**Voorbereiden**

|  |
| --- |
| Een functie (of methode) is een wezenlijk onderdeel van JavaScript. Functies maken het mogelijk dezelfde code op meerdere plaatsen in te zetten. Door een functie een passende naam te geven, kunt u de begrijpelijkheid van uw code vergroten. Functies kunnen heel flexibel inzetbaar zijn door ze argumenten te geven. Een functie is als een werknemer die precies uitvoert wat u hebt bepaald, simpelweg door zijn naam te noemen. Kortom, door functies wordt uw leven als programmeur een stuk aangenamer.  In dit hoofdstuk leert u functies te schrijven, met en zonder parameters, met en zonder return values. Deze kennis past u toe op event-handlers, en u maakt kennis met drie manieren om event-handling te implementeren. Ook leert u JavaScript-projecten te maken in een geïntegreerde ontwikkelomgeving.  Dit hoofdstuk bevat online oefening met voorgedefinieerde scripts, waarbij u direct resultaat ziet en zelf kleine wijzigingen kunt aanbrengen in de code.  Dit hoofdstuk vormt de grondslag voor het verder ontdekken van de mogelijkheden van JavaScript in de volgende hoofdstukken.  **Leerdoelen** Na dit hoofdstuk:   * kunt u functies definiëren * kunt u argumenten doorgeven aan functies * weet u wat return values zijn * kent u het onderscheid tussen lokale en globale variabelen * kunt u ingebouwde functies gebruiken, zoals een aantal methoden van Date * leert u hoe u de uitvoer van getallen kunt verfraaien * kunt u op diverse manieren event-handlers implementeren * kunt u gebruikmaken van een ontwikkelomgeving voor JavaScript * kunt u oplossingen testen in verschillende browsers.   Dit hoofdstuk bevat uitleg over een aantal begrippen en nodigt u uit delen uit de verplichte literatuur (Flanagan) te bestuderen en online oefeningen uit te voeren via de website ww3.schools.com.  Aan het eind van dit hoofdstuk maakt u een inzendopgave, waarin u een groot aantal aspecten uit dit hoofdstuk moet toepassen.  *Ingeschatte tijdsduur: 15 minuten.*  Bovenkant formulier  **Status**  U hebt deze activiteit voltooid.  Onderkant formulier |

**Bestuderen**

|  |
| --- |
| Een functie is een stukje code met een naam. Door de naam van de functie in een script te noemen, wordt de betreffende code uitgevoerd. Dat heeft onder andere als voordeel dat u hetzelfde stukje code meerdere malen kunt laten uitvoeren, door telkens de naam van de functie te noemen. Dat maakt de code korter. Het noemen van de naam van een functie heet het *aanroepen* van de functie (function call). Een ander voordeel is dat, als de naam van de functie goed gekozen is, het de code makkelijker te begrijpen maakt.  Er zijn in JavaScript al veel functies voorgedefinieerd. Die kunt u direct in uw eigen scripts gebruiken. In de voorbeelden en oefeningen in het vorige hoofdstuk hebben we daar al veel gebruik van gemaakt, misschien vaak zonder het te beseffen.  Het schrijven van een nieuwe functie is niet moeilijk. Ook dat hebt u in het vorige hoofdstuk al gedaan door een functie te definiëren die als event-handler fungeert, bijvoorbeeld na het klikken op een knop. In veel gevallen heette die functiemyFunction().  In dit hoofdstuk gaan we ons wat intensiever bezighouden met het zelf schrijven van functies.  Lees [hier](https://www5.loi.nl/access/content/group/255889f1-27c3-4e90-944f-11761ec46ec4/Bestanden%20SA/Functies.pdf) meer over functies.  *Ingeschatte tijdsduur: 60 minuten.*  Bovenkant formulier  **Status**  U hebt deze activiteit voltooid.  Onderkant formulier |

**Oefenen**

|  |
| --- |
| In de volgende oefenopgave past u de zojuist geleerde kennis toe. Het is erg belangrijk de oefenopgave eerst zelf te maken, voordat u de uitwerking bekijkt.  Maak opgave 1, de opgave vindt u [hier](https://www5.loi.nl/access/content/group/255889f1-27c3-4e90-944f-11761ec46ec4/Bestanden%20oefenopgaven/Oefenopgaven_hoofdstuk_3.pdf).  Controleer uzelf aan de hand van [deze uitwerkingen](https://www5.loi.nl/access/content/group/255889f1-27c3-4e90-944f-11761ec46ec4/Bestanden%20oefenopgaven/Uitwerkingen_oefenopgaven_H3.pdf).  *Ingeschatte tijdsduur: 20 minuten.*  Bovenkant formulier  **Status**  Klik Voltooien als u deze activiteit hebt afgerond.  Onderkant formulier |

**Oefenen**

|  |
| --- |
| Test uw parate kennis over functies via:   * [Functions](http://www.asmarterwaytolearn.com/js/35.html) * [Functions: passing them data](http://www.asmarterwaytolearn.com/js/36.html).   *Ingeschatte tijdsduur: 60 minuten.*  Bovenkant formulier  **Status**  Klik Voltooien als u deze activiteit hebt afgerond.  Onderkant formulier |

**Bestuderen**

|  |
| --- |
| Sommige functies verrichten een taak, zoals iets in een HTML-pagina zetten, andere functies leveren een waarde af die verder in JavaScript gebruikt kan worden.  **Een functie die een waarde aflevert** De functies die u hiervoor hebt bestudeerd, berekenden iets en zorgden er ook voor dat de uitkomst op het scherm te zien was. Dat is niet altijd handig. Een functie kan ook iets berekenen, of opvragen, en uiteindelijk een waarde afleveren zonder dat deze op het scherm te zien is. Die waarde kan dan in verdere delen van het script gebruikt worden. Dat wat een functie aflevert, hoeft niet per se een getal te zijn, maar kan ook een string zijn of een object (meer over objecten in volgende hoofdstukken).  Een voorbeeld is een functie die het gemiddelde berekent van twee getallen:  function gemiddelde( a, b ) {   var g = (a + b) / 2;   return g; }  De functie berekent eerst het gemiddelde van a en b, en bergt deze waarde op in g. Met de opdracht return g levert de functie de waarde van g af. Deze waarde wordt de *terugkeerwaarde*of *return value* genoemd. Die waarde moet u op het scherm zetten, opvangen in een variabele of er iets anders mee doen, anders gaat hij verloren. Bijvoorbeeld:  var x = gemiddelde(100, 200);  In deze opdracht wordt de functie gemiddelde() aangeroepen met argumenten 100 en 200. Deze waarden komen in de parameters a en b van de functie terecht. In die volgorde, dus a=100 en b=200. De functie berekent het gemiddelde en levert dat af: de afgeleverde waarde (150) komt in de variabele x terecht.  Bestudeer [w3schools.com - functions](http://www.w3schools.com/js/js_functions.asp).  Als een functie parameters heeft, moet u hem in principe met evenzoveel argumenten aanroepen. De waarde van het eerste argument komt in de eerste parameter terecht, het tweede argument in de tweede parameter et cetera.  Als u een functie aanroept met minder argumenten dan er parameters zijn, krijgen de ontbrekende parameters de waarde *undefined*. Het is dan maar de vraag of de functie met deze ongedefinieerde waarden zijn werk goed kan doen. In het algemeen is het dus verstandig alle parameters van een functie in de aanroep een waarde te geven.  Parameters van een functie gedragen zich als lokale variabelen van die functie, zie het begrip lokale variabele.  **De scope van variabelen en hoisting** Een variabele die met var **buiten**een functie is gedeclareerd, heet een *globale*variabele. Hij is overal op de pagina bekend en bruikbaar.  Een variabele die met var **in de body** van een functie is gedeclareerd, heet een*lokale*variabele. Hij is uitsluitend bekend en bruikbaar in de body van de functie.  Het gedeelte van de code waar een variabele bekend is en gebruikt kan worden, heet de *scope*van de variabele. Een globale variabele heeft de hele pagina als scope, een lokale variabele heeft de body van de functie waarin hij is gedeclareerd als scope. Lokale variabelen verdwijnen op het moment dat de functie met zijn werk klaar is. Globale variabelen verdwijnen op het moment dat de gebruiker naar een andere pagina gaat, of het venster van de pagina sluit.  Zie ook [w3schools.com - scope](http://www.w3schools.com/js/js_scope.asp).  In JavaScript kunt u een variabele gebruiken en op een aantal regels daaronder pas declareren, dit heet *hoisting*. Hoisting betekent hijsen, de declaratie wordt als het ware naar omhoog gehesen. In feite komt het erop neer dat de browser eerst de hele pagina doorleest en alle declaraties verzamelt en doet alsof ze aan het begin van hun scope gedeclareerd zijn. Dus alle lokale variabelen van functies zijn als het ware aan het begin van de body van de betreffende functie gedeclareerd, en alle globale variabelen zijn als het ware aan het begin van de pagina gedeclareerd.  Het is voor JavaScript-programmeurs daarom aan te bevelen declaraties van variabelen zo veel mogelijk aan het begin te doen (van de pagina of van een functie), omdat de code dan de intentie van JavaScript goed weerspiegelt. Een variabele eerst gebruiken en hem dan pas declareren, getuigt van een slechte programmeerstijl.  Als in een pagina een globale en een lokale variabele dezelfde naam hebben, heeft binnen de hele body van de functie de lokale variabele voorrang. De scope van de globale variabele is dan de hele pagina, behalve de body van die functie. In de volgende code ziet u daar een voorbeeld van.  <script> var x;  // globale variabele x x=10;  function test() {   var x;   x = 5;   alert("Lokale variabele " + x); }  test(); alert("Globale variabele " + x); </script>  Het eerste pop-up-venster laat de waarde 5 zien, het tweede de waarde 10.  Als een lokale variabele niet aan het begin van de body van de functie gedeclareerd wordt, zoals in het volgende voorbeeld, heeft die variabele toch de hele body van de functie als scope:  <script> var x;  // globale variabele x x=10;  function test() {   alert("Lokale variabele " + x);   var x;   x = 5; }  test(); alert("Globale variabele " + x); </script>  Het eerste pop-upvenster laat *undefined*zien (hoisting van de lokale x, maar deze heeft nog geen waarde en is dus undefined), het tweede laat 10 zien.  Zelfs als u de lokale variabele gelijk met de declaratie initialiseert, levert dat hetzelfde resultaat:  function test() {   alert("Lokale variabele " + x);      // undefined   var x = 5; }  Hoisting vindt alleen plaats wat betreft de declaratie, niet wat betreft de initialisatie.  Zie ook [w3schools.com - hoisting](http://www.w3schools.com/js/js_hoisting.asp) en Flanagan paragraaf 3.10.1 (Function scope and Hoisting).  *Ingeschatte tijdsduur: 60 minuten.* |

**Oefenen**

|  |
| --- |
| In de volgende oefenopgaven past u de zojuist geleerde kennis toe. Het is erg belangrijk de oefenopgaven eerst zelf te maken, voordat u de uitwerking bekijkt.  Maak opgave 2 t/m 4, de opgaven vindt u [hier](https://www5.loi.nl/access/content/group/255889f1-27c3-4e90-944f-11761ec46ec4/Bestanden%20oefenopgaven/Oefenopgaven_hoofdstuk_3.pdf).  Controleer uzelf aan de hand van [deze uitwerkingen](https://www5.loi.nl/access/content/group/255889f1-27c3-4e90-944f-11761ec46ec4/Bestanden%20oefenopgaven/Uitwerkingen_oefenopgaven_H3.pdf#page=2).  *Geschatte tijdsduur: 60 minuten.*  Bovenkant formulier  **Status**  Klik Voltooien als u deze activiteit hebt afgerond.  Onderkant formulier |

**Oefenen**

|  |
| --- |
| Test uw parate kennis over het tweede gedeelte over functies via:   * [Functions: passing data back from them](http://www.asmarterwaytolearn.com/js/37.html) * [Functions: local vs. global variables](http://www.asmarterwaytolearn.com/js/38.html).   *Ingeschatte tijdsduur: 60 minuten.*  Bovenkant formulier  **Status**  Klik Voltooien als u deze activiteit hebt afgerond.  Onderkant formulier |

**Bestuderen**

|  |
| --- |
| U hebt zich nader verdiept in functies in JavaScript. Functies zijn erg wezenlijk en er valt nog veel meer over te zeggen. Flanagan biedt hierover ook veel achtergrondinformatie, er is een heel hoofdstuk aan gewijd. Houd u er rekening mee dat dit boek niet in de eerste plaats een leerboek is, maar een naslagwerk. Dit is een goed moment om enkele paragrafen uit Flanagan te lezen.  Bestudeer over functies:   * Flanagan: de inleiding van hoofdstuk 8 (Functions) * Flanagan: paragraaf 8.1 (Defining Functions) en paragraaf 8.2 tot en met 8.2.1 (Function Invocation).   *Ingeschatte tijdsduur: 60 minuten.* |

**Bestuderen**

|  |
| --- |
| Een event wordt in veel gevallen door een gebruiker veroorzaakt, handelingen met de muis of via het toetsenbord kunnen events veroorzaken die in de browser worden opgevangen en eventueel doorgegeven aan JavaScript. Event-handling is dus een wezenlijk deel van een interactieve webpagina. Er zijn veel soorten events, te veel om ze allemaal te bespreken. Maar als u er enkele kent en het principe begrijpt, is de rest niet erg moeilijk.  **Event-handlers** Een event (gebeurtenis) is als het ware een signaal dat afgaat op het moment dat er iets speciaals gebeurt. Vaak, maar niet altijd, is de gebruiker de veroorzaker van de event: klikken, de muis bewegen, scrollen, een venster vergroten of sluiten; het zijn allemaal events.  Waarschijnlijk is klikken de meest voorkomende event. Er kunnen dan, afhankelijk van de situatie,  zeer uiteenlopende dingen gebeuren. Wat er precies gebeurt na een event hangt af van wat er in het betreffende programma gedefinieerd is in de zogeheten *event-handler*.  In het algemeen is een event-handler een functie waarin staat wat er moet gebeuren als de event plaatsvindt.  Nogal verwarrend is dat JavaScript verschillende manieren kent om event-handlers te implementeren.  **De eerste manier** De eerste manier is het eenvoudigst (deze is gebruikt in veel van de voorbeelden op de site w3schools.com). Het aangeven van de event-handler gebeurt in een attribuut van het HTML-element waar de event kan plaatsvinden.  Bijvoorbeeld: <button type="button" onclick="myFunction()">  Na het klikken op de knop wordt de functie myFunction() aangeroepen.  Ander voorbeeld: <input type="text" id="inputgetal" onchange="myFunction()">  Na het invoeren van tekst wordt de functie myFunction() aangeroepen.  De belangrijke events beginnen allemaal met het woord on. Merk op dat het aangeven van myFunction() in HTML tussen aanhalingstekens staat. Tussen de aanhalingstekens kunnen in principe meerdere JavaScript-statement staan:  Bijvoorbeeld: <button onclick = "var x = 3; getElementById('demo').innerHTML=x" > Klik </button>  Tussen de aanhalingstekens staat in feite een script.  Voordeel van deze manier is dat het nogal kort is en voor kleine voorbeelden dus snel werkt. Ideaal om even iets te testen.  Zie [w3schools.com - events](http://www.w3schools.com/js/js_events.asp) en vooral ook [w3schools.com - dom\_events](http://www.w3schools.com/js/js_htmldom_events.asp). Bekijk tevens het gedeelte More Examples onder aan de pagina.  Nog meer voorbeelden staan op een overzichtelijke manier op [w3schools.com - events\_examples](http://www.w3schools.com/js/js_events_examples.asp).  *Ingeschatte tijdsduur: 50 minuten* |

**Oefenen**

|  |
| --- |
| In de volgende oefenopgave past u de zojuist geleerde kennis toe. Het is erg belangrijk de oefenopgave eerst zelf te maken, voordat u de uitwerking bekijkt.  Maak opgave 5, de opgave vindt u [hier](https://www5.loi.nl/access/content/group/255889f1-27c3-4e90-944f-11761ec46ec4/Bestanden%20oefenopgaven/Oefenopgaven_hoofdstuk_3.pdf).  Controleer uzelf aan de hand van [deze uitwerkingen](https://www5.loi.nl/access/content/group/255889f1-27c3-4e90-944f-11761ec46ec4/Bestanden%20oefenopgaven/Uitwerkingen_oefenopgaven_H3.pdf#page=4).  *Geschatte tijdsduur: 20 minuten.*  Bovenkant formulier  **Status**  Klik Voltooien als u deze activiteit hebt afgerond.  Onderkant formulier |

**Bestuderen**

|  |
| --- |
| **De tweede manier** Nadeel van de eerste manier van event-handling is dat het in grotere toepassingen nogal onoverzichtelijk wordt. Het belangrijkste bezwaar is dat HTML en JavaScript gemengd worden. Om dezelfde reden dat CSS (opmaak) en HTML (inhoud) van elkaar gescheiden worden, is het ook nuttig JavaScript (functionaliteit) te scheiden van de rest.  Door het scheiden van HTML en JavaScript wordt foutopsporing (debugging) eenvoudiger en is de functionaliteit makkelijker aan te passen.  Lees [hier](https://www5.loi.nl/access/content/group/255889f1-27c3-4e90-944f-11761ec46ec4/Bestanden%20SA/De_tweede_manier.pdf) meer over de tweede manier.  *Geschatte tijdsduur: 45 minuten.*  Bovenkant formulier  **Status**  Klik Voltooien als u deze activiteit hebt afgerond.  Onderkant formulier |

**Oefenen**

|  |
| --- |
| In de volgende oefenopgave past u de zojuist geleerde kennis toe. Het is erg belangrijk de oefenopgave eerst zelf te maken, voordat u de uitwerking bekijkt.  Maak opgave 6, de opgave vindt u [hier](https://www5.loi.nl/access/content/group/255889f1-27c3-4e90-944f-11761ec46ec4/Bestanden%20oefenopgaven/Oefenopgaven_hoofdstuk_3.pdf).  Controleer uzelf aan de hand van [deze uitwerkingen](https://www5.loi.nl/access/content/group/255889f1-27c3-4e90-944f-11761ec46ec4/Bestanden%20oefenopgaven/Uitwerkingen_oefenopgaven_H3.pdf#page=5).  *Ingeschatte tijdsduur: 20 minuten.*  Bovenkant formulier  **Status**  Klik Voltooien als u deze activiteit hebt afgerond.  Onderkant formulier |

**Bestuderen**

|  |
| --- |
| **Aanroepen met this** De aanroep van myFunction in de vorige paragraaf kan ook met het keywordthis, in plaats van de namen van de vakken.    vak1.onfocus = function() {myFunction(this, "yellow")};   vak2.onfocus = function() {myFunction(this, "red")};  In de context van vak1.onfocus is this een verwijzing (referentie) naar vak1, en in de context van vak2.onfocus verwijst this naar vak2. We zullen this later nog vaker tegenkomen.  *Ingeschatte tijdsduur: 15 minuten.*  Bovenkant formulier  **Status**  Klik Voltooien als u deze activiteit hebt afgerond.  Onderkant formulier |

**Oefenen**

|  |
| --- |
| In de volgende oefenopgaven past u de zojuist geleerde kennis toe. Het is erg belangrijk de oefenopgaven eerst zelf te maken, voordat u de uitwerking bekijkt.  Maak opgave 7 en 8, de opgaven vindt u [hier](https://www5.loi.nl/access/content/group/255889f1-27c3-4e90-944f-11761ec46ec4/Bestanden%20oefenopgaven/Oefenopgaven_hoofdstuk_3.pdf).  Controleer uzelf aan de hand van [deze uitwerkingen](https://www5.loi.nl/access/content/group/255889f1-27c3-4e90-944f-11761ec46ec4/Bestanden%20oefenopgaven/Uitwerkingen_oefenopgaven_H3.pdf#page=5).  *Ingeschatte tijdsduur: 45 minuten.*  Bovenkant formulier  **Status**  Klik Voltooien als u deze activiteit hebt afgerond.  Onderkant formulier |

**Bestuderen**

|  |
| --- |
| Flanagan biedt veel informatie over events, hoofdstuk 17 is er in zijn geheel aan gewijd.  Leest u in elk geval de volgende paragrafen uit Flanagan om een indruk te krijgen van de veelzijdige achtergrond:   * paragraaf 17.1 (Types of Events) * nadere informatie over de verschillende typen events vindt u in paragraaf 17.4 (Document Load Events), 17.5 (Mouse Events), 17.6 (Mousewheel Events), 17.7 (Drag and Drop Events), 17.8 (Tekst Events) en 17.9 (Keyboard Events).   *Ingeschatte tijdsduur: 300 minuten.*  Bovenkant formulier  **Status**  Klik Voltooien als u deze activiteit hebt afgerond.  Onderkant formulier |

**Oefenen**

|  |
| --- |
| Test uw parate kennis over events via:   * [Events: button](http://www.asmarterwaytolearn.com/js/46.html) * [Events: mouse](http://www.asmarterwaytolearn.com/js/47.html) * [Events: fields](http://www.asmarterwaytolearn.com/js/48.html).   *Ingeschatte tijdsduur: 90 minuten.*  Bovenkant formulier  **Status**  Klik Voltooien als u deze activiteit hebt afgerond.  Onderkant formulier |

**Bestuderen**

|  |
| --- |
| **JavaScript in een apart bestand** Niet alleen is scheiden van content (HTML) en opmaak (CSS) een goed idee, ook scheiden van content en functionaliteit (JavaScript) is dat.  In de voorbeelden en oefeningen hebt u steeds HTML- en JavaScript-code in één bestand gezet. Naarmate bestanden groter en ingewikkelder worden, ligt het voor de hand de twee soorten code te scheiden. Dit kan vrij gemakkelijk, zie bijvoorbeeld[w3schools.com - whereto](http://www.w3schools.com/js/js_whereto.asp) onder het kopje External JavaScripts.  Zo'n extern scriptbestand heeft altijd de extensie js. In het externe js-bestand staan geen <script>-tags, die staan alleen in het HTML-bestand.  In veel gevallen kunt u de verwijzing naar het externe script vlak voor het einde van de body van het HTML-bestand zetten.  <!DOCTYPE html> <html>   <body>     ...      <script src="myScript.js"></script>   </body> </html>  Een andere plaats om de verwijzing naar het script te plaatsen, is in het <head>-gedeelte van het HTML-bestand. Dit betekent wel dat het script vrijwel als eerste door de browser gelezen en geïnterpreteerd wordt (*geparsed*). Tijdens het lezen van het script wordt het parsen van de HTML-uitgesteld. Het kan zijn dat in het script verwijzingen staan naar HTML-elementen die de browser nog niet gelezen heeft. Want pas als het script in zijn geheel is gelezen en de statements in het script die daarvoor in aanmerking komen, heeft uitgevoerd, gaat de browser verder met de HTML-tekst.  Kortom, de volgorde waarin script en HTML gelezen wordt, kan problemen opleveren. De plaats waar de verwijzing in het script staat, is dus belangrijk.  Veel van deze problemen kunnen worden opgelost door aan de <script>-tag het attribuut async of het attribuut defer toe te voegen. Zie voor een toelichting:[Growingwiththeweb](http://www.growingwiththeweb.com/2014/02/async-vs-defer-attributes.html" \t "_blank).  Het HTML-document komt er dan bijvoorbeeld zo uit te zien:  <!DOCTYPE html> <html> <head>     <title>Oefening 313</title>     <script async src="oefening313.js" ></script> </head>  <body>   ...  </body> </html>  In de meeste gevallen werkt het attribuut async goed.  *Ingeschatte tijdsduur: 30 minuten.*  Bovenkant formulier  **Status**  Klik Voltooien als u deze activiteit hebt afgerond.  Onderkant formulier |

**Oefenen**

|  |
| --- |
| In de volgende oefenopgave past u de zojuist geleerde kennis toe. Het is erg belangrijk de oefenopgave eerst zelf te maken, voordat u de uitwerking bekijkt.  Maak opgave 9, de opgave vindt u [hier](https://www5.loi.nl/access/content/group/255889f1-27c3-4e90-944f-11761ec46ec4/Bestanden%20oefenopgaven/Oefenopgaven_hoofdstuk_3.pdf#page=2).  Controleer uzelf aan de hand van [deze uitwerkingen](https://www5.loi.nl/access/content/group/255889f1-27c3-4e90-944f-11761ec46ec4/Bestanden%20oefenopgaven/Uitwerkingen_oefenopgaven_H3.pdf#page=7).  *Ingeschatte tijdsduur: 25 minuten.*  Bovenkant formulier  **Status**  Klik Voltooien als u deze activiteit hebt afgerond.  Onderkant formulier |

**Voorbereiden**

|  |
| --- |
| **Komodo Edit** Om op een overzichtelijk manier met aparte bestanden te kunnen werken, en om nog een tal van redenen, is het goed gebruik te maken van een speciaal voor dit doel ontworpen programma, een zogeheten *ontwikkelomgeving*. Daar bestaan er talloze van, betaalde en niet-betaalde. Een voor ons doel geschikte gratis ontwikkelomgeving is Komodo Edit. Voor het ontwikkelen van JavaScript-toepassingen is een goede ontwikkelomgeving (editor plus andere voorzieningen) onmisbaar.  Komodo Edit is een Integrated Development Environment (IDE, ontwikkelomgeving) waarmee u JavaScript-projecten kunt maken, opslaan, testen en veranderen. Komodo Edit is beschikbaar voor Windows, Mac OS X en Linux en u kunt het programma [hier](http://www.activestate.com/komodo-edit/downloads) downloaden. Let op: er zijn twee versies: de gratis versie heet Komodo Edit, de betaalde versie Komodo IDE. Voor ons doel voldoet de gratis versie.  Lees [hier](https://www5.loi.nl/access/content/group/255889f1-27c3-4e90-944f-11761ec46ec4/Bestanden%20SA/Beknopte_handleiding_KE.pdf) de beknopte handleiding van KE.  *Ingeschatte tijdsduur: 90 minuten.*  Bovenkant formulier  **Status**  Klik Voltooien als u deze activiteit hebt afgerond.  Onderkant formulier |

**Oefenen**

|  |
| --- |
| In de volgende oefenopgave past u de zojuist geleerde kennis toe. Het is erg belangrijk de oefenopgave eerst zelf te maken, voordat u de uitwerking bekijkt.  Maak opgave 10, de opgave vindt u [hier](https://www5.loi.nl/access/content/group/255889f1-27c3-4e90-944f-11761ec46ec4/Bestanden%20oefenopgaven/Oefenopgaven_hoofdstuk_3.pdf#page=2).  Deze oefenopgave heeft geen standaarduitwerking.  *Ingeschatte tijdsduur: 35 minuten.*  Bovenkant formulier  **Status**  Klik Voltooien als u deze activiteit hebt afgerond.  Onderkant formulier |

**Inzenden**

|  |
| --- |
| Dit hoofdstuk besluit met een inzendopgave die uit vier onderdelen bestaat. Maak uw oplossing in Komodo Edit. Splits de HTML-code en het script in twee bestanden. Test uw oplossing via KE in verschillende browsers voor u deze inzendt.  Stuur alle bestanden, zowel het script als HTML, in. Zend van elk onderdeel zowel het .html- als het .js-bestand in.  U ontvangt het beoordeelde werk binnen enkele dagen retour. Het resultaat kunt u vinden in de huiswerkresultaten.  *Ingeschatte tijdsduur: 210 minuten.*  Bovenkant formulier  **Status**  Klik Voltooien als u deze activiteit hebt afgerond.  Onderkant formulier |

**Inzendopgave 242S2**

**Onderdeel 1**

* Maak een pagina met daarop een tekstvak waarin de gebruiker een of meer bedragen kan invoeren door na elk bedrag op Enter te drukken. Maak na elke invoer het tekstvak leeg.
* De ingevoerde bedragen moeten bij elkaar worden opgeteld en hun totaal moet worden getoond in een tweede tekstvak.
* Scheid de event-handling van de HTML-code. Maak zo veel mogelijk gebruik van functies.
* Test uw oplossing vanuit KE in de verschillende browsers die op uw systeem zijn geïnstalleerd. Bekijk ook telkens het console van uw browser om te zien of daar interessante informatie staat.

**Onderdeel 2**  
Voeg aan de code controle van de invoer toe. Dat wil zeggen, controleer of er werkelijk een getal is ingevoerd. Zo niet, laat dan een boodschap verschijnen.

**Onderdeel 3**  
Voeg aan de code een knop toe waarmee u de btw kunt berekenen over de som van de ingevoerde bedragen van dit moment. U mag ervan uitgaan dat de ingevoerde bedragen inclusief btw waren.  
Laat de btw door een functie berekenen. Zet de btw en het bedrag zonder btw op de pagina.

**Onderdeel 4**  
Voeg aan de code een reset-knop toe waarmee u het totaal op 0 zet en de overige informatie op de pagina wist.