JS - Variabelen en constanten

**JavaScript is een dynamisch getypeerde taal die als onderdeel van een webapplicatie gehost wordt in een browser en dus door de browser geïnterpreteerd en uitgevoerd wordt.**

Gegevenstype

***Statisch getypeerd*** houdt in dat elke variabele een vast type is. Het type van een variabele kan tijdens de uitvoering van het programma niet veranderen. Bij het declareren van een variabele wordt het type ervan vastgelegd en dat type blijft gedurende de levensduur van het programma hetzelfde.

***Dynamisch getypeerd*** wil zeggen dat elke waarde een type heeft. Het type van een variabele iwordt afgeleid uit de waarde die er aan wordt toegekend. De waarde van een variabele kan tijdens z’n levensduur veranderen en dus ook het gegevenstype ervan. Bij een dynamisch getypeerde taal is het dus niet nodig bij declaratie van een variabele het gegevenstype ervan mee te geven.

Een statisch getypeerde taal heeft het voordeel dat veel fouten al gemeld kunnen worden tijdens het compileren. Een dynamisch getypeerde taal heeft het voordeel dat casten (expliciet aangeven dat een waarde van type Object behandeld moet worden alsof het van het type String is) niet nodig is. Hiermee stellen we robuustheid tegenover flexibiliteit en dat is een vaak voorkomend discussiepunt!

Variabelen

Een variabele heeft:

* een **naam**: de naam die aan de variabele in de code gegeven is;
* een **adres**: de naam van de variabele verwijst naar een geheugenadres;
* een **waarde**: de waarde die op het geheugenadres waarnaar de variabele verwijst is opgeslagen; een gegevenstype: bepaalt welk type waarde de variabele kan opslaan;
* een **bereik** (**scope**): bepaalt tot waar een variabele in de code "zichtbaar" is en dus gebruikt kan worden;
* een **levensduur**: is de tijd dat de variabele geassocieerd blijft met een bepaald geheugenadres. Deze levensduur begint op het moment dat de variabele gecreëerd wordt en eindigt op het moment dat hij gedumpt wordt door het programma zelf of expliciet door de programmeur in de code. De levensduur van een variabele betekent niet hetzelfde als het bereik van een variabele.

Naam

* hoofdlettergevoelig;
* combinatie van letters, cijfers, onderlijningsteken, maar mag niet beginnen met een cijfer;

Variabelen gebruiken

var

Variabelen die door var worden gedeclareerd hebben functiebereik en worden naar boven gehesen (hoisted to the top). Het betekent dat een variabele kan worden gebruikt vooraleer die gedeclareerd is.

Met var kan je:

* declareren  
  var getal;
* initialiseren  
  getal = 9;
* declareren en initialiseren
  + één variabele  
    var getal = 9;
  + meerdere variabelen in één keer  
    var getal1, getal2 = 4, naam1, naam2 = 'Jan', naam = 'Anna', getal = 1;

let

Het [let](https://developer.mozilla.org/nl/docs/Web/JavaScript/Reference/Statements/let) statement declareert een lokale variabele met block scope en initialiseert die eventueel met een waarde. Variabelen gedeclareerd met let worden niet naar de top gehesen en kunnen niet gebruikt worden vooraleer ze gedeclareerd zijn.

Hoe je var en let gebruikt lees je in Kyle Simpson, For and against 'let'.

Constanten

JavaScript (ES6) kent ook het keyword const dat wordt gebruikt voor constanten. Het is de gewoonte om de naam van een constante met hoofdletters te schrijven.

Constant Reference, geen Value

Het wordt vaak verkeerd geïnterpreteerd als een "constante waarde". Maar in ES6 is const een verwijzing naar een constante verwijzing naar een waarde (hetzelfde geldt in de meeste talen). Met andere woorden, de pointer die de naam van de variabele gebruikt kan niet veranderd worden in het geheugen, maar datgene waarnaar de variabele verwijst wel.

In de onderstaande code creëren we een nieuwe variabele met een constante verwijzing naar een array. We kunnen waarden toe voegen aan de array en aangezien dit niet de referentie wijzigt, lukt dat:

const namen = [];

namen.push ( "Jetmir");

console.log (namen);

Maar als we proberen om de variabele referentie naar de nieuwe array te wijzigen - zelfs één met dezelfde inhoud - krijgen we een SyntaxError ( "Assignment constant variabele"):

const namen = [];

namen = []; // Error!

Als je een const die verwijst naar een waarde van het primitieve gegevenstype, zoals een string of number, dan valt er niets veranderen aan die waarde. Alle methoden op String en Number retourneren nieuwe waarden (kopies van objecten).

Let er tenslotte op dat const dezelfde nieuwe scoping regels volgt als let!