Inhoud

1 Hoofdstuk

Kennismaken met ECMAScript 2015

HTML is al ruim 25 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 van oudsher de aanvullende programmeertaal om HTML interactief te maken. Het is de populairste programmeertaal op internet. JavaScript staat daarmee aan de basis van elke techniek die webdevelopers moeten kennen. Het is de motor achter webapps als Gmail, Facebook, Instagram, webwinkels en meer. Maar JavaScript stamt ook alweer uit 1995. ECMAScript 2015 (ook wel 'ES6') is de opvolger van JavaScript en biedt nieuwe keywords en programmeermodellen die beter aansluiten bij de mogelijkheden van moderne browsers en de eisen die aan hedendaagse webapps worden gesteld. TypeScript breidt op zijn beurt ECMAScript 2015 weer uit met strikte typeringen en tools die we kennen uit programmeertalen als Java en C#. Dit hoofdstuk biedt een inleiding op beide technieken, waarna u vanaf het volgende hoofdstuk met de praktijk aan de slag gaat.

U leert in dit hoofdstuk:

Een korte geschiedenis van JavaScript en ECMAScript.

Waarom ECMAScript 2015?

Wat is TypeScript?

Benodigde voorkennis.

De werkomgeving instellen en aanvullende tools.

Indeling van dit boek.

Een korte geschiedenis van JavaScript en ECMAScript

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).

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 Facebook, Twitter, e-commerce of internetbankieren niet bestaan. Alle huidige browsers (Internet Explorer, Firefox, Chrome, 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.

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**.

ECMAScript 2015, 2016, 2017 en verder

Vroeger spraken we dus van JavaScript 1.0, 1.2 enzovoort. Tegenwoordig wordt gesproken van ECMAScript. Eerst kreeg ECMAScript ook versienummers (ECMAScript 3 en ECMAScript 5). Sinds 2015 wordt een jaartal gebruikt. Elk jaar in juni is er een congres waarin wordt besloten welke nieuwe features officieel worden toegevoegd aan de programmeertaal. Zo krijgen we ECMAScript 2015, ECMAScript 2016 en binnenkort ECMAScript 2017 (enzovoort).

Zolang voorgestelde nieuwe features nog geen deel uitmaken van de taal zelf, worden ze geschaard onder het kopje *ECMAScript Next*. De tabel geeft een overzicht van de geschiedenis van JavaScript en ECMAScript.

JavaScript/ECMAScript	Jaartal
1.0	1996
1.2	1997
1.5	1999/2000
ECMAScript 3	1999
ECMAScript 5	2009
ES6/ECMAScript 2015	2015
ECMAScript 7/2016/Next	2016 en verder

De begrippen ES6, ECMAScript 6 en ECMAScript 2015 worden vaak door elkaar gebruikt. Er wordt dan de versie uit 2015 bedoeld. Dat is niet erg handig, maar het is in de loop der jaren nu eenmaal zo gegroeid.

ECMAScript 4?

Misschien is het u opgevallen dat ECMAScript 4 in het overzicht ontbreekt. Dit was jarenlang een voorstel, maar de ontwikkeling is uiteindelijk gestopt. Dit had te maken met verschillen in inzicht binnen het comité over de richting die JavaScript op moest gaan en onnodige complexiteit van nieuwe onderdelen. ECMAScript 4 is dus nooit verschenen en er is geen browser die het ondersteunt. Men is direct verdergegaan met de ontwikkeling van ECMAScript 5.

Wat is ECMAScript 2015?

ECMAScript 2015 is een grote upgrade van JavaScript. In de taal zijn tal van nieuwe keywords en functies aanwezig. Nieuw zijn bijvoorbeeld keywords als `let`, `const`, `class`, `super` en meer. Ook kunt u gebruik maken van arrow functions en lambda

expressions. Met ECMAScript 2015 maakt JavaScript een enorme stap voorwaarts en wordt de taal echt volwassen.

JavaScript core

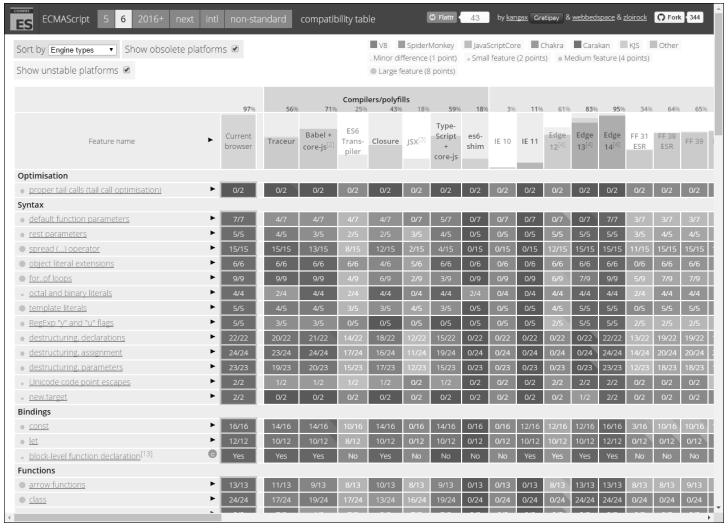
Het is belangrijk om u te realiseren dat ECMAScript 2015 (net als 'ouderwets' JavaScript) nog steeds bestaat uit een relatief kleine set kernopdrachten die worden uitgevoerd in een *hosting environment*. De kernopdrachten zijn gestandaardiseerd door ECMA. Dit zijn bijvoorbeeld `var`, `let`, `for`, `while`, `if` enzovoort. De hosting environment is in de meeste gevallen de browser. Maar JavaScript kan ook in andere omgevingen worden uitgevoerd. Denk aan JavaScript op de server (Node.js) of aan JavaScript in pdf, om interactieve pdf-documenten te maken. JavaScript – en dus ook ECMAScript 2015 – is niet slechts beperkt tot gebruik in de browser.

Onderdelen van ECMAScript 2015

Een handig overzicht van alle onderdelen van ECMAScript 2015 is te vinden in de *ECMAScript compatibility table*, die te vinden is op **kangax.github.io/compat-table/es6/**. (Googel op de trefwoorden `Kangax` en `es6`, dan is het het eerste resultaat in delijst.

De tabel uit afbeelding 1.1 is een prima naslagwerk en is als volgt opgebouwd:

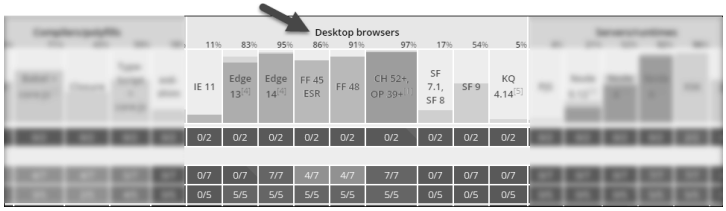
- Bovenin kiest u een versie. In de afbeelding is ES6 geselecteerd; dit is ook de standaardselectie.
- Aan de linkerkant zie u welke features deel uitmaken van ECMAScript 6. Dat zijn bijvoorbeeld onderdelen als default `function parameters`, `rest parameters` en verder naar beneden zaken als `arrow functions` en `class`. Veel van deze onderdelen worden in dit boek besproken.



Afbeelding 1.1 *De ECMAScript compatibility table geeft een overzicht van nieuwe JavaScript-features aan de linkerkant, en de mate waarin ze ondersteund worden door de verschillende browsers (de gekleurde blokjes).*

- Aan de rechterkant ziet u vervolgens in welke mate de betreffende feature door de verschillende omgevingen wordt ondersteund. Er zijn secties `Compilers/polyfills`, `Desktop browsers`, `Servers/runtimes` en `Mobile`.
- Een groen blokje betekent dat de feature deels of volledig wordt ondersteund door de betreffende omgeving, een rood blokje betekent dat dit niet het geval is.

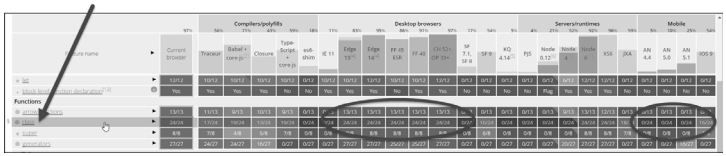
In de categorie `Desktop browsers` is bijvoorbeeld te zien dat (op het moment van schrijven van dit boek) Internet Explorer 11 slechts 11% van de nieuwe features ondersteunde, maar dat Edge, de nieuwe browser van Microsoft, maar liefst 95% van de ECMAScript 6-features goed verwerkt. Ook Firefox scoort ruim boven de 90% en Chrome en Opera scoren 97%.



Afbeelding 1.2 *Bekijk hoe de verschillende desktop browsers scoren bij de ondersteuning van ES6-kenmerken.*

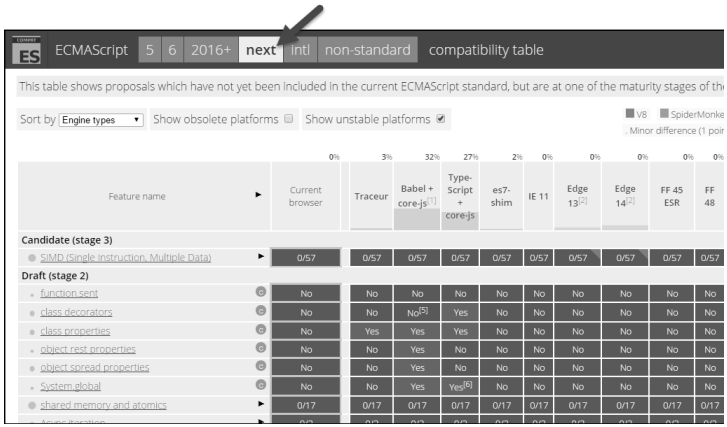
Kenmerken per feature

Door een bepaalde feature te kiezen, kunt u van links naar rechts bekijken hoe deze feature wordt ondersteund. In afbeelding 1.3 is bijvoorbeeld te zien dat het nieuwe keyword `class` (zie hoofdstuk 3 en 4) door moderne browsers goed wordt verwerkt, maar dat mobiele browsers (Android, iOS) het laten afweten. Dit betekent dat u `class` niet rechtstreeks kunt gebruiken in deze browsers.



Afbeelding 1.3 *Mobiele browsers scoren in het algemeen slecht op de ondersteuning van nieuwe ECMAScript 6-features.*

Kies boven in de tabel een ander versienummer (bijvoorbeeld 2016+ of next) om te zien welke nieuwe mogelijkheden voor de toekomst zijn voorgesteld en hoe de ondersteuning van browsers en andere omgevingen op dit moment is. Waarschijnlijk knalt een tabel met allemaal rode vakjes u tegemoet. Deze ondersteuning wordt in de loop der jaren ingebouwd, met elke nieuwe browser- of runtime-versie.



Afbeelding 1.4 *U kunt alvast zien welke onderdelen de komende jaren aan JavaScript worden toegevoegd. Ondersteuning door browsers en runtimes is op dit moment uiteraard nog zeer beperkt.*

Nieuwe features toch gebruiken

Als een vakje rood is weergegeven, betekent dit dat het niet door de aangegeven browser of runtime wordt ondersteund. Maar wat nu als u toch graag nieuwe onderdelen als `class` wilt gebruiken, terwijl uw publiek mogelijk beschikt over een oude browser (Internet Explorer 9, 10 of 11) of een smartphone met Android 4.4. Hoe zit dat dan? Moet u deze toevoegingen dan maar achterwege laten? Of erger nog, bij elke nieuwe eigenschap die u wilt toepassen urenlang in de tabel lopen zoeken en zelf een uitzonderingstabel opstellen? Gelukkig niet!

Compilers en polyfills

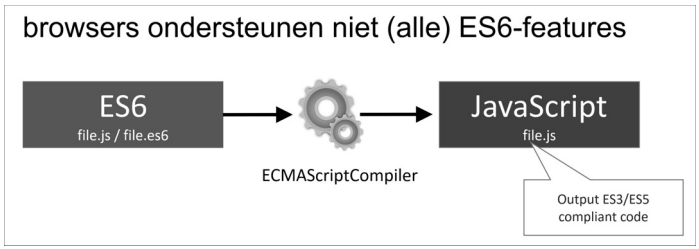
Als u een webapp ontwikkelt voor een intranet waarbij u zeker weet dat iedereen Edge gebruikt, of Chrome 50+, dan is er geen probleem. Gebruik zoveel ECMAScript 2015 als u wilt, want alle vakjes zijn groen. Maar in werkelijkheid wordt uw app waarschijnlijk door honderden of duizenden bezoekers gebruikt, die

allemaal over andere apparaten en browserversies beschikken. Dan schieten compilers en polyfills u te hulp.

Uitbreiding

ECMAScript 2015 is een uitbreiding op het oorspronkelijke JavaScript. Er is dus 100% overlapping.

- Polyfills zijn kleine scripts of codefragmenten die kunnen worden toegevoegd aan een HTML-pagina om nieuwe functionaliteit ook in oude browsers beschikbaar te maken. Ze vullen letterlijk het gat op dat in de browser aanwezig is.
- *Compilers* zijn programma's die het moderne ECMAScript 2015 dat u schrijft, omzetten naar klassiek ECMAScript 5, dat in alle browsers kan worden gebruikt. Een andere naam voor een compiler is ook wel *transpiler*, omdat deze nieuwe code terug-compileert naar 'oude' code.

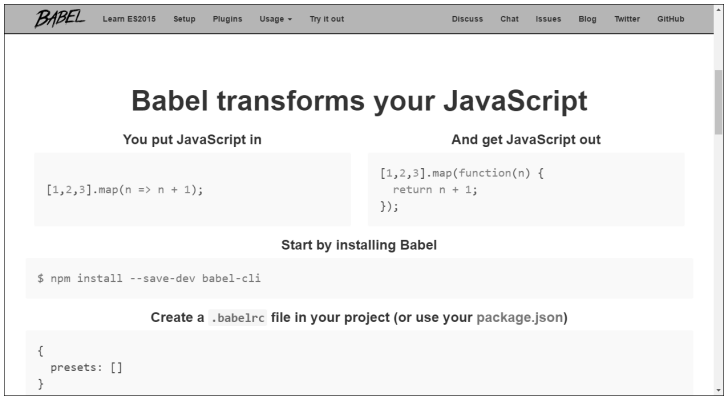


Afbeelding 1.5 *De werking van transpilers en polyfills. ES6-code wordt omgezet naar voor de browser begrijpelijke ES5-code*

Traceur, Babel en TypeScript

Er zijn verschillende compilers/transpilers beschikbaar. De bekendste zijn:

- **Traceur** Een product van Google. Door traceur.js toe te voegen aan de pagina, kunt u verderop in de pagina ECMAScript 2015-features gebruiken als classes, arrow functions enzovoort. Ze worden ter plekke in de browser vertaald. Voor betere prestaties kunt u traceur ook gebruiken op de server, of in een Grunt/Gulp-workflow. Meer informatie over traceur is te vinden op **github.com/google/traceur-compiler**.
- **Babel** Ook Babel (spreek uit: 'Babbel', niet 'Beebel') is een JavaScript-compiler. Het is een onafhankelijk opensource-product dat met configuratiescripts in hoge mate ingesteld kan worden. Veel mensen vinden de code die door Babel wordt gegenereerd beter leesbaar dan code die door traceur wordt gegenereerd, maar het gebruik en de configuratie van Babel 6 zijn lastig. Alles over Babel is te vinden op **babeljs.io**.



Afbeelding 1.6 *Babel is een bekende JavaScript-compiler, maar is lastiger in het gebruik.*

- **TypeScript** En jawel, ook TypeScript is een JavaScript-compiler. U kunt er ECMAScript 6-code mee schrijven, die door TypeScript wordt omgezet naar ES5-code. TypeScript ondersteunt nog niet álle ECMAScript 2015-features (zie de compatibility table), maar wel voldoende en is relatief eenvoudig in het gebruik. Bovendien biedt het gebruik van TypeScript tal van andere voordelen. Daar komen we in de rest van het boek uiteraard nog uitgebreid op terug. Het algemene adres van TypeScript is **typescriptlang.org**.

Afbeelding 1.7 *TypeScript omschrijft zichzelf als: 'JavaScript that scales. Any browser, any host, any OS'.*

Wat is TypeScript?

We hebben in dit boek niet voor niets gekozen voor TypeScript. Het is meer dan alleen een JavaScript-compiler. Door TypeScript te schrijven krijgt u de beschikking over een serie mogelijkheden die voorheen alleen voor applicatieprogrammeurs (Java, C#, Pascal/Delphi) waren weggelegd. Op de website omschrijft team TypeScript de tool als *JavaScript that scales*. De door TypeScript gegenereerde JavaScript-code is beschikbaar voor elke browser en elk besturingssysteem. TypeScript zelf is opensourcesoftware.

Wie maakt TypeScript?

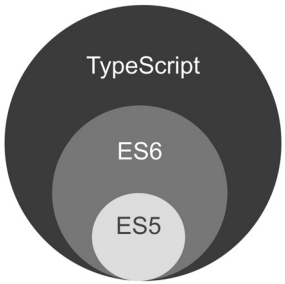
TypeScript is oorspronkelijk afkomstig uit de stal van Microsoft. En als u vindt dat TypeScript behoorlijk wat overeenkomsten lijkt te hebben met C#, dan hebt u het goed gezien. TypeScript is namelijk ontwikkeld door Anders Hejlsberg, de man die ook aan de wieg stond van de programmeertaal C# (2000). Daarvóór heeft Hejlsberg ook de talen Delphi (1995) en Turbo Pascal (1983) geschreven. Een zeer invloedrijk programmeur dus. Meer over Hejlsberg is te vinden op **en.wikipedia.org/wiki/Anders_Hejlsberg**. TypeScript is nu echter opensourcesoftware, iedereen kan de broncode inzien en verbeteren. TypeScript is daarmee eigenlijk van ons allemaal.

In het kort biedt TypeScript extra features in de vorm van data typing (oftewel: het datatype aangeven, dus afdwingen dat een bepaalde variabele van het type `string`, `number`, `boolean` of custom type is), type definition files om andere JavaScript-bibliotheken te kunnen gebruiken in een TypeScript-project, interfaces, enums, generics en nog veel meer. Uiteraard leest u hier veel meer over in de rest van het boek.

Omdat TypeScript vooraf wordt omgezet naar JavaScript, worden fouten in de code al in een vroeg stadium ontdekt. Dat is een groot verschil met de huidige generatie JavaScript-code, waarmee fouten pas run-time (of via uitgebreide testscripts met tal van if-statements), dus achteraf zichtbaar worden. Met TypeScript kunt u robuuste code schrijven die minder fouten bevat. Dit is uiteindelijk zowel voor de programmeur als voor de eindgebruiker een groot voordeel.

Relatie tussen ES5, ECMAScript 2015 en TypeScript

In afbeelding 1.8 is de relatie tussen het aloude ECMAScript 5, het nieuwe ECMAScript 6/ECMAScript 2015 en TypeScript goed te zien.



Afbeelding 1.8 *ECMAScript 2016 en TypeScript zijn beide supersets van gewoon JavaScript. Daarom kan elke geldige regel ES6 (of TypeScript) ook worden teruggebracht naar een geldige ES5-syntaxis. Leesbaarheid en vereenvoudiging van de code zijn belangrijke doelstellingen.*

- In de kern van de cirkels staat ES5. Dit is een – naar hedendaagse maatstaven – beperkte set opdrachten, die echter goed worden ondersteund door alle omgevingen: zowel browsers, mobiele apparaten als server-runtimes.
- ES6/ECMAScript 2015 is een uitbreiding van ES5. Dit betekent dus dat er 100% overlapping bestaat. Elke geldige regel ES5 is ook geldig in ECMAScript 2015. Omgekeerd is dat uiteraard niet het geval, daarvoor zijn de compilers/transpilers nodig. Maar het is belangrijk te realiseren dat ES6 in alle gevallen een *uitbreiding* van ES5 is. Er zijn geen nieuwe syntaxisregels, geen nieuwe tekens die verboden zijn, al uw JavaScript-kennis blijft geldig. Deze wordt alleen uitgebreid met nieuwe mogelijkheden.

- TypeScript breidt op zijn beurt ES5 en ES6 weer uit (met strikte typeringen, interfaces en al die andere zaken die we hierboven genoemd hebben). Oftewel: elke geldige regel ES5 is óók geldig in TypeScript. Dit is een groot voordeel, want het betekent opnieuw dat u geen afwijkende syntaxis of puntkomma's hoeft te leren. Alle JavaScript blijft geldig; er worden alleen nieuwe, extra mogelijkheden aan toegevoegd.

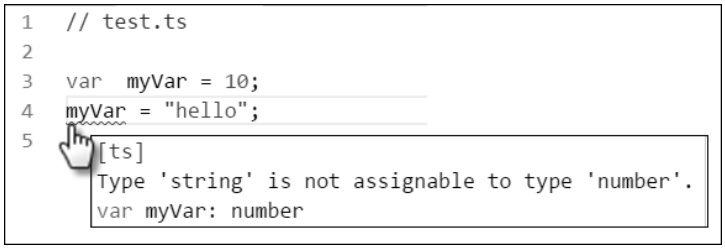
Alles is optioneel

Een ander kenmerk is dat zowel ES6 als TypeScript volledig optioneel is. Het werken met klassen, interfaces en zelfgedefinieerde typen is mogelijk (en het maakt uw leven als front-end programmeur een stuk aangenamer), maar het is geenszins verplicht.

Wat is een JavaScript-fout?

U weet waarschijnlijk dat alles in JavaScript van het type `var` is. Alle variabelen zijn *loosely typed*. U mag een statement schrijven als `var myVar = 'Hello'` (myVar is nu impliciet van het type `string`). Verderop mag u vervolgens schrijven `myVar = 10` (nu is `myVar` van het type `number`). In JavaScript is dit volledig geaccepteerd. TypeScript zal er echter over vallen en een compiler-error laten zien. Het script zal wel werken, maar waarschijnlijk is het ongewenst dat een variabele de ene keer een string en de andere keer een nummer bevat. TypeScript controleert dit soort zaken en waarschuwt u. Door alleen de bestandsextensie aan te passen van `*.js` naar `*.ts` wordt u al op dit soort fouten gewezen. U hoeft er geen regel code voor aan te passen.

We zullen het zeker niet aanbevelen, maar ook zonder één regel TypeScript te schrijven kunt u desgewenst JavaScript-code aan de TypeScript-compiler aanbieden. Mochten er logische fouten in de – bestaande – JavaScript-code staan, dan worden deze ook opgemerkt door de TypeScript-compiler.



Afbeelding 1.9 *TypeScript controleert het type van variabelen. Door alleen de extensie aan te passen naar .ts krijgt u direct inzicht in dit soort logische fouten. Ze komen veel voor in JavaScript-apps.*

```
// test.ts
var myVar = 10;
myVar = "hello"; // ts-error: "Type 'string' is not assignable to type 'number'
```

Overige mogelijkheden van ECMAScript 2015 en TypeScript komen in de rest van het boek uiteraard uitgebreid aan de orde. U kent nu echter de relatie tussen de technieken en weet waar u de begrippen kunt plaatsen in het landschap van webdevelopmenttechnieken.

Benodigde voorkennis

Dit boek maakt deel uit van de *Web Development Library* (**www.webdevelopmentlibrary.nl**). In elk deel wordt een op zichzelf staande techniek besproken die te maken heeft met webdevelopment. Andere, gerelateerde technieken worden bekend verondersteld. Zo betaalt u alleen voor datgene wat u echt nodig hebt.

In dit boek gaan we in op ECMAScript 2015 en TypeScript 1.8. TypeScript 2.0 is op het moment van schrijven van dit boek in bèta, maar bevat geen aanvullingen of wijzigingen die voor deze uitgave van belang zijn. ECMAScript 2015 en TypeScript zijn beide uitbreidingen op JavaScript, zoals u in de vorige paragraaf hebt gelezen.

U kent JavaScript

We gaan er dan ook van uit dat u JavaScript redelijk tot goed beheerst. U hebt al ervaring met het schrijven van JavaScript en eventueel het werken met aanvullende JavaScript-bibliotheken of -frameworks, zoals jQuery of Angular. Voorkennis van jQuery of Angular (of andere frameworks) is echter niet nodig. We gaan in dit boek niet in op zaken als variabelen, `for`- en `while`-lussen, voorwaardelijke statements (`if`) en dergelijke. Deze worden bekend verondersteld.

Wat hoeft u niet te weten?

Kennis van andere JavaScript-bibliotheken (zoals jQuery, Backbone, Knockout) of -frameworks is niet nodig. Ook hoeft u niks te weten van server-sided talen als PHP, Python, Ruby, Java of C#. Mocht u wel Java of C# kennen, dan zullen veel concepten in TypeScript u bekend voorkomen. Maar voorkennis van deze talen is niet per se nodig.

Bovendien worden ECMAScript en TypeScript, zoals u inmiddels hebt begrepen, gebruikt in de context van een website of webapp. U moet daarom enige kennis hebben van HTML en CSS. U moet een basispagina met tags, attributen, `console.log()`-statements enzovoort goed kunnen schrijven en begrijpen.

Tips voor meer lezen

Kan uw HTML- of JavaScript -voorkennis wel een opfrisbeurt gebruiken? Lees dan eerst de volgende titels (van dezelfde auteur):

- *Web Development Library – JavaScript*, ISBN 9789059407589
- *Web Development Library – HTML*, ISBN 9789059408081

Indeling van dit boek

Dit boek bestaat uit twee delen.

- **Hoofdstuk 1 tot en met 3** In deze hoofdstukken gaan we in op ECMAScript 2015/ES6. U leert op welke manier de standaard JavaScript/ES5-code wordt uitgebreid met nieuwe keywords en mogelijkheden. U schrijft 'gewoon' JavaScript-code, die direct kan worden uitgevoerd in moderne browsers zoals Chrome, Firefox en Edge.
- **Hoofdstuk 4 tot en met 7** Dit deel gaat in op TypeScript. De kennis uit de eerste hoofdstukken wordt dan bekend verondersteld. U leert hoe ES6-code verder wordt uitgebreid met TypeScript-specifieke kenmerken. Uw code moet eerst door de TypeScript-compiler bewerkt worden voordat hij uitvoerbaar is in de browser. Lees vooral de rest van dit hoofdstuk om hier alvast kort kennis mee te maken.