

# Inhoud

<b>1</b>	<b>Kennismaken met JavaScript</b>	<b>1</b>
	<b>Een korte geschiedenis van JavaScript</b>	<b>2</b>
	Brendan Eich	2
	ECMAScript, JavaScript en versienummers	2
	Waarvoor wordt JavaScript gebruikt?	4
	<b>Kernbegrip – JavaScript core</b>	<b>5</b>
	Indeling van dit boek	6
	Oefenbestanden downloaden	6
	<b>Voorkennis</b>	<b>7</b>
	Bekendheid met HTML en CSS	7
	Wat hoeft u niet te weten?	8
	<b>Ontwikkelhulpmiddelen voor JavaScript</b>	<b>8</b>
	<b>JavaScript-debuggers</b>	<b>10</b>
	<b>Uw eerste JavaScript</b>	<b>11</b>
	Commentaar gebruiken	11
	<b>JavaScript-functies</b>	<b>13</b>
	Parameters en prompt()	13
	Een inline event handler schrijven	15
	<b>De debugger gebruiken</b>	<b>17</b>
	<b>JavaScript-code in extern bestand</b>	<b>20</b>
	<b>Conclusie</b>	<b>21</b>
	<b>Praktijkoefeningen</b>	<b>22</b>
<b>2</b>	<b>Statements, gegevenstypen en variabelen</b>	<b>25</b>
	<b>De syntaxis van JavaScript</b>	<b>26</b>
	<b>Statements</b>	<b>26</b>
	Structuur van statements	27
	Hoofdletters en kleine letters	27
	<b>Werken met variabelen</b>	<b>28</b>
	De naamgeving van variabelen	29
	<b>Gereserveerde woorden</b>	<b>30</b>

<b>Commentaar</b>	<b>30</b>
<b>Gegevenstypen</b>	<b>31</b>
Primitieve- of enkelvoudige gegevenstypen	31
Strongly typed en loosely typed	32
Numbers	33
<b>Getallen converteren met parseInt() en parseFloat()</b>	<b>33</b>
Verkorte schrijfwijze: het plusteken	35
Verkorte schrijfwijze: nesting	36
<b>Tekenreeksen of strings</b>	<b>36</b>
Lege string	36
Speciale tekens in strings	37
Escapetekens	37
Een backslash tonen	38
<b>Werken met stringfuncties</b>	<b>38</b>
<b>Booleaanse waarden</b>	<b>39</b>
<b>Objecttypen</b>	<b>40</b>
<b>Conclusie</b>	<b>41</b>
<b>Praktijkoefeningen</b>	<b>41</b>
<b>3 Operatoren</b>	<b>43</b>
<b>Variabelen bewerken met operatoren</b>	<b>44</b>
Toewijzingsoperatoren	44
Verkorte schrijfwijze	45
Nog kortere schrijfwijze: increment en decrement	45
Wiskundige operatoren	46
Stringoperatoren	46
Logische operatoren	47
Vergelijkingsoperatoren	48
De operatoren == en ===	48
Drie isgelijktokens	49
De voorwaardelijke operator ?, :	50
De operator typeof	51
<b>Bewerkingsvolgorde</b>	<b>52</b>
Vorrangsregels of operator precedence	52
Voorbeelden	54
Bij twijfel, gebruik haakjes!	54
<b>Praktijkoefeningen</b>	<b>54</b>

<b>4</b>	<b>Program flow controleren</b>	<b>57</b>
	<b>Inleiding – verschillende typen lussen</b>	<b>58</b>
	<b>If-else</b>	<b>58</b>
	Accolades	59
	else	60
	Veelgemaakte fout: toekenning in plaats van vergelijking	60
	Nogmaals: de vergelijingsoperator	60
	Conclusie	61
	<b>while()</b>	<b>62</b>
	<b>Het statement for()</b>	<b>63</b>
	Drie parameters voor de for-lus	63
	Voorbeeld – de tafel van tien met for()	64
	<b>De statements break, continue en return</b>	<b>65</b>
	<b>Het statement for-in</b>	<b>66</b>
	<b>Conclusie</b>	<b>68</b>
	<b>Praktijkoefeningen</b>	<b>68</b>
<b>5</b>	<b>Beginnen met functies, arrays en objecten</b>	<b>71</b>
	<b>Complex gegevenstype 1 – Functies</b>	<b>72</b>
	Herhaald taken uitvoeren	72
	<b>Structuur van een functie</b>	<b>73</b>
	Moderne notatie	73
	Anonieme functies	73
	Functies aanroepen	74
	Parameters	75
	<b>Parameters doorgeven</b>	<b>76</b>
	Naamgeving van parameters binnen en buiten de functie	76
	Regels voor parameters	77
	<b>Waarden retourneren</b>	<b>78</b>
	Eén waarde retourneren	78
	Returnwaarde toekennen	79
	Meerdere waarden retourneren	79
	<b>Moderne syntaxis, de arrow function</b>	<b>80</b>
	Kenmerken	80
	Eén parameter voor de functie	81
	Meerdere parameters voor de functie	81
	<b>Complex gegevenstype 2 – Arrays</b>	<b>82</b>
	Arrayelementen uitlezen en toevoegen	83
	Arrays in de debugger	83
	Lengte van array	84

<b>Arraymethoden</b>	<b>85</b>
.join()	85
.reverse()	86
.sort()	86
Gevorderd – eigen sorteerfunctie meegeven	87
.push()	88
.pop()	88
Overige arraymethoden	88
.forEach()	89
<b>Gevorderd – Meer arraymethoden</b>	<b>90</b>
.map()	90
.filter()	91
.reduce()	92
Meer arraymethoden	94
<b>Complex gegevenstype 3 – Objecten</b>	<b>94</b>
Eigenschappen, namen en waarden	94
Accolades	95
Complexe objecten	96
this	96
Classes gebruiken	96
<b>Waarden van objecten uitlezen</b>	<b>97</b>
<b>Conclusie</b>	<b>99</b>
<b>Praktijkoefeningen</b>	<b>99</b>
<b>6 JavaScript-events en event handlers</b>	<b>101</b>
<b>Wat zijn events?</b>	<b>102</b>
Procedureel programmeren	102
Eventgeoriënteerd programmeren	102
Naamgeving van events	103
Target	104
Event handlers of callbacks	104
De functie addEventListener()	105
<b>Voorbeelden van events en event handlers</b>	<b>106</b>
Controleren of het document geladen is	106
Muisevents afvangen	108
De parameter e gebruiken in event handlers	110
Eigenschappen van de event e analyseren	112
<b>Klikken op knoppen afvangen</b>	<b>114</b>
Alternatieve notaties voor de event handler	115
<b>Inhoud van een tekstvak ophalen</b>	<b>116</b>
<b>Toetsenbordevents afvangen</b>	<b>118</b>
<b>Conclusie</b>	<b>120</b>
<b>Praktijkoefeningen</b>	<b>121</b>

<b>7</b>	<b>Werken met het DOM</b>	<b>125</b>
	<b>Wat is het DOM?</b>	<b>126</b>
	Begrippen	127
	<b>Elementen in het DOM selecteren</b>	<b>128</b>
	Selecteren via id	129
	Selecteren via type	129
	Selecteren via CSS-klasse	130
	querySelector() en querySelectorAll() – selecteren met CSS-selectors	132
	Voorbeeld – radiobuttons selecteren en uitlezen	133
	<b>Elementen in het DOM toevoegen en verwijderen</b>	<b>135</b>
	Elementen maken met document.createElement()	135
	Textnodes maken	136
	Elementen invoegen in het DOM	136
	.appendChild()	137
	.insertBefore()	138
	.removeChild()	139
	.replaceChild()	140
	<b>Overige DOM-functies</b>	<b>141</b>
	<b>Conclusie</b>	<b>142</b>
	<b>Praktijkoefeningen</b>	<b>143</b>
<b>8</b>	<b>Kennismaken met jQuery</b>	<b>145</b>
	<b>Wat is jQuery?</b>	<b>146</b>
	jQuery – sinds 2006	146
	jQuery in een notendop	148
	Waarom jQuery gebruiken?	148
	<b>Versies van jQuery</b>	<b>149</b>
	<b>Varianten van jQuery</b>	<b>150</b>
	<b>Praktijk – jQuery toevoegen en gebruiken</b>	<b>151</b>
	jQuery insluiten in de pagina	151
	Werken met een Content Delivery Network, CDN	152
	<b>Enkele jQuery-basisvoorbeelden</b>	<b>153</b>
	HTML-code	153
	Selecties maken	153
	Het jQuery-object	154
	Chaining	155
	De jQuery API	156
	<b>Elementen selecteren met jQuery</b>	<b>157</b>
	<b>De functie document.ready()</b>	<b>159</b>
	<b>Enkele korte voorbeelden</b>	<b>161</b>
	Oefeningen	162
	<b>Conclusie</b>	<b>163</b>
	<b>Praktijkoefeningen</b>	<b>163</b>

<b>9</b>	<b>jQuery API: HTML- en CSS-functies</b>	<b>165</b>
	<b>CSS-eigenschappen lezen en schrijven</b>	<b>166</b>
	De functie <code>.css()</code>	166
	Voorbeeld van <code>.css()</code>	166
	<b>Opties meegeven als object</b>	<b>167</b>
	Configuratieobject	168
	<b><code>.addClass()</code> en <code>.removeClass()</code></b>	<b>168</b>
	<b><code>.toggleClass()</code></b>	<b>170</b>
	<b><code>.hasClass()</code></b>	<b>171</b>
	<b>Werken met HTML en attributen</b>	<b>173</b>
	Voorbeeld-HTML	173
	<code>.html()</code>	173
	<code>.text()</code>	174
	<code>.attr()</code>	175
	Object meegeven als parameter	177
	<b>Elementen invoegen en verwijderen uit het DOM</b>	<b>178</b>
	<code>.append()</code> en <code>.prepend()</code>	178
	<code>.before()</code> en <code>.after()</code>	179
	<code>.appendTo()</code> en <code>.prependTo()</code> – Andere manieren van invoegen	179
	<code>.wrap()</code> en <code>.wrapInner()</code> – Elementen omsluiten	180
	<code>.empty()</code> en <code>.remove()</code> – Elementen verwijderen	181
	<b>Formulievelden verwerken met jQuery</b>	<b>182</b>
	<code>.val()</code>	182
	<code>.is()</code>	183
	Keuzerondjes uitlezen	184
	Selectievakjes uitlezen en de functie <code>.each()</code>	185
	<b>Conclusie</b>	<b>187</b>
	<b>Praktijkoefeningen</b>	<b>188</b>
<b>10</b>	<b>jQuery-animatiefuncties</b>	<b>191</b>
	<b>Basisanimatiefuncties</b>	<b>192</b>
	Inleiding	192
	Animatiesnelheid	192
	Standaardcode bij de voorbeelden	193
	<b><code>.hide()</code> en <code>.show()</code> – Eenvoudige animatie</b>	<b>193</b>
	<code>.toggle()</code>	194
	<code>.slideDown()</code> en <code>.slideUp()</code>	195
	<code>.slideToggle()</code>	196
	<b>Elementen infaden en uitfaden</b>	<b>196</b>
	<code>.fadeIn()</code> en <code>.fadeOut()</code>	196
	<code>.fadeToggle()</code>	197
	<code>.fadeTo()</code> – Zelf transparantie instellen	197

<b>Callbackfuncties na animatie</b>	<b>198</b>
Asynchroon	198
Callbackfunctie	199
Arrow functions gebruiken	200
<b>Eigen animaties maken met .animate()</b>	<b>201</b>
Parameters voor .animate()	201
Configuratieobject	202
Callback na animatie	202
Reset? Zelf schrijven!	203
<b>Wat kunnen we animeren en hoe?</b>	<b>204</b>
Relatieve notaties	204
<b>Easing gebruiken</b>	<b>206</b>
Meer easingmogelijkheden	207
Plug-in van easings.net gebruiken	208
<b>Geavanceerde animatiefuncties</b>	<b>208</b>
<b>Globale eigenschappen voor animaties</b>	<b>209</b>
<b>Case: tabbladen maken</b>	<b>210</b>
Tabbladen als user interface	210
Stap 1 – de tabs maken	210
Stap 2 – de inhoud van de tabs maken	211
Stap 3 – de tabs vormgeven	211
Stap 4 – de tabs functionaliteit geven	213
Tabs faden	214
<b>Case: een luxe tooltip</b>	<b>214</b>
Stap 1 – de HTML-code	214
Stap 2 – CSS schrijven voor de tooltip	215
Stap 3 – het script schrijven	215
Stap 4 – de tooltip tonen en verbergen	216
Stap 5 – de muis volgen	216
Stap 6 – de browsertooltip verwijderen	217
<b>Conclusie</b>	<b>219</b>
<b>Praktijkoefeningen</b>	<b>219</b>
<b>11 Eventafhandeling in jQuery</b>	<b>221</b>
<b>Eenvoudige event binding en -afhandeling</b>	<b>222</b>
Eenvoudige events	222
.click()	222
.hover()	224
.focus() en .blur()	226
<b>Betere eventafhandeling met .on()</b>	<b>228</b>
Fragmentatie	228
Live events met .on()	229
Context selector	230

Live events in lijsten	231
.off()	233
<b>Event object</b>	<b>233</b>
Muispositie onderzoeken	234
Het event object inspecteren	234
Conclusie	235
<b>Browser events</b>	<b>236</b>
<b>Formulierevents</b>	<b>236</b>
.focus() en .blur()	236
.select()	236
.change()	237
.submit()	238
<b>Toetsenbordevents</b>	<b>239</b>
<b>Muisevents</b>	<b>240</b>
<b>Conclusie</b>	<b>241</b>
Live event binding	241
<b>Praktijkoefeningen</b>	<b>242</b>
<b>12 jQuery en Ajax</b>	<b>245</b>
<b>Wat is Ajax?</b>	<b>246</b>
Ajax, XML en JSON	246
Ajax in de browser en op de server	247
jQuery en Ajax	248
<b>Ajax – alleen in combinatie met een server</b>	<b>248</b>
Het object XMLHttpRequest	248
Een kleine webserver met serve	249
<b>HTML-documenten laden met .load()</b>	<b>250</b>
Debuggen van netwerkverkeer	251
Toepassingen	252
<b>Uitbreidingen van .load()</b>	<b>253</b>
Aangegeven fragment laden	253
Gegevens meesturen	254
Callbackfunctie uitvoeren	254
<b>JavaScript same origin policy</b>	<b>255</b>
Geen foutmelding bij .load()	256
<b>jQuery Ajax-functies</b>	<b>257</b>
\$.ajax()	258
<b>De functie .ajax()</b>	<b>258</b>
Opbouw van \$.ajax()	258
Success-callback voor \$.ajax()	259
Foutafhandeling	260
Meer parameters voor \$.ajax()	262
Het object jqXHR	263
Enkele veelgebruikte parameters	263



<b>Case – werken met openweathermap.org</b>	<b>264</b>
Stap 1 – wat is openweathermap.org?	264
Stap 2 – de interface	265
Stap 3 – het script beginnen	266
Stap 4 – de Ajax-call schrijven	266
Stap 5 – de eerste versie testen	266
Stap 6 – Gegevens tonen in de UI	268
Stap 7 – gegevens aanpassen en UI uitbreiden	268
Stap 8 – foutcontrole inbouwen	270
Meer API's	271
<b>Standaardinstellingen maken met .ajaxSetup()</b>	<b>272</b>
<b>Ajax-events</b>	<b>273</b>
Toepassingen van Ajax-events	274
<b>Conclusie</b>	<b>274</b>
<b>Praktijkoefeningen</b>	<b>275</b>
<b>13 Werken met jQuery UI</b>	<b>277</b>
<b>Wat is jQuery UI?</b>	<b>278</b>
Andere projecten	278
Onderdelen van jQuery UI	279
Leer de algemene werkwijze	279
Aparte plug-ins	280
<b>Wat zijn plug-ins?</b>	<b>281</b>
jQuery kan niet alles	281
<b>Kenmerken van plug-ins</b>	<b>281</b>
Plug-ins vinden en downloaden	282
Stappenplan	282
Wat hebben plug-ins met jQuery UI te maken?	284
<b>jQuery UI downloaden en gebruiken</b>	<b>284</b>
Downloaden	284
Toevoegen aan de pagina	286
<b>Uw eerste widget – de datepicker gebruiken</b>	<b>287</b>
De datumkiezer lokaliseren	288
De gekozen datum uitlezen	289
<b>De component slider en werken met events</b>	<b>290</b>
Configuratieobject voor widgets	290
Een slider maken	291
De slider configureren	291
Events voor de slider	292
Parameters voor events	293
Andere notatie voor event handlers	293

<b>Werken met tabs</b>	<b>294</b>
Thema's voor tabs	295
Opties voor tabs	295
<b>Interacties maken met drag-and-drop</b>	<b>297</b>
De categorie Interactions	297
Draggable	297
Opties voor draggable	299
Dropzones maken	299
De event drop afhandelen	301
Terugkeren ongedaan maken	302
De positie verbeteren	303
Conclusie	304
<b>Werken met thema's</b>	<b>305</b>
Wat is een thema?	305
ThemeRoller	305
Een kant-en-klaar thema downloaden en gebruiken	306
Downloaden	307
Twee bestanden jquery-ui.css	309
Een eigen thema maken	309
Conclusie	311
<b>Conclusie</b>	<b>312</b>
Web	312
Twitter	313
<b>Praktijkoefeningen</b>	<b>314</b>
<b>Index</b>	<b>317</b>

# Kennismaken met JavaScript

**D**us u wilt websites verrijken met interactie, animaties en intelligentie? Of gewoon begrijpen wat er aan geheimzinnige code ‘achter’ veel websites hangt? Welkom bij JavaScript! HTML is al 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: zonder JavaScript bent u nergens. Dit inleidende hoofdstuk toont de algemene kenmerken van JavaScript en laat zien wat u nodig hebt om met JavaScript aan de slag te 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 automatisch 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 standaardise-

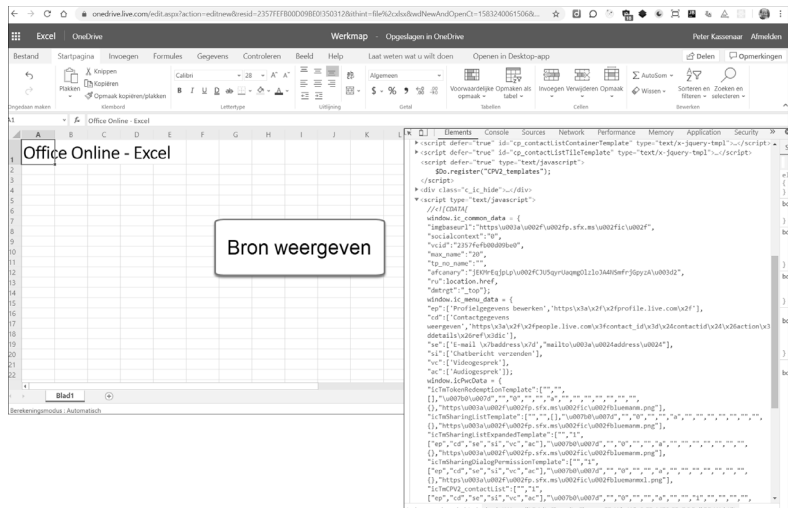


Eigenlijk moeten we dus spreken van ECMAScript. Maar voor het gemak heeft iedereen het altijd gewoon over *JavaScript*. Dat doen we ook in dit boek.

### 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. De Engelstalige term hiervoor die u op internet ook vaak tegenkomt is *client-sided validation*.
- **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 ververs 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 fotocarrousel's 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 ververs 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-toepassing 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 netwerk mogelijkheden 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 om teksten op de pagina te tonen en te manipuleren. Als JavaScript communiceert met een script op de webserver, maakt het gebruik van de browser mogelijkheden om netwerkverbindingen en (Ajax-)aanroepen op te zetten. De *taal* JavaScript zelf biedt hiervoor geen voorzieningen.



**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.

### 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. Ook Node.js is een bekende omgeving om JavaScript op de server uit te voeren (zie [nodejs.org](http://nodejs.org) voor meer informatie). Maar ook acties om PDF-bestanden of Photoshop-filters te automatiseren worden geschreven in JavaScript. In dit boek gebruiken we overigens altijd een webbrowser. Elke programmeur heeft immers wel een computer met een browser. U hoeft er niks extra voor te installeren of te downloaden. Of u gebruikt maakt van Windows, Apple of Linux maakt niet uit.

---

### Oefenbestanden downloaden

Alle codevoorbeelden en -fragmenten die in de tekst worden genoemd zijn als voorbeeldbestanden te downloaden. U vindt ze op [www.kassenaar.com/hbjs4](http://www.kassenaar.com/hbjs4) of [www.vanduurenmedia.nl/downloads](http://www.vanduurenmedia.nl/downloads). Soms gaat het maar om enkele regeltjes code. Dat kan echter 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 gelukkig geen probleem. Begin gewoon te oefenen in het volgende hoofdstuk. Dan hebt u aan het einde van het boek JavaScript goed onder de knie. Maar houd er wel rekening mee dat u moet gaan denken in *code*. JavaScript biedt geen visuele leeromgeving of handige werkbalken zoals Word of Excel.



### 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.



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Titel van het document</title>
6   <style>
7     /*Stijlen van het document*/
8   </style>
9 </head>
10 <body>
11 <div class="container">
12   <!--Inhoud van de HTML-pagina-->
13 </div>
14
15 <script>
16   // Script in het document
17 </script>
18 </body>
19 </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.

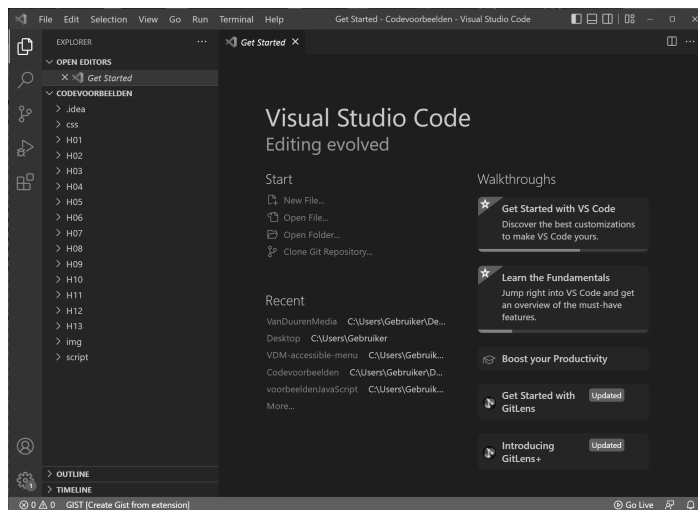
Naast het schrijven van JavaScript is het opsporen en verhelpen van fouten in een script erg belangrijk. Hiervoor is de debugger in de browser beschikbaar. Deze komt aan het eind van de paragraaf aan de orde.



## 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, Figma of Storybook. 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](https://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/](http://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](http://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.

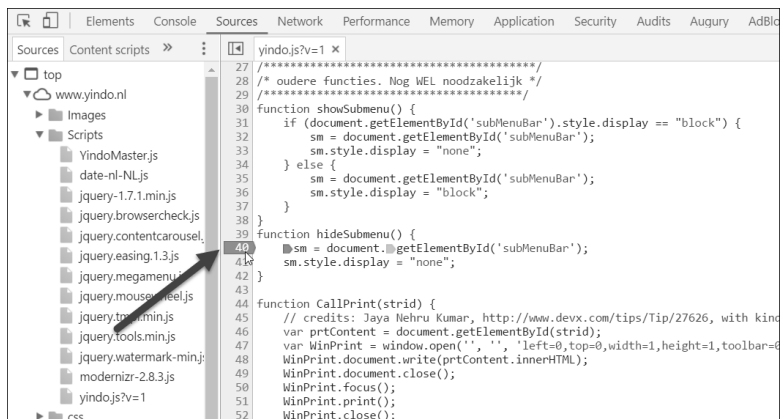


**Afbeelding 1.6** Visual Studio Code is een goede, gratis editor. Erg geschikt voor programmeren in JavaScript.

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.

# 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).



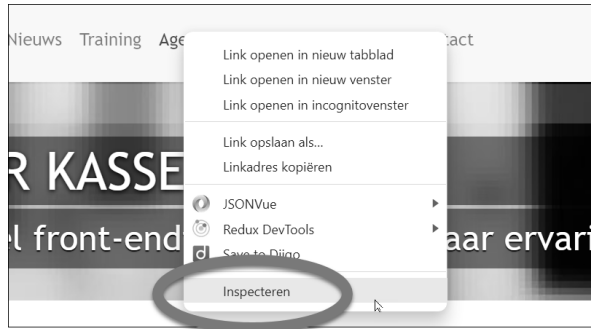
**Afbeelding 1.7** De debugger in Developer Tools van Google Chrome. Op regel 40 van de code is een breekpunt ingesteld. Als de code-uitvoering daar is aangekomen, wordt het programma onderbroken. In de deelvensters van de Developer Tools is de uitvoering vervolgens te sturen.



### 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 vergelijkbaar. Deze opdracht heet in elke browser anders.



**Afbeelding 1.8** De opdracht *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 technisch is dat niet verplicht.

```
<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.

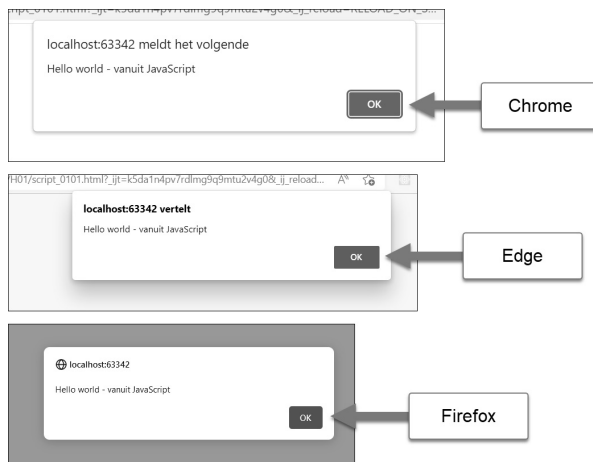
### Commentaar gebruiken

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 pop-up 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 verschillend 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 `;`.



### Dialogvenster in plaats van alert

Met aanvullende bibliotheken als jQuery kunt u eigen dialogvensters, 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 en kleur 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.



### Parameters of argumenten?

In plaats van *parameters* wordt ook wel gesproken van *argumenten* of *opties*. Het is een andere naam, maar betekent hetzelfde. Het zijn de waarden die tussen de haakjes aan een functie worden meegegeven.

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 'promotiecode' 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

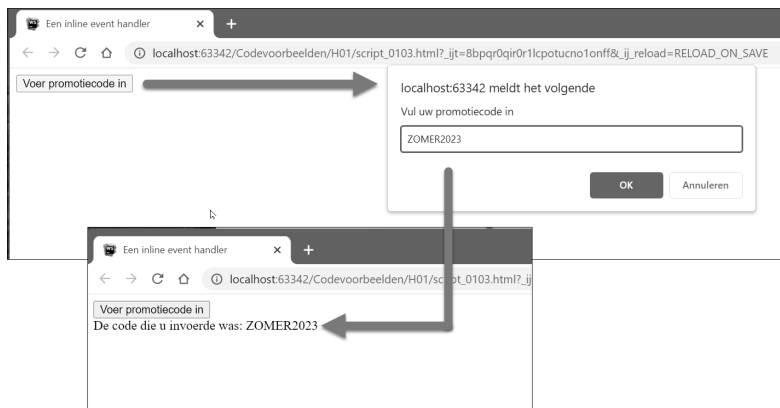
We passen het script nu zodanig aan dat het promptvenster pas wordt geopend op het moment dat de bezoeker op een knop klikt. Ook de uitvoer van het script wordt op die manier gebruikersafhankelijk. Het wordt niet meer direct uitgevoerd zodra de pagina wordt geopend. Hiervoor passen we verschillende dingen 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 promotiecode in</button>
<div id="divResult"></div>
<script>
// functie
function toonPrompt(){
  const code = prompt('Vu] 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.



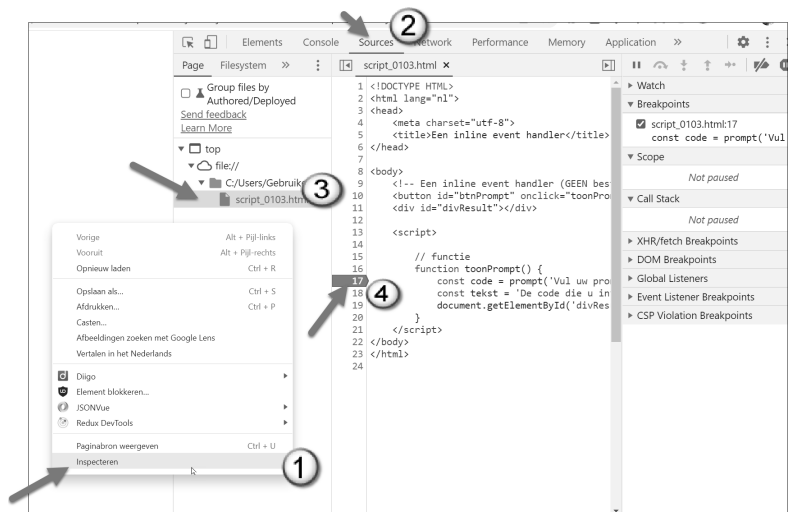
### Liever geen inline event handlers

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 om de nieuwste versie van Chrome te installeren via [google.nl/chrome](http://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 **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 17 in de marge te klikken. Er wordt een kleine blauwe pijl zichtbaar en het breekpunt verschijnt in het paneel Breakpoints aan de rechterkant.

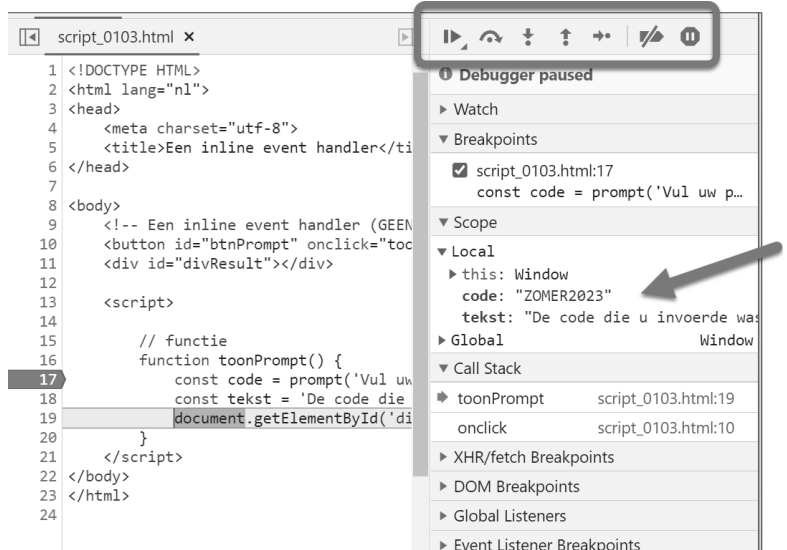


**Afbeelding 1.12** Open de Developer Tools met de opdracht *Element inspecteren*. Daarna kunt u in het tabblad *Sources* het juiste bestand selecteren en breekpunten plaatsen op de gewenste locatie(s).

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 In de debugger wordt nu regel 17 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 **Breakpoints**, **Scope** en **Call Stack** geopend.
  - 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.
  - 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 **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.
- 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.



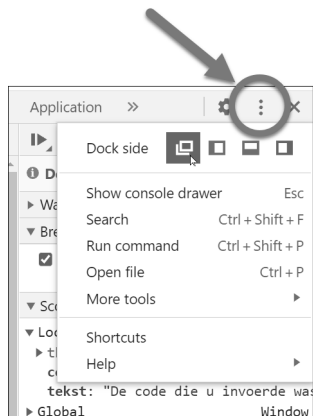
**Afbeelding 1.13** Twee statements zijn stap voor stap uitgevoerd. De inhoud van de variabelen is te zien in het paneel Scope.

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 links onder 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.



**Afbeelding 1.14** De Developer Tools kunnen in een eigen venster worden geplaatst of vastgekoppeld zijn aan het hoofdvenster. Er staan ook andere handige opdrachten en sneltoetsen in het venstertje.

## JavaScript-code in extern bestand

Wilt u HTML en JavaScript graag gescheiden houden, dan kunt u de JavaScript-code in een apart bestand laten staan. Dit werkt ongeveer net zo als het opslaan van CSS-code in een extern stijlenbestand. In het HTML-document plaatst u dan de volgende tag:

```
<script src="uwCodeBestand.js"></script>
```

Deze methode heeft als voordeel dat de HTML-documenten kleiner en overzichtelijker blijven. Code en structuur worden dan goed van elkaar gescheiden. Bovendien kunt u hetzelfde script in meerdere HTML-documenten gebruiken, zonder het in elk document afzonderlijk op te nemen. Wijzigingen hoeft u in dat geval maar in één bestand door te voeren (het JavaScript-broncodebestand).

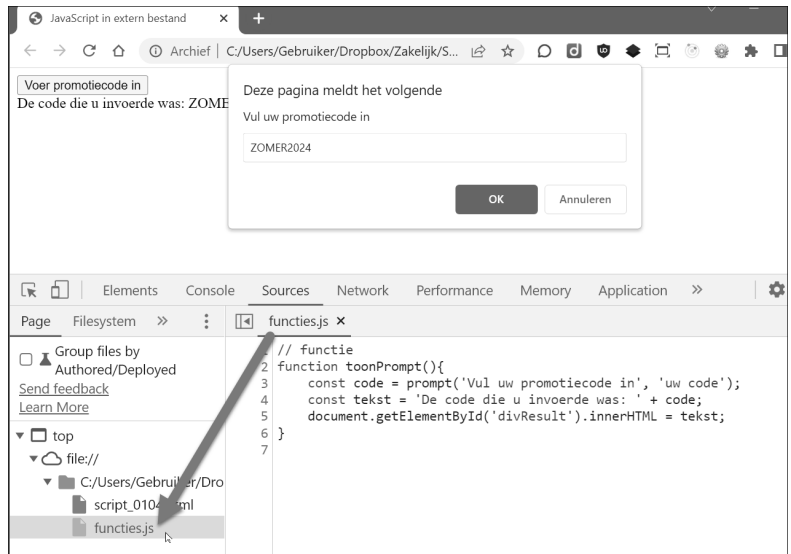
Het voorgaande voorbeeld kan op de volgende manier worden omgezet naar een extern bestand. De functie `toonPrompt()` zouden we dan vanuit verschillende pagina's kunnen aanroepen. Let erop dat in het bestand functies.js géén `<script>`-tags gebruikt worden.

```
<body>
  <button id="btnPrompt" onclick="toonPrompt();">Voer promotiecode in</button>
  <div id="divResult"></div>
  <script src="functies.js"></script>
</body>
```



## Script-tag afsluiten

Let erop dat de verwijzing naar `functies.js` een complete tag `<script src="..."></script>` moet zijn. Het is niet toegestaan de korte notatie te gebruiken door de tag direct af te sluiten. De notatie `<script src="..." />` is dus ongeldig.



**Afbeelding 1.15** Als het JavaScript in een gekoppeld bestand staat, is dit ook te zien in de boomstructuur van de Developer Tools.

Ook in de debugger is te zien dat het script nu in een afzonderlijk bestand staat. In de boomstructuur zijn zowel het HTML-bestand als het JavaScript-bestand te zien. Plaats uw breekpunten in dit geval ook in het `.js`-bestand

## Conclusie

Tot zover een snelle eerste kennismaking met JavaScript. In de volgende hoofdstukken leert u meer over de achtergronden van JavaScript en de syntaxis. Wilt u online vast wat meer informatie opzoeken, lees dan bijvoorbeeld:

- [nl.wikipedia.org/wiki/JavaScript](https://nl.wikipedia.org/wiki/JavaScript) Het (beknopte) Wikipedia-artikel over JavaScript.
- [www.voorbeginners.info/javascript/](http://www.voorbeginners.info/javascript/) Een Nederlandstalige beginners-cursus over JavaScript.

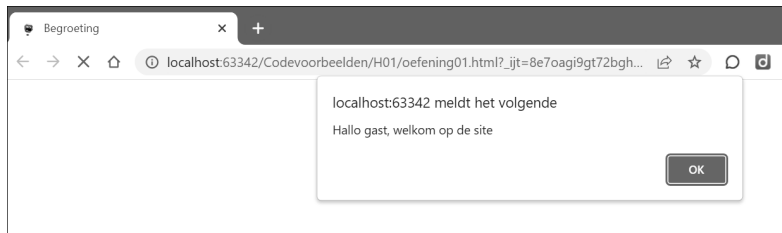
- [developer.mozilla.org/en/JavaScript](https://developer.mozilla.org/en/JavaScript) Voor de echte programmeur: de officiële naslaggids (*reference guide*) met alle JavaScript-opdrachten en -functies). Alleen Engelstalig.
- [www.w3schools.com/js/](http://www.w3schools.com/js/) De tutorial bij w3schools is iets toegankelijker, maar ook minder volledig. Alleen Engelstalig.

## Praktijkoefeningen

Met dit boek willen we graag bereiken dat u in een wat breder verband met JavaScript en later met jQuery aan de slag gaat. Aan het einde van de hoofdstukken vindt u daarom een aantal praktijkoefeningen. Zo kunt u nog eens op een andere manier herhalen wat u in het hoofdstuk hebt gelezen. De oefeningen zijn bedoeld om u verder op weg te helpen met JavaScript-voorbeelden. U kunt ze onbeperkt uitbreiden of aanpassen.

De uitwerkingen van de oefeningen zijn meestal *niet* opgenomen in de code-voorbeelden die u bij dit boek kunt downloaden, het is de bedoeling dat u zelf de uitwerkingen maakt.

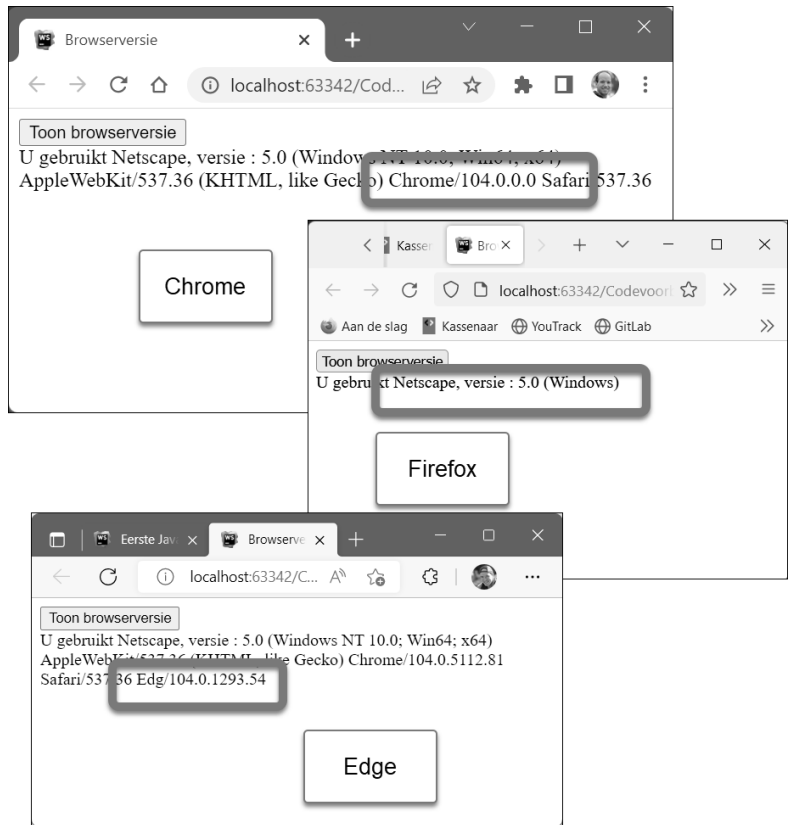
- 1 Onderzoek welke versie van JavaScript door uw standaardbrowser ondersteund wordt.
- 2 Kunt u een pagina maken waarbij de bezoeker met een welkomstboodschap wordt begroet zodra hij de pagina opent? (Waarschuwing: doe dit op een serieuze site liever niet op deze manier; weinig internetters zullen dit een leuke gadget vinden).



**Afbeelding 1.16** Een JavaScript-alert heet de bezoeker welkom.

- 3 Met het standaardobject `navigator` kunt u informatie verkrijgen over het type browser dat de lezer gebruikt (welk platform, welke versie, enzovoort). Met `navigator.appName` krijgt u een korte versie van de naam van de browser, met `navigator.appVersion` krijgt u een lange versie. Gebruik deze eigenschappen om een script te schrijven dat bij een klik op de knop de huidige browserversie op het scherm toont zoals is te zien in de afbeelding.





**Afbeelding 1.17** Verwerk de browserversie van de bezoeker in de webpagina. Verschillende browsers laten verschillende uitvoer zien. Ze 'liegen' allemaal dat ze Netscape zijn. Veel oudere websites checken nog op browserversie en als er geen Netscape in navigator.appName staat weigeren ze te laden.

- 4 Neem een van de oefeningen die u hiervoor hebt gemaakt en plaats deze in een extern JavaScript-bestand. Structuur en code worden zo beter van elkaar gescheiden. De pagina/code moet wel blijven werken!
  - In het JavaScript-bestand staat alleen JavaScript-code, geen tags `<script>...</script>`.
  - Het is een afspraak dat het JavaScript-bestand eindigt op `.js`.
  - Plaats het JavaScript-bestand in dezelfde map als de webpagina waarin het wordt ingevoegd.