Primitieve gegevenstypen

**Number, String en Boolean zijn primitieve typen. Een, variabele bestaat uit een naam en een toegekende waarde. Bij een toekenning van de waarde van de ene variabele aan een andere variabele krijgt de tweede variabele een kopie van de waarde als de waarde een primitief type heeft. Dat wil dus zeggen dat als de waarde verandert van de kopie, de oorspronkelijke waarde niet mee verandert.**

Beschrijving

Als je waarden in een programma wilt gebruiken kan je kiezen uit verschillende voorstellingen voor die waarden afhankelijk van wat je met die waarden wilt doen. Deze verschillende representaties voor waarden worden *types* genooemd. JavaScript heeft ingebouwde types de primitieve waarden:

Als je met waarden wilt rekenen gebruik je ***number***.

Als je een waarde wilt af drukken op het scherm of gewoon 'iets' wil neerschrijven gebruik je een ***string***, namelijk één of meer letters, woorden of zinnen.

Als je een beslissing wilt nemen in het programma, gebruik je *boolean*, namelijk true of false.

Waarden die direct in de broncode staan worden ***letterlijk* (literal)** genoemd. letterlijke tekenreeksen zijn omgeven door dubbele aanhalingstekens ( "...") of enkele aanhalingstekens ( '...') -het enige verschil is stilistische voorkeur. number en boolean literals worden weergegeven zoals ze zijn, bijvoorbeeld 42, true, etc.).

"Dit is een tekst." ;

'Nog een tekst' ;

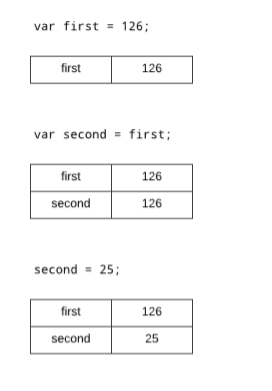
4;

true ;

false ;

By value not by reference

**Primitieve gegevenstypen zijn by value** en niet by reference in tegenstelling met objecten.



*Schema geheugen toekenningsopdrachten var*

Type

Elke waarde die je in een variabele stopt is van een bepaald type. In JavaScript zijn er slechts 5 primitieve gegevenstypen:

* **number**: floating point getallen, en ook de gehele getallen, voorbeeld: 1, 100, 3.14; er wordt geen onderscheid tussen gehele en gebroken getallen; er zijn geen typen zoals Integer of Double;
* **string**: een reeks van een willekeurig aantal tekens, bijvