefox, Internet Explorer en Safari zijn zonder twijfel de bekendste omgevingen om JavaScript-toepassingen uit te voeren. Maar er zijn meer varianten van JavaScript. Omdat het een gestandaardiseerde taal is, zijn er veel afgeleiden ontwikkeld. Zo zijn Adobe Flex en Flash ActionScript ook dialecten van JavaScript. Daarbij is Flash Player de hosting environment waarin het programma wordt uitgevoerd. Inmiddels is node.js een bekende omgeving om JavaScript op de server uit te voeren (zie **nodejs.org** voor meer informatie). En ook in interactieve PDF-documenten kan JavaScript worden geschreven. Kortom, de browser is zonder twijfel de bekendste, maar zeker niet de enige omgeving waarin JavaScript wordt ingezet. In dit boek gebruiken we overigens altijd een webbrowser, omdat die voor iedereen die met JavaScript aan de slag wil direct toegankelijk is. U hoeft er niks extra voor te installeren of te downloaden.

Oefenbestanden downloaden

Alle codevoorbeelden en -fragmenten die in de tekst worden genoemd zijn als voorbeeldbestanden te downloaden. Het adres hiervoor is **www.kassenaar.com/hbjs**. Dit brengt u naar het blogartikel over dit boek. Ongeveer halverwege de pagina vindt u de link om de voorbeeldbestanden te downloaden.

Soms gaat het maar om enkele regeltjes code, maar dat kan net genoeg zijn om u op weg te helpen of om als startpunt te dienen voor uw eigen experi-

menten. De voorbeelden zijn verdeeld in mappen per hoofdstuk. In een aantal algemene mappen zoals \css en \script staan aanvullende stijlbestanden en de gebruikte versie van jQuery.

Handige voorkennis

Kan iedereen JavaScript gebruiken? Worden aan de JavaScript-programmeur speciale eisen gesteld? We geven kort aan welke voorkennis nodig is en op welke wijze u uw kennis eventueel kunt bijspijkeren.

Om JavaScript te kunnen gebruiken is in principe heel weinig voorkennis nodig. JavaScript staat als leesbare platte tekst in de broncode van het webdocument of in een apart (gekoppeld) scriptbestand. Het is dus voldoende als u in het besturingssysteem de functies knippen en plakken kunt activeren om de scripts van andermans pagina naar de uwe te kopiëren. Maar dat was waarschijnlijk niet wat u in gedachten had, dus gaan we iets dieper in op de randvoorwaarden om succesvol met JavaScript aan de slag te kunnen gaan.

Omdat JavaScript een programmeertaal is, is het zonder meer een voordeel als u enige ervaring hebt met programmeren. Zelfs met uitsluitend wat basiskennis van PHP, Java of het schrijven van macro's voor Office hebt u al een voorsprong bij het programmeren in JavaScript. De statements, de code, de controlestructuren en de syntaxis zult u wat sneller onder de knie kunnen krijgen.

Hebt u eerder vooral andere scripts en jQuery-codes gekopieerd en geplakt, dan leert u nu eindelijk wat al die haakjes, puntkomma's en accolades betekenen.

Bekendheid met HTML en CSS

We gaan er wel van uit dat u bekend bent met HTML, de opmaaktaal van webpagina's. Als u vertrouwd bent met tags, id's, attributen en de andere elementen waaruit een webpagina is opgebouwd, zult u deze snel kunnen aanpassen om ze zo met behulp van JavaScript op de pagina te manipuleren.

Het DOM, elementen, formulieren en afbeeldingen spelen een grote rol in JavaScript. Ook deze geavanceerde HTML-elementen zult u moeten beheersen. Is dat niet het geval, lees dan eerst een ander boek, waarin de basisbeginselen van HTML uiteengezet worden, bijvoorbeeld *Handboek (X)HTML, CSS en JavaScript* van dezelfde auteur. In dit boek staan we niet verder stil bij de HTML- en CSS-syntaxis van elementen. Zij worden bekend verondersteld.

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>Titel van het document</title>
<style>
    /* alle stijlen in het document */
</style>
</head>

<body>
<div id="container">
    <!-- inhoud van het document -->
</div>
</body>
</html>

```

Afbeelding 1.1 *Bekendheid met de notatie van HTML en CSS is een vereiste. We gebruiken in dit boek het HTML5-documenttype.*

Bekendheid met CSS is vooral handig als u met JavaScript of jQuery het uiterlijk van elementen op de pagina wilt aanpassen. Niet alleen de CSS-eigenschappen voor kleuren, randen, lettertype en meer kunt u gebruiken, maar ook de CSS-syntaxis om elementen op de pagina te selecteren. Hier gaan we vooral in de jQuery-hoofdstukken op in.

Wat hoeft u niet te weten?

U hoeft niet iets te weten van serversided programmeertalen zoals PHP, Java of C#. Noch hoeft u op systeembeheerderniveau beschikking te hebben over een webserver. In dit boek gaan we zoals gezegd uit van clientsided JavaScript, wat betekent dat scripts op letterlijk elke pagina gebruikt kunnen worden. Het script maakt deel uit van de webpagina. In de meeste gevallen is het zelfs niet nodig dat u online bent voor het uitvoeren van de oefeningen. Een editor en een browser zijn veelal voldoende.

Ajax – alleen op de server

Gaat u werken met Ajax, dan gebruikt u JavaScript om vanuit de pagina te communiceren met een webserver. Als het zover is, geven we dat echter duidelijk aan. Ajax-requests kunt u alleen maar uitvoeren in combinatie met een webserver. Op het bestandssysteem (met het protocol file://) werkt dat niet.

U hebt nog niet eerder geprogrammeerd

Ook als u alleen HTML-voorkennis hebt en nooit eerder hebt geprogrammeerd, is er geen man overboord. JavaScript is wellicht niet de eenvoudigste taal, maar zeker niet zo moeilijk als Java of C/C++. In JavaScript is het heel eenvoudig om kleine stukjes code te schrijven, apart te testen en deze later samen te voegen tot een compleet werkend script. In de meeste 'hogere' programmeertalen is dit niet mogelijk. Daar moet u bijvoorbeeld minstens een interface schrijven om een functie te kunnen testen. In JavaScript is dat niet nodig. De browser neemt immers de elementaire zaken van de interface voor zijn rekening.

Maar, eerlijk is eerlijk. Als u tot nu toe vooral met web*design* bezig bent geweest, wordt de overstap naar JavaScript vaak als lastig ervaren. U betreedt nu het domein van de web*developer*. U wordt programmeur, in plaats van ontwerper.

Syntaxis

De eerste (en wellicht grootste) hindernis die u tegenkomt bij het programmeren in JavaScript is de volgorde waarin de woorden, getallen, instructies en statements moeten worden geschreven. De regels voor deze volgorde noemen we de *syntaxis*. De syntaxis van een programmeertaal is te vergelijken met de grammatica van een gewone taal: als u wel alle afzonderlijke woorden van het Nederlands kent maar ze niet in een voor een gesprekspartner logische volgorde kunt plaatsen, is een zinnig gesprek niet mogelijk.

Hetzelfde geldt voor JavaScript. U moet niet alleen de juiste notatiewijze van de code kennen, u moet deze ook in de juiste volgorde schrijven. Hierin is JavaScript – zoals elke programmeertaal – erg streng. Een verkeerd geplaatste puntkomma of een vergeten haakjesluiten kan er al voor zorgen dat het script niet werkt. In spreektaal luistert de volgorde van de woorden meestal niet zo nauw; in het algemeen begrijpen wij wel wat de ander bedoelt. Bij programmeren, waar de computer de gesprekspartner is, lukt dat niet. Als de syntaxis niet perfect is, stopt de machine subiet met het uitvoeren van het script en kan hij niets anders verzinnen dan ons mede te delen dat er een fout in het script zit. Als u geluk hebt, toont de browser nog waar de fout zich bevind, zodat u hem snel kunt herstellen, maar niet alle browsers kunnen dit. Deze fouten maakt iedereen, zelfs ervaren programmeurs, u kunt ze dan ook het best als leerzaam beschouwen.

