Inhoud

1 Hoofdstuk

Kennismaken met JavaScript

HTML is al bijna dertig jaar de standaard voor het maken van websites. HTML kan echter niet alles. In HTML wordt alleen de structuur van pagina's beschreven. JavaScript is de aanvullende programmeertaal om HTML interactief te maken. Het is de populairste programmeertaal op internet. Elke browser heeft een ingebouwde JavaScript-motor, waardoor moderne webapps mogelijk worden. JavaScript staat daarmee aan de basis van elke techniek die de moderne webdeveloper moet kennen. Of u later nu aan de slag gaat met Angular, webapps gaat maken met Vue of React, of uw eigen bibliotheekje met helperfuncties maakt: zonder JavaScript bent u nergens. Dit inleidende hoofdstuk toont de algemene kenmerken van JavaScript en laat zien 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.

In dit hoofdstuk:

Een korte geschiedenis van JavaScript.

Waarvoor wordt JavaScript gebruikt?

Belangrijke begrippen die u moet kennen bij het werken met JavaScript.

Welke tools hebt u nodig 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

Brendan Eich

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. 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).

Afbeelding 1.1 *Brendan Eich is de maker van de oorspronkelijke versie van JavaScript.*

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. JavaScript is uitgegroeid tot een van de belangrijkste pijlers binnen de browser en in moderne webapplicaties ('webapps'). Zonder JavaScript zouden geen Facebook, Instagram, Gmail, e-commerce of internetbankieren bestaan. Alle browsers (Microsoft Edge, Mozilla Firefox, Google Chrome, Apple Safari enzovoort) kunnen JavaScript uitvoeren.

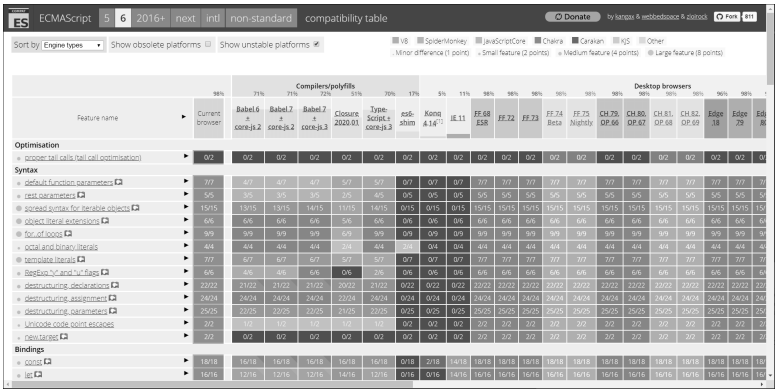
ECMAScript, JavaScript en versienummers

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. Nu is de ontwikkeling van JavaScript losgekoppeld van nieuwe versies van de browser. 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.

Vroeger spraken we dus van JavaScript 1.0, 1.5 enzovoort. Tegenwoordig wordt gesproken van ECMAScript 5 (2009), of ECMAScript 6 uit 2015 (vaak afgekort tot ES6).

In juni van elk jaar vind een congres plaats waar wordt besloten welke nieuwe onderdelen worden verheven tot de JavaScript-standaard. We kennen daarom ES2017, ES2018 enzovoort. De verschillen tussen deze versies zijn voor gevorderde programmeurs belangrijk, maar niet als u juist kennismaakt met JavaScript. Alle code in dit boek is geschikt voor moderne versies van JavaScript.



Afbeelding 1.2 *Bekijk eventueel de ECMAScript compatibility table op kangax.github.io/compat-table/es6/ als u precies wilt weten welke onderdelen tot welke versie van JavaScript behoren en hoe de browserondersteuning is. Dit is vooral leuk voor nerds.*

Wat doet ECMA?

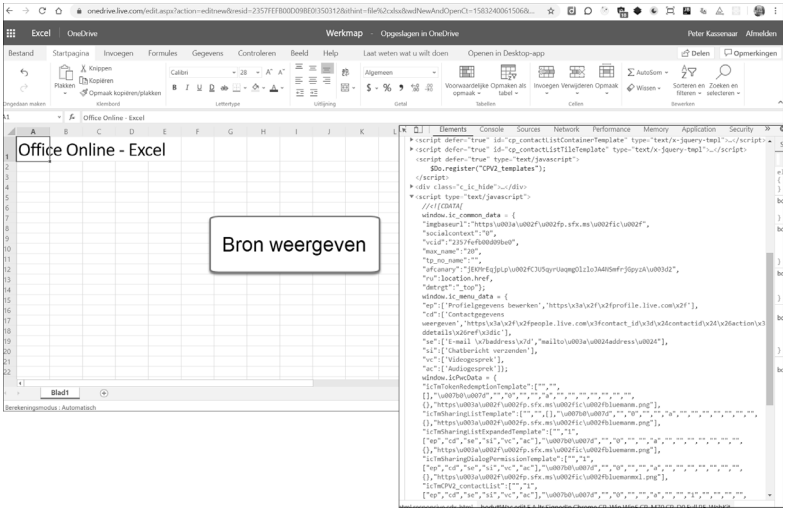
ECMA heeft tot taak technische standaarden te publiceren en hierop toe te zien. ECMA houdt zich niet alleen bezig met JavaScript, maar ook met andere computergerelateerde standaarden, zoals het bestandsformaat voor cd-roms, de specificaties van de programmeertaal C# en het bestandsformaat Office Open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

Eigenlijk moeten we dus spreken van ECMAScript. Maar voor het gemak heeft iedereen het altijd gewoon over *JavaScript*. Dat doen we ook in dit boek.

Waarvoor wordt JavaScript gebruikt?

We hebben al globaal aangegeven dat JavaScript wordt gebruikt om gedrag of interactie aan webpagina's toe te voegen. Maar wat wordt daar dan precies mee bedoeld? Denk bijvoorbeeld aan de volgende toepassingen. Er zijn er nog veel meer, maar dit is alvast een begin:

- **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. Door het gebruik van JavaScript is dus geen *roundtrip* nodig naar de server en kan via een lokale melding direct worden aangegeven dat iemand bijvoorbeeld een ongeldig e-mailadres heeft ingevuld.
- **Single Page Applications** Moderne frameworks zoals React, Vue en Angular gebruiken het principe van Single Page Applications om webapps te maken. Dit betekent dat in de browser in principe maar één pagina wordt geladen en dat delen van de pagina door JavaScript ververst worden, zonder dat daarvoor een complete *refresh* van de pagina nodig is. Zonder JavaScript zouden dergelijke frameworks niet bestaan.
- **Uitrolmenu'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** JavaScript kan in een pagina de aanwezigheid, positie en inhoud van elk element (teksten, afbeeldingen enzovoort) ophalen en manipuleren. 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. *jQuery* is hiervan een van de bekendste. We bespreken jQuery uiteraard verderop in dit boek.
- **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. Tegenwoordig wordt hiervoor JSON gebruikt in plaats van XML. Het hele idee van applicaties op het web zoals Microsoft 365 (voorheen Office 365), Facebook, Gmail en Twitter is gebaseerd op gegevensuitwisseling op de achtergrond (*asynchroon*) met JavaScript en JSON. Zonder JavaScript zou het web in zijn huidige vorm niet bestaan!



Afbeelding 1.3 *Office Online (Microsoft 365) is geheel geschreven in JavaScript. Het wordt ook wel 'de grootste JavaScript-applicatie ter wereld' genoemd.*

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. Verder is er in vergelijking met andere programmeertalen echter niet zo veel bijzonders aan. JavaScript bevat bijvoorbeeld geen opdrachten voor invoer en uitvoer via het toetsenbord, er zijn geen netwerkmogelijkheden of mogelijkheden voor het werken met lokale bestanden. De browser is een zogeheten 'sandbox'.

In talen als Java, PHP of C# zijn dergelijke zaken wel opgenomen. In JavaScript worden dit soort uitgebreide handelingen overgelaten aan de zogenoemde *hosting environment*. En op internet is de webbrowser die host waarin JavaScript draait. JavaScript maakt bijvoorbeeld gebruik van het browservenster

Afbeelding 1.4 *JavaScript wordt uitgevoerd in een hostingomgeving. Meestal is dit de webbrowser, maar het kan ook NodeJS zijn (JavaScript buiten de browser) dat voor uitvoering van het script zorgt.*

om teksten op de pagina te tonen en te manipuleren. 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 7 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 de kern van elke programmeertaal. JavaScript is hierop geen uitzondering. Als u deze onderdelen goed beheerst, kunt u JavaScript in tal van omgevingen toepassen. U leert ook beknopt werken met het DOM (de webpagina).
- **Deel 2 – Werken met jQuery** In de resterende hoofdstukken 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). 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. U vindt ze op **www.kassenaar.com/hbjs3** of **www.vanduurenmedia.nl/downloads**. 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 experimenten. 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.