```
234 //-----------------------------------
235 // Alle opdrachten voor de huidige user ophalen via webservice en tonen op de homepage
236 //-----------------------------------
237 function getOpdrachten() {
238     // 0. Test of de user wel onLine is
239     if (!navigator.onLine) {
240         $.mobile.hidePageLoadingMsg();
241         alert(ic.offlineMessage);
242         return;
243     }
244     // alle inspecties ophalen, URL staan in de globale configuratieparameter ic
245     console.log('start opdrachten ophalen via webservice');
246     var url = ic.urls.user_behandeling + ic.username;
247     $.ajax({
248         type: 'GET',
249         url: url,
250         success: function (data, textStatus, jqXHR) {
251             // 1. Opdrachten voor huidige user succesvol opgehaald
252             console.log('Alle opdrachten ophalen: success"!');
253             if (jqXHR.status != ic.HTTPconstants.RETURN_STATUS_GEEN_INHOUD) {
254                 var opdrachten = JSON.parse(data);
```

Afbeelding 1.2 *Programmeerervaring is wel handig, maar geen must. Aan het einde van het boek kunt u zelf functies als deze schrijven.*

Ook als u nog geen enkele programmeerervaring hebt, kunt u rustig de rest van dit boek doornemen. Besteed veel aandacht aan de codevoorbeelden en de oefeningen, zodat u zich snel de specifieke wijze eigen maakt waarop JavaScripts geschreven worden.

U hebt al programmeerervaring

Misschien hebt u al eerder geprogrammeerd of bijvoorbeeld macro's voor Word geschreven. Dat is zonder meer een voordeel als u met JavaScript aan de slag gaat. Als u ervaring hebt met bijvoorbeeld programmeren in Java of PHP, is het nut van de correcte syntaxis en wellicht de notatiewijze van verschillende statements bekend. In JavaScript hoeft u echter geen user interface te programmeren. De browser is de interface. U werkt veelal met objecten en functies.

Objectgeoriënteerd programmeren in JavaScript?

Strikt genomen is JavaScript niet een echte objectgeoriënteerde programmeertaal. JavaScript ontbeert enkele typische kenmerken die we in hogere objectgeoriënteerde programmeertalen zoals C# en Smalltalk wel tegenkomen. JavaScript kent bijvoorbeeld geen strikte klassenhiërarchie, waardoor objecten die van een bepaalde klasse zijn afgeleid, automatisch de eigenschappen en methoden van het hoger gelegen object overerven (*multiple inheritance*). Ook zijn functies van een object niet goed af te schermen voor de buitenwereld. Er is geen strikt onderscheid tussen publieke en private methoden. Omdat JavaScript echter wel gebruikmaakt van de objecten en methoden die in de browser aanwezig zijn, spreken we in dit boek voor het gemak over *JavaScript-objecten* en *objectgeoriënteerd*. Dit verhoogt de duidelijkheid.

Objectgeoriënteerd programmeren

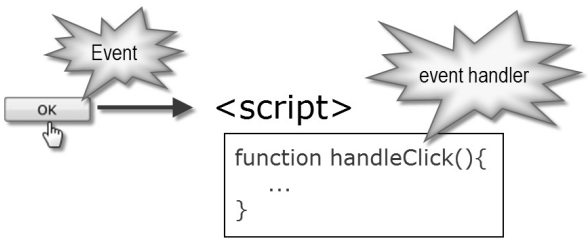
In JavaScript werkt u veel met objecten – laten we het dan toch maar zo noemen. Dit kunnen objecten zijn die u zelf maakt in script (bijvoorbeeld een object `user` met eigenschappen als `name`, `password`, `age` en meer), maar ook elementen die in de pagina worden getoond. (DOM-elementen). U kunt ze in de pagina selecteren en met JavaScript besturen. Denk aan afbeeldingen, formulieren, de inhoud van div's, enzovoort.

Objecten in het Document Object Model (DOM) van de browser zijn zelfstandige elementen die kunnen reageren op gebeurtenissen die optreden. Stel dat u een HTML-knop hebt aangegeven met `<input type="button" id="btnKnop1">`. Deze knop is in de pagina een object dat zelfstandig gebeurtenissen kan waarnemen en daar adequaat op kan reageren. Het object (in dit geval de knop) heeft eigenschappen zoals hoogte, breedte, positie, tekst, enzovoort. Bovendien kan het object zelfstandig reageren als er bepaalde gebeurtenissen (*events*) optreden. Deze gebeurtenissen zijn bijvoorbeeld een muisklik of een mouseover, maar ook het verslepen van een element (*drag-and-drop*), het selecteren van tekst, enzovoort.

Events en event handlers

Als de gebruiker op de knop klikt (of hij tikt erop in een touchscreenbrowser, zoals op een tablet of mobiele telefoon), dan kan een script in actie komen. Het script is gekoppeld aan de knop en bevat de instructies die worden uitgevoerd zodra de knop door de gebruiker wordt geactiveerd.

Preciezer gezegd, de gebruiker genereert de *event* `click`, waarna door het besturingssysteem de *event handler* wordt geactiveerd. Een event handler is een functie die de event afhandelt. Deze schrijft u zelf. Vooraf moet u aangeven dat de browser luistert of de betreffende gebeurtenis (het klikken op de knop) optreedt. Dit heet ook wel het aanhaken van een *event listener*. In hoofdstuk 6 komen we hier nog uitgebreid op terug.



Afbeelding 1.3 *JavaScript in de browser draait voor een groot deel om het luisteren naar en afvangen van events. Vervolgens schrijft u script om iets nuttigs te doen.*

Het script zelf bevat meestal een aantal instructies die volgens de regels van de taal JavaScript in een bepaalde volgorde worden uitgevoerd. Een eenvoudig voorbeeld: u hebt een webfotoalbum geprogrammeerd. Met een klik op de knop kan de gebruiker de volgende foto in het album laden. Het script wijzigt daartoe het attribuut `src="..."` van de tag `<img>` (bijvoorbeeld van `foto1.jpg` naar `foto2.jpg`).

Bij objectgeoriënteerd programmeren kunt u als programmeur evenwel niet voorspellen wanneer een bepaalde gebeurtenis zal optreden. Misschien vult de gebruiker een formulier van achteren naar voren in, of klikt hij eerst op de 'verkeerde' knop (anders dan u als programmeur had bedacht). De gebruiker bepaalt. Het is de taak van de programmeur om voor elke event die kan optreden een juiste event handler te schrijven (aangenomen dat hij de event wil gebruiken). Als programmeur moet u anticiperen op al die 'vervelende gebruikers' die niet snappen wat u als programmeur zo duidelijk hebt bedoeld.

Ontwikkelhulpmiddelen voor JavaScript

JavaScript is gewoon platte tekst. In principe zou u dus met Kladblok (Windows) of Teksteditor (Mac) al aan de slag kunnen. Maar in de praktijk is het wel erg handig om met een gespecialiseerde editor aan de slag te gaan. We noemen er in deze paragraaf enkele, zodat u zelf een keuze kunt maken. Naast het schrijven van JavaScript is het opsporen en verhelpen van fouten in een script erg belangrijk. Hiervoor zijn inmiddels gelukkig ook goede hulpmiddelen beschikbaar. Deze komen aan het eind van de paragraaf aan de orde.

WYSIWYG-editors ongeschikt

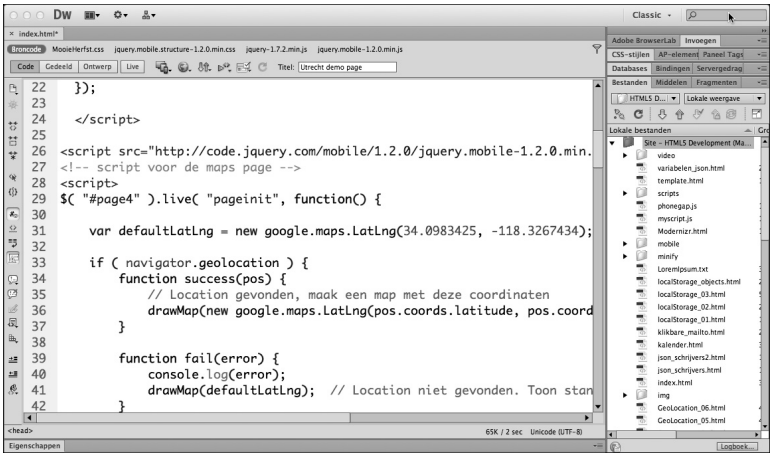
Webontwikkelaars die uit de vormgevingshoek afkomstig zijn, gebruiken graag een editor waarmee webpagina's op visuele wijze worden vormgegeven. Zij werken bijvoorbeeld in de Ontwerpweergave van Adobe Dreamweaver of gebruiken aparte tools als Adobe Muse of Microsoft Expression Web. Het samenvattende begrip hiervoor is What You See Is What You Get-editors (*WYSIWYG*). Voor het schrijven van JavaScript zijn die niet geschikt. U zult echt met de code zelf aan de slag moeten om de beste resultaten te bereiken. In dit boek worden visuele editors niet gebruikt.

Adobe Dreamweaver

Een bekende tool is Adobe Dreamweaver. Onder webdesigners die veel met andere Adobe-toepassingen werken, zoals Photoshop of InDesign, is dit vrijwel de automatische keuze. Dreamweaver biedt uitstekende ondersteuning voor

JavaScript-kleurcodering, geeft codehints en kan ook helpen bij het werken met jQuery. Ook hiervoor zijn coderingshints en aanwijzingen voor het gebruik aanwezig.

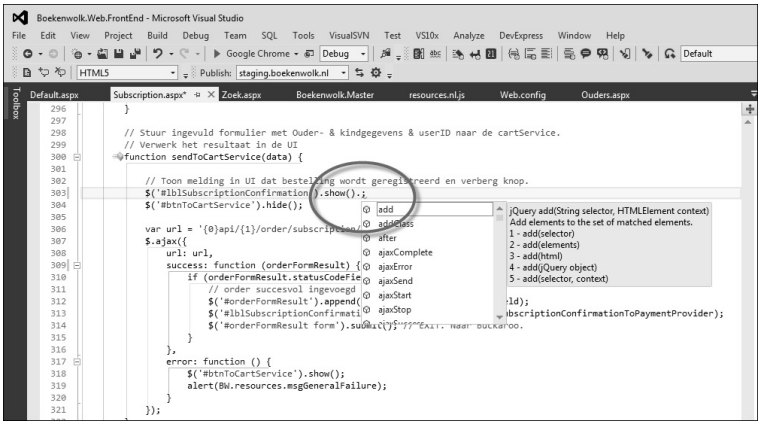
Dreamweaver is niet goedkoop, maar het is een zeer goede keuze als u serieus met zowel HTML, CSS als JavaScript en jQuery aan de slag wilt. Er is een probeerversie beschikbaar op **www.adobe.com/go/trydreamweaver**. Op het moment van schrijven van dit boek was Dreamweaver CS6 de meest recente versie. Dreamweaver is beschikbaar voor Windows en Macintosh en is ook verkrijgbaar in het Nederlands.



Afbeelding 1.4 *Adobe Dreamweaver is een professionele toepassing voor webdesign. Het prijskaartje is er dan ook naar.*

Microsoft Visual Studio

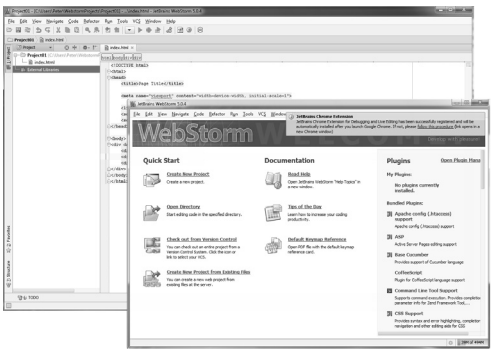
Als u meer in de Microsoft-hoek zit, ligt het gebruik van Visual Studio voor de hand. Visual Studio is het vlaggenschip voor productontwikkeling van Microsoft. U kunt er Windows-toepassingen mee maken en apps voor Windows Phone, Office en Sharepoint, maar Visual Studio is ook erg geschikt voor webontwikkeling. De functies voor code completion, code hints, configuratie en uitbreiding behoren tot de beste die er zijn. Visual Studio is niet in het Nederlands verkrijgbaar. Wel is er een gratis versie beschikbaar. Visual Studio 2012 Express for Web was op het moment van schrijven de meest recente versie. De betaalde versies heten bijvoorbeeld Visual Studio 2012 Professional of Ultimate. Download uw versie van **www.microsoft.com/visualstudio/eng/products/visual-studio-express-for-web**.



Afbeelding 1.5 *Microsoft Visual Studio Express is gratis en bevat zo goed als alle mogelijkheden van de betaalde variant. Handig zijn de opties om automatisch veelgebruikte functies aan te vullen. Ze verschijnen in een tooltip.*

JetBrains WebStorm

Behalve de onvoorstelbaar uitgebreide totaaloplossingen zoals Dreamweaver en Visual Studio zijn er ook toepassingen die zich profileren als 'bij uitstek geschikt voor JavaScript'. Deze programma's zoeken het dus in de specialisatie in plaats van in de verbreding. Een voorbeeld hiervan is JetBrains WebStorm (**www.jetbrains.com/webstorm/**). Deze tool noemt zichzelf *The smartest JavaScript IDE*. WebStorm zoekt het naast bekende zaken zoals kleurcodering, automatisch inspringen en formatteren van code in slimme toetsencombinaties om razendsnel complete codeblokken te typen en sjablonen voor JavaScript-uitbreidingen zoals node.js, CoffeeScript en unit testing. Een indrukwekkende demo (inclusief de functie Live Editing in Google Chrome) is beschikbaar op YouTube, **youtu.be/wCVwdvufTds**.

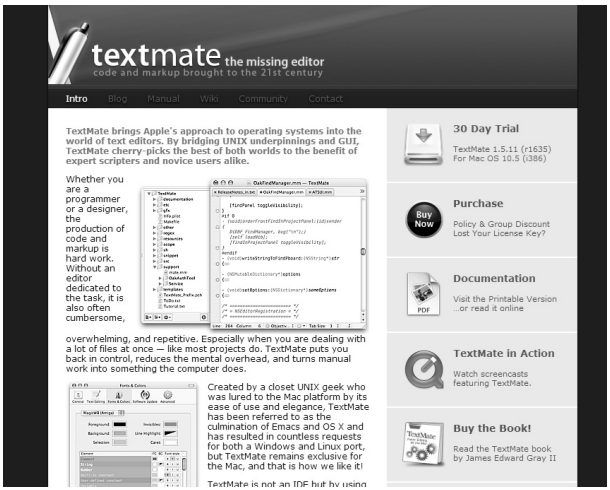


Afbeelding 1.6 *JetBrains WebStorm is een editor die zich heeft gespecialiseerd in JavaScript-toepassingen. De moeite van het beoordelen waard.*

WebStorm is niet gratis, maar voor educatie of persoonlijk gebruik ook niet duur. De registratieprijs was op het moment van schrijven van dit boek ongeveer vijftig euro. Er is een dertigdagenversie verkrijgbaar.

TextMate

Als u een goede editor zoekt voor gebruik op de Mac, moet u eens kijken naar TextMate. Dit is een complete toolbox voor webontwikkeling met goede ondersteuning voor zowel HTML, CSS als JavaScript. Er is veel documentatie voor beschikbaar en een goede ondersteuning vanuit de community. TextMate kent een plug-insysteem en kan daarom worden uitgebreid met tal van handige extra tools. Textmate is verkrijgbaar via **www.macromates.com**.



Afbeelding 1.7 *Mac-developers moeten zeker TextMate eens bekijken. Het is een goede editor met tal van scriptingmogelijkheden.*

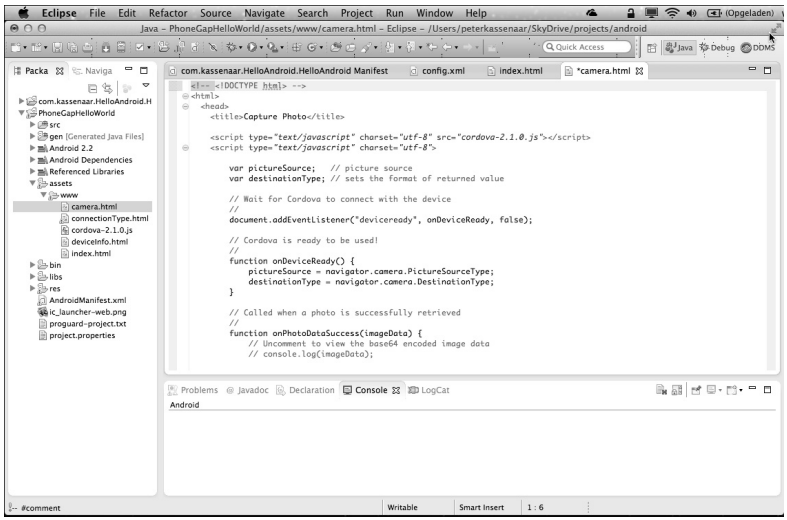
Overige tools

Zoals gezegd kunt u met elke editor die een document als platte ASCII-tekst kan opslaan uit de voeten. Kan geen van de hiervoor genoemde editors u bekoren, dan kunt u het eens proberen met een van de volgende tools. Als die ook niet bevallen, dan weten wij het ook niet meer. Veel plezier met Kladblok in dat geval.

- **Notepad++ – notepad-plus-plus.org**. Een opensource-editor met vele plug-ins en ondersteuning voor allerlei programmeertalen. Ook in het Nederlands.
- **Eclipse – www.eclipse.org**. Een toepassing die vooral onder Java- en Android-programmeurs bekend is. Met de plug-in *Eclipse Web Developer*

Tools maakt u Eclipse ook geschikt voor het programmeren van webprojecten en worden JavaScript-bibliotheken en -syntaxis ondersteund.

- **CoffeeCup – www.coffeecup.com**. Een uitgebreide editor die zowel voor Windows als voor Mac beschikbaar is. Er worden veel voorbeeldscripts meegeleverd, zodat u kunt leren van de voorbeelden van anderen. Er is een probeerversie beschikbaar.

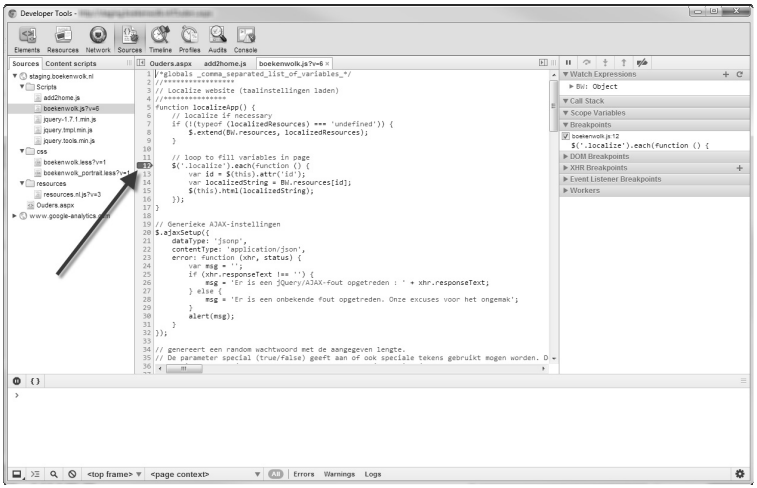


Afbeelding 1.8 *Eclipse is in principe een toepassing voor complete Java- of Android-apps, maar kan na installatie van een plug-in ook worden gebruikt voor webdevelopment.*

JavaScript-debuggers

Een programmeeromgeving is niet compleet zonder debugger. In een debugger kan de programmeur de uitvoering van het script volgen en kunnen desgewenst breekpunten worden geplaatst. Als de code is aangekomen op de plek van het breekpunt, wordt de uitvoering gestopt. U kunt dan de waarde van variabelen inspecteren, stapsgewijs door de code lopen en in sommige gevallen de uitvoering van de code verleggen. Debuggers worden buitengewoon veel gebruikt. Met Chrome, Safari en Internet Explorer zijn debuggers meegeleverd, voor Firefox kunt u de populaire add-on Firebug gebruiken.

De debuggers worden op de volgende wijze geactiveerd in de browser. In de rest van het boek gaat u er nog uitgebreid mee aan slag, dus al te lang staan we er nu niet bij stil.



Afbeelding 1.9 *De debugger in Developer Tools van Google Chrome. Op regel 12 van de code is een breekpunt ingesteld. Als de code-uitvoering daar is aangekomen, wordt het programma onderbroken en kunt u in de panelen aan de rechterkant of in de console (onderaan) de uitvoering van het script volgen.*

- **Google Chrome** Druk op Ctrl+Shift+I (Cmd+Shift+I op de Mac) om het venster Developer Tools te openen. U kunt ook met de rechtermuisknop klikken en kiezen voor **Element inspecteren**.
- **Apple Safari** Druk op Ctrl+Alt+I (Cmd+Option+I op de Mac) om de Web Inspector te openen. Dit venster lijkt erg op Developer Tools van Chrome, omdat ook Safari is gebaseerd op de WebKit-browserengine. In het snelmenu van de rechtermuisknop is de optie **Inspecteer element** beschikbaar.
- **Microsoft Internet Explorer** Druk op F12 om het venster met Internet Explorer-ontwikkelhulpmiddelen te openen. Dit ziet er anders uit dan de hulpmiddelen van Chrome en Safari, maar bevat ook tabbladen voor het starten van JavaScript-foutopsporing en meer.
- **Mozilla Firefox** Download en installeer eerst Firebug vanaf **www.getfirebug.com**. Daarna is in de werkbalk een knop met een kevertje (de 'bug') aanwezig. Klik hierop om Firebug voor de huidige pagina te openen.

Onze keuze: Chrome Developer Tools

In dit boek maken we gebruik van Chrome Developer Tools, maar alle handelingen zijn ook uit te voeren met Firebug of de ontwikkelhulpmiddelen van Internet Explorer (F12). Hierin ontwikkelt u na verloop van tijd vanzelf een voorkeur. In ieder geval weet u nu dat het werken met een debugger onontbeerlijk is voor de serieuze JavaScript-programmeur.

Uw eerste JavaScript

Na al deze theorie bent u er waarschijnlijk wel aan toe zelf een werkend script te schrijven! Laten we daarom eens kijken op welke manier JavaScript aan een webpagina wordt toegevoegd. Overal in een HTML-document kunt u JavaScript-code plaatsen. Het is niet gebonden aan een plaatsje binnen de tag `<head>...</head>` of binnen de body van het document. U schrijft eenvoudig de JavaScript-code daar waar u er behoefte aan hebt. Voor de organisatie van uw scripts en het onderhoud van de site is het uiteraard wel handig de code te groeperen of hiervoor een vaste volgorde aan te houden, maar verplicht is het niet. Bovendien mag u zoveel code schrijven als u maar wilt. Zolang u de JavaScript-code maar plaatst binnen de tag `<script>...</script>`. Gebruikt u JavaScript, dan komt dus ergens in het webdocument het volgende blok voor:

```
<script>
    // Alle JavaScript code komt hier
</script>
```

Type aangeven?

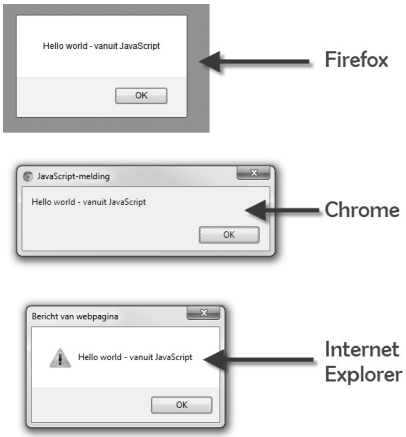
Vroeger was het gebruikelijk in de tag `<script>` aan te geven welke type scripttaal u gebruikt. Dit deed u door het attribuut `type="text/javascript"` toe te voegen. In HTML5-documenten hoeft dit niet meer. JavaScript is het standaardscripttype, dus u bespaart een paar bytes aan bandbreedte en typewerk door eenvoudig de tag `<script>` te gebruiken. In dit boek wordt het attribuut `type` niet meer gebruikt.

Commentaar gebruiken

Binnen JavaScript gebruikt u de tekens `//` voor commentaar tot het eind van de regel of `/*.....*/` voor een commentaarblok dat zich over meer regels uitstrekt. Beide notatiewijzen van commentaar kent u misschien uit Java of PHP.

Een van de makkelijkste manieren om de werking van JavaScript te demonstreren is door een scriptje te schrijven dat een venstertje met een boodschap (*alert*) op het scherm toont, zoals in de afbeelding is te zien. Typ daartoe de volgende code in een gewoon HTML-document:

```
<script>
    // Toon een venster met een korte boodschap op het scherm
    alert ("Hello world - vanuit JavaScript");
</script>
```

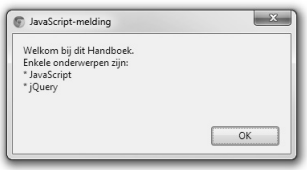


Afbeelding 1.10 *De browser voert het JavaScript uit; merk op dat het alertvenster er in de diverse browsers verschillende uitziet; dit is door de programmeur niet te beïnvloeden.*

Vergelijkt u de vensters uit de afbeelding met elkaar, dan ziet u dat de verschillende browsers ook verschillende manieren hebben om het JavaScript weer te geven. Hier kunt u als programmeur niets aan veranderen. Andere browsers hebben ook een andere titel of ander uiterlijk voor het JavaScript-venster.

Binnen een JavaScript-dialoogvenster werken HTML-tags als `<br />` en `<p>` niet. Als u binnen een venster tekst over meer regels wilt verdelen moet u de escapecode `\n` (van *newline*) gebruiken. Er zijn nog meer escapecodes, zoals `\t` voor een tab, `\r` voor een backspace en `\\` voor het teken '\' zelf. Ook deze escapecodes komen uit de programmeertaal C. Beschouw het volgende voorbeeld:

```
<script>
   // Een venster met meerdere regels.
   alert ('Welkom bij dit Handboek.\n' +
       'Enkele onderwerpen zijn:\n' +
       '\t* JavaScript\n' +
       '\t* jQuery\n');
</script>
```



Afbeelding 1.11 *Meer regels in een alertvenster.*

Enkele dingen vallen op:

- Alle tekst die in het venster komt te staan, staat binnen het hakenpaar `alert (...)`.
- Tekst kunt u in de editor over meer regels verspreiden, zolang u elke afzonderlijke regel tekst maar tussen enkele aanhalingstekens (`'...'`) zet en met het plusteken (+) met elkaar verbindt.
- Het afsluitende teken **;** (de puntkomma) plaatst u pas nadat u het haakje gesloten hebt, dus aan het eind van een logische regel; niet per se aan het eind van een fysieke regel.

Dialoogvenster in plaats van alert

De bibliotheek jQuery UI bevat de component `.dialog()`. Deze zou u prima kunnen gebruiken als vervanging van het (lelijke) JavaScript-alertvenster. Dan weet u zeker dat het venster er in alle browsers hetzelfde uitziet en kunt u bovendien zelf de opmaak bepalen. Zie hoofdstuk 14 voor meer informatie over het werken met jQuery UI.

JavaScript-functies

De JavaScript-code `alert()` is een ingebouwde functie van JavaScript. U hoeft hem niet apart te definiëren. De browser zorgt ervoor dat het venster op het scherm wordt gezet, dat er een knop **OK** in staat enzovoort. Zoals u verderop zult zien zijn er ook door de gebruiker gedefinieerde functies. Dit zijn functies die u zelf schrijft.

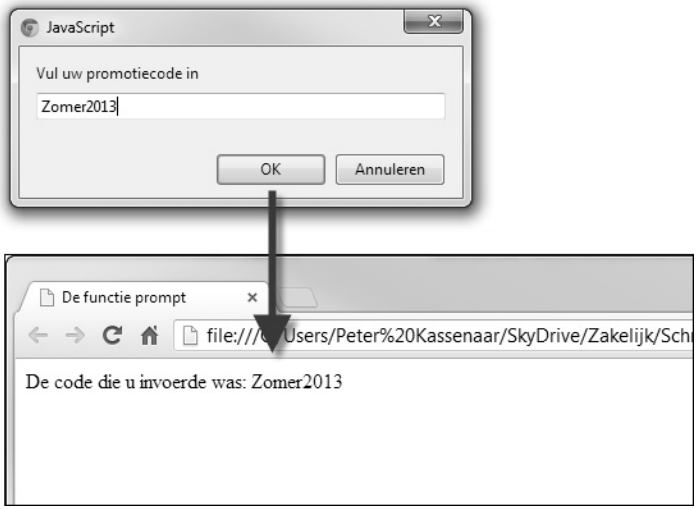
Parameters

Een andere voorgedefinieerde functie is de functie `prompt()`. Met deze functie kunt u de gebruiker vragen iets in te vullen in een venster. De functie heeft twee parameters. Dit zijn de tekst die in het venster verschijnt en een eventuele standaardtekst die in het tekstvak getoond wordt. Als we het over functies hebben, spreken we over *parameters* van een functie. Ze worden binnen het hakenpaar getoond. U kunt dit vergelijken met de attributen van tags. U gebruikt parameters om speciale argumenten of extra kenmerken op te geven.

Het volgende codefragment is iets uitgebreider. We laten de complete pagina zien (dus inclusief de HTML-code).

```
<body>
<div id="divResult"></div>
<script>
```

```
    // twee variabelen
    var code = prompt('Vul uw promotiecode in', 'uw code');
    var tekst = 'De code die u invoerde was: ' + code ;
    // resultaat in de pagina plaatsen
    document.getElementById('divResult').innerHTML = tekst;
</script>
</body>
```



Afbeelding 1.12 *Gebruikersinvoer wordt verwerkt in de pagina.*

De uitvoer is zoals in de afbeeldingen is te zien.

Analyse

- Bij het openen van de pagina is nog niets te zien. Dat komt doordat de `<div id="divResult">` een lege div is.
- Met de functie `prompt()` wordt een 'code' gevraagd. De door de bezoeker ingevulde waarde wordt toegekend aan een variabele die we de naam `code` geven. Een variabele wordt gemaakt met het JavaScript-keyword `var`.
- We maken een tweede variabele met de naam `tekst`. Hieraan wordt een standaardberichttekst toegekend en we voegen er ook de ingevulde code aan toe.
- Daarna selecteren we een element op de pagina. Dit doen we met het statement `document.getElementById('divResult')`. Dat betekent: selecteer op de pagina het element met de id `divResult`. Dit is de lege div die we in HTML hebben gedefinieerd.

- Van deze div passen we de eigenschap `innerHTML` aan door de waarde van de variabele tekst er aan toe te kennen. Oftewel: de door ons samengestelde tekst wordt in de pagina geplaatst.

Zodra u het venster sluit met **OK**, zult u zien dat het resultaat uit de afbeelding wordt bereikt. Test zelf: wat gebeurt er als u in het promptvenster **Annuleren** of **Cancel** (dit hangt af van de browser) kiest?

Een inline event handler schrijven

We passen het script nu zodanig aan dat het promptvenster pas wordt geopend op het moment dat de bezoeker op een knop klikt. Ook de uitvoer van het script wordt op die manier gebruikersafhankelijk. Het wordt niet meer direct uitgevoerd zodra de pagina wordt geopend. Hiervoor passen we verschillende dingen aan.

- We voegen een knop toe aan de pagina. In de code van de knop wordt aangegeven dat een JavaScript-functie wordt aangeroepen op het moment dat erop wordt geklikt.
- De code van het JavaScript plaatsen we in een eigen functie. U maakt hiervoor kennis met het keyword `function()`.
- De werking van het script blijft verder gelijk. De invoer van de bezoeker wordt toegevoegd aan een element op de pagina.

De code wordt als volgt:

```
<body>
<button id="btnPrompt" onclick="toonPrompt();">Voer code in</button>
<div id="divResult"></div>

<script>
// functie
function toonPrompt(){
    var code = prompt('Vul uw promotiecode in', 'uw code');
    var tekst = 'De code die u invoerde was: ' + code ;
    document.getElementById('divResult').innerHTML = tekst;
}
</script>
</body>
```

In de afbeelding is een voorbeeld van de uitvoer te zien.

Afbeelding 1.13 *Het script is in een eigen functie geplaatst. De functie wordt aangeroepen zodra op de knop wordt geklikt.*

Analyse

- Nieuw in het HTML-deel van de code is de knop. Deze is aangegeven met `<button>…</button>`.
- In het `<script>`-deel is de code nu omgeven door een functiedefinitie. We hebben als functienaam `toonPrompt` gekozen. De naam van een functie mag u zelf verzinnen.
- Achter de functienaam staat een hakenpaar `()`. De functie heeft geen parameters, daarom zijn de haken leeg. De inhoud van de functie staat tussen accolades `{…}`. Dit is verplicht. In hoofdstuk 4 leest u meer over de opbouw van functies.
- Een kenmerk van functies is dat de code die er in staat pas wordt uitgevoerd op het moment dat deze wordt aangeroepen. Dat gebeurt hier via de event handler `onclick="…"` in de definitie van de knop.

Liever geen inline event handlers

In het voorbeeld is te zien dat rechtstreeks in de HTML-code de event handler `onclick="toonPrompt();"` wordt geschreven. Dit is de eenvoudigste manier om een event handler te maken. Daarom laten we hem hier als eerste zien. Maar het is niet meer de aanbevolen manier. In programmeertermen zeggen we dat het geen *best practice* is om de event handler rechtstreeks te schrijven binnen het element (een andere naam hiervoor is *anti-pattern*). U mixt op deze manier als het ware HTML en JavaScript. Dat is niet netjes. Het is beter om in het scriptgedeelte een eventListener te definiëren en hierin de code van de functie te koppelen aan de `id` van de knop. Hierover leest u meer in hoofdstuk 6.

Het codevoorbeeld is beschikbaar als script_0104.html in het zipbestand met codevoorbeelden dat bij dit boek hoort (zie **www.kassenaar.com/hbjs**).