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 weinig voorkennis nodig. JavaScript staat als leesbare platte tekst in de broncode van het webdocument of in een apart (gekoppeld) scriptbestand.
- 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 of eenvoudig Java hebt u al een voorsprong. 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 misschien wat jQuery-code gekopieerd en geplakt, dan leert u nu eindelijk wat al die haakjes, puntkomma's en accolades betekenen.
- Hebt u nog geen programmeerervaring, dan is dat geen enkel probleem. Begin gewoon te oefenen in het volgende hoofdstuk, dan hebt u aan het einde van het boek JavaScript goed onder de knie.

Minimaliseren en bundelen

In veel moderne websites en frameworks worden extra tools zoals *Webpack* of *Babel* gebruikt om JavaScript-code te minimaliseren en te bundelen in één groot bestand. Computers kunnen daar prima mee overweg, maar de broncode is dan onleesbaar voor mensenogen. U kunt er helaas vrijwel niets van leren. In dit boek doen we dat niet. Alle broncode en voorbeelden zijn leesbare codebestanden die u zelf kunt aanpassen en uitbreiden.

Bekendheid met HTML en CSS

We gaan ervan uit dat u bekend bent met HTML. 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.

Kent u dit nog niet, lees dan eerst een ander boek, waarin de basisbeginselen van HTML uiteengezet worden, bijvoorbeeld *Web Development Library HTML5* (ISBN 978-90-5940-808-1) en *Web Development Library - CSS3* (ISBN 978-90-5940-809-8). In dit boek staan we niet verder stil bij de HTML- en CSS-syntaxis van elementen. Zij worden bekend verondersteld.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Titel van het document</title>
    <style>
        /*Stijlen van het document*/
    </style>
</head>
<body>
<div class="container">
<!--Inhoud van de HTML-pagina-->
</div>

<script>
    // Script in het document
</script>
</body>
</html>
```

Afbeelding 1.5 *Bekendheid met de notatie van HTML en CSS is een vereiste. We gebruiken in dit boek het HTML5-documenttype.*

Wat hoeft u niet te weten?

U hoeft niet iets te weten van serversided programmeertalen zoals PHP, Java of C#. Ook hoeft u geen beschikking te hebben over een webserver of eigen domein. In dit boek gaan we uit van clientsided JavaScript. Dit 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 voldoende.

Ontwikkelhulpmiddelen voor JavaScript

JavaScript is gewoon platte tekst. In principe kunt u daarom met Kladblok (Windows) of Teksteditor (Mac) al aan de slag. 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.

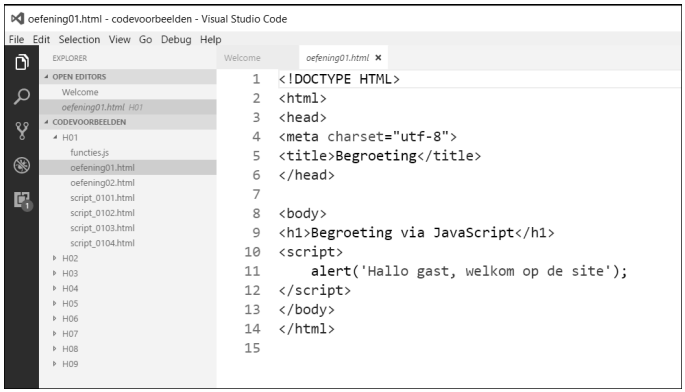
Naast het schrijven van JavaScript is het opsporen en verhelpen van fouten in een script erg belangrijk. Hiervoor is de debugger in de browser beschikbaar. Deze komt aan het eind van de paragraaf aan de orde.

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.

- **Visual Studio** Code Microsoft heeft sinds enige jaren een uitstekende, gratis, open source editor beschikbaar. Deze heet Visual Studio Code (niet te verwarren met het grote, betaalde Visual Studio) en is beschikbaar voor Mac, Windows en Linux. Uw exemplaar is te downloaden vanaf **code.visualstudio.com**.
- **JetBrains** Webstorm WebStorm is een editor die zich profileert als 'speciaal gemaakt voor JavaScript'. Er zijn uitgebreide voorzieningen aanwezig voor het profileren van documenten, herschrijven van code, testen en het automatisch springen naar functies. Zie **www.jetbrains.com/webstorm/** voor meer informatie en een 30-dagenversie.
- **Sublime Text** Een andere bekende open source editor is Sublime Text, te vinden op **www.sublimetext.com**. Deze editor is van zichzelf een tamelijk 'kaal' product, maar kan exact naar wens worden ingesteld met tientallen plug-ins voor kleurcodering, automatisch formateren, FTP, codehints en testingtools.

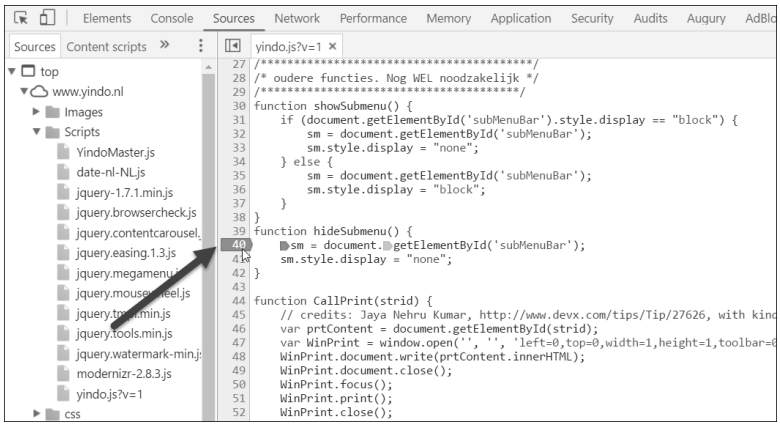
Maar zoals gezegd kunt u met elke editor die een document als platte ASCII-tekst kan opslaan uit de voeten. Kan geen van de hiervoor genoemde editors u bekoren, dan kunt u het proberen met Atom, Sublime Text, Brackets, Notepad++ of een andere editor.



Afbeelding 1.6 *Visual Studio Code is een goede, gratis editor. Erg geschikt voor programmeren in JavaScript.*

JavaScript-debuggers

Een programmeeromgeving is niet compleet zonder debugger. Met een debugger zijn fouten of onverwacht gedrag in een programma op te sporen en – hopelijk – te verhelpen. Hiervoor plaatst de programmeur een *breekpunt* in de code, op de plek waar hij de fout verwacht. 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 meer. Debuggers worden buitengewoon veel gebruikt. Alle moderne browsers hebben een debugger (druk op F12).



Afbeelding 1.7 *De debugger in Developer Tools van Google Chrome. Op regel 40 van de code is een breekpunt ingesteld. Als de code-uitvoering daar is aangekomen, wordt het programma onderbroken. In de deelvensters van de Developer Tools is de uitvoering vervolgens te sturen.*

Onze keuze: Chrome Developer Tools

In dit boek gebruiken we de Chrome Developer Tools. Maar alle handelingen zijn ook uit te voeren met Firefox of de ontwikkelhulpmiddelen van Microsoft Edge (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.

Een andere manier om de debugger te openen is door te klikken met de rechtermuisknop en **Element inspecteren** (*Inspect Element*) te kiezen, of iets vergelijkbaars. Deze opdracht heet in elke browser anders.

Afbeelding 1.8 *De opdracht Element inspecteren in de browser en vervolgens het tabblad Sources opent ook de debugger.*

Uw eerste JavaScript

Na al deze theorie bent u er waarschijnlijk wel aan toe zelf een werkend script te schrijven! Overal in een HTML-document kunt u JavaScript-code plaatsen. Het is niet verplicht dit binnen `<head>...</head>` te doen. Voor de organisatie 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.

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

Type aangeven?

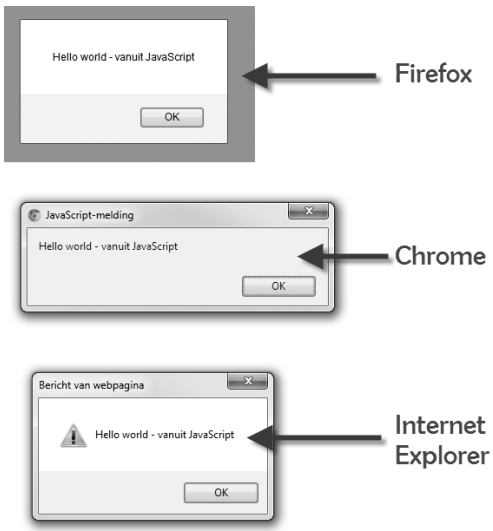
Vroeger was het gebruikelijk in de tag `<script>` aan te geven welke type scripttaal u gebruikt. Dit deed u door het attribuut `type="text/javascript"` toe te voegen. In HTML5-documenten hoeft dit niet meer. JavaScript is het standaardscripttype. In dit boek wordt het attribuut `type` niet meer gebruikt.

Commentaar gebruiken

Binnen JavaScript gebruikt u de tekens `//` voor commentaar tot het eind van de regel of `/*.....*/` voor een commentaarblok dat zich over meerdere regels uitstrekt. Commentaar wordt vaak gebruikt voor toelichting in een script. Het wordt niet uitgevoerd.

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.

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



Afbeelding 1.9 *De browser voert het JavaScript uit; merk op dat het alert-venster er in de diverse browsers verschillende uitziet. Dit kunt u als programmeur niet instellen.*

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.

Enkele dingen vallen op:

- Alle tekst die in het venster komt te staan, staat binnen het hakenpaar `alert (...)`.
- Het teken `;` (de puntkomma) staat aan het einde van het statement. Dit betekent voor JavaScript dat de opdracht is afgesloten. Elk statement wordt afgesloten met een `;`.

Dialoogvenster in plaats van alert

Met aanvullende bibliotheken als jQuery kunt u eigen dialoogvensters, pop-ups of overlays maken. 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.

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 en prompt()

Een andere standaardfunctie is de functie `prompt()`. Met deze functie kunt u de gebruiker vragen iets in te vullen in een venster. De functie heeft twee *parameters*. Parameters zijn de argumenten die tussen de haakjes van een functieaanroep staan. Een functie kan nul, één of meerdere parameters hebben.

In het geval van `prompt()` zijn de parameters de tekst die in het venster verschijnt en een eventuele standaardtekst die in het tekstvak getoond wordt. U kunt parameters vergelijken met de attributen van HTML-tags. Ze worden gebruikt 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). Bij de downloads voor dit boek is dit script te vinden als script_0102.html.

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

De uitvoer is zoals in de afbeeldingen is te zien.

Afbeelding 1.10 *Gebruikersinvoer wordt verwerkt in de pagina.*

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 `const`.
  - Er zijn verschillende soorten variabelen in JavaScript. Ze worden aangegeven met `var` (verouderd), `let` en `const`. Het keyword `const` betekent *constante*. De waarde van de variabele wordt toegekend, maar gedurende het script verder niet meer gewijzigd. Dit is de aanbevolen manier voor variabelen. Ook `var` en `let` worden in de volgende hoofdstukken nog besproken.

- We maken een tweede variabele met de naam `tekst`. Hieraan wordt een standaardberichttekst toegekend. We voegen hem samen met de eerder toegekende `code`.
- 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` eraan 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. Het keyword `function()` is hiervoor beschikbaar.
- De werking van het script blijft verder gelijk. De invoer van de bezoeker wordt in een element op de pagina geplaatst.

De code wordt als volgt (script_0103.html):

```
<body>
<button id="btnPrompt" onclick="toonPrompt();">Voer code in</button>
<div id="divResult"></div>
<script>
// functie
function toonPrompt(){
    const code = prompt('Vul uw promotiecode in', 'uw code');
    const 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 1.11 *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 5 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.

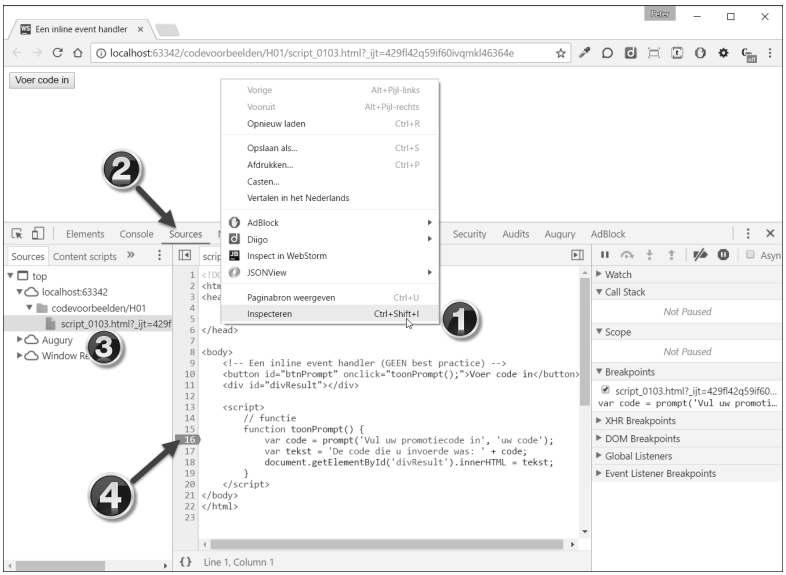
Liever geen inline event handlers

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 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.

De debugger gebruiken

Dit is een mooi moment om ook in de praktijk te zien hoe de debugger wordt ingezet tijdens het werken met JavaScript. In dit voorbeeld (en de rest van het boek) gebruiken we Google Chrome. We adviseren u om de nieuwste versie van Chrome te installeren via **google.nl/chrome**. De werkwijze is als volgt.

1. Zorg ervoor dat de pagina geopend is in Chrome. U ziet de kale pagina met een knop.
2. Klik met de rechtermuisknop en kies onderin **Element inspecteren**. In de Engelse versie van Chrome heet deze opdracht **Inspect Element**. In de rest van het boek spreken we kortweg over het 'openen van de Developer Tools'.
3. Selecteer de tab **Sources**.
4. Als nog geen broncode zichtbaar wordt, controleer dan of in de lijst aan de linkerkant het bronbestand (script_0103.html) geselecteerd is.
5. Plaats een breekpunt door op regel 16 in de marge te klikken. Er wordt een kleine blauwe pijl zichtbaar en het breekpunt verschijnt in het paneel Breakpoints aan de rechterkant.



Afbeelding 1.12 *Open de Developer Tools met de opdracht Element inspecteren. Daarna kunt u in het tabblad Sources het juiste bestand selecteren en breekpunten plaatsen op de gewenste locatie(s).*

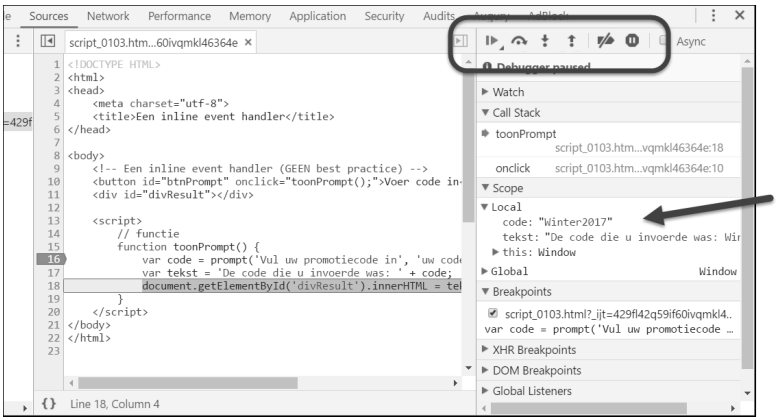
De pagina draait nu in de browser, er gebeurt niets. Pas op het moment dat u op de knop klikt, komt de browser in actie en wordt de event handler `onclick` uitgevoerd. Ga op de volgende wijze verder.

6. Klik op de knop. Het venster wordt nu grijs weergegeven en de tekst `Paused in debugger` verschijnt in een kleine gele tooltip.
7. Onder in het scherm wordt nu regel 16 gearceerd (blauw) weergegeven. Het script is op deze regel aangekomen, omdat via de event handler van de knop de functie `toonPrompt()` wordt aangeroepen.
   - In programmeertermen zeggen we ook wel dat het script 'in het breekpunt valt'.
8. De knoppen met de pijlen rechts boven de deelvensters zijn nu ook actief. Hiermee kunt u:
   - het script vervolgen (de eerste knop);
   - de opdracht in één keer uitvoeren (de tweede knop);
   - stap voor stap de huidige opdracht doorlopen (de derde knop);
   - uit de huidige opdracht/functie stappen (de vierde knop). De werking hiervan zal in de loop van het boek duidelijk worden.
9. Kijk naar de deelvensters aan de rechterkant. In de afbeelding zijn de deelvensters **Call Stack**, **Scope** en **Breakpoints** geopend.
   - Het deelvenster Call Stack bevat de volgorde van functieaanroepen door het script. De onderste vermelding (`onclick`) was de eerste functie die werd uitgevoerd en dat gebeurde op regel 9 van script_0104.html. Daarna werd `toonPrompt` aangeroepen op regel 15 van dezelfde pagina. Vooral bij ingewikkelde scripts, waarbij functies andere functies aanroepen en daarbij parameters doorgeven, werkt het bestuderen van de Call Stack verhelderend om inzicht te krijgen in de structuur van het programma.
   - Het deelvenster Scope Variables bevat de variabelen in dit deel van het script. Op dit moment zijn dat `code` en `tekst` (de variabele `this: Window` mag u nog even negeren). Ze hebben nog geen waarde: `undefined`. Dat klopt, want pas op regel 15 en 16 worden die waarden ingesteld.
   - Het deelvenster Breakpoints laat zien welke breekpunten aanwezig zijn. Dat is op dit moment alleen het breekpunt op regel 15. De scriptuitvoering staat daar te wachten.
10. Klik op de knop **Step over** of druk op F10. Dit is de tweede knop boven de panelen. Het statement op regel 16 wordt dan uitgevoerd.
    - U kunt dit zien aan het promptvenster dat verschijnt.
11. Typ een code in het tekstvak en klik op **OK**. U keert dan terug in de debugger, op regel 17.
12. Klik nog een keer op **Step over**.
    - Het statement op regel 16 wordt uitgevoerd en de cursor springt naar regel 18.

13 Kijk in het paneel **Scope**.
- U ziet dat de waarden van de variabelen wordt getoond. Zo kunt u dus tijdens het uitvoeren van script controleren welke waarden bepaalde variabelen hebben.

14 Klik nogmaals op **Step over**. Op de achtergrond wordt het statement van regel 18 uitgevoerd; de tekst wordt in de pagina geplaatst.

15 Kies nog enkele keren de knop **Step over**, totdat het script helemaal is doorlopen.
- Om helemaal door te gaan kunt u ook op de meest linkse knop klikken (deze heet **Pause Script Execution**) of op F8 drukken. Het script wordt dan niet meer stap voor stap uitgevoerd.



Afbeelding 1.13 *Twee statements zijn stap voor stap uitgevoerd. De inhoud van de variabelen is te zien in het paneel Scope.*

Voer op deze manier de oefening nog enkele keren uit. Probeer ook eens wat er gebeurt als u het breekpunt op een andere positie plaatst of deactiveert. Bekijk de werking van de knoppen boven de panelen. Op deze manier raakt u enigszins bekend met het debuggen van JavaScript-toepassingen. Klik op het kruisje rechtsboven in het deelvenster om de debugger te sluiten.

Tips bij debuggen

1) Breekpunten kunnen alleen op scriptposities worden geplaatst. Het is dus niet mogelijk een breekpunt te plaatsen op regel 9 of 10, want hier staat HTML-code. 2) Met de knop dock/undock linksonder in het venster kunt u de Developer Tools in een eigen venster openen. Ook Firebug en de Internet Explorer-ontwikkelhulpmiddelen hebben hiervoor een knop. Vooral bij grotere scripts kan het handig zijn de vensters met de webpagina en de debugger naast elkaar op het (breed)beeldscherm te plaatsen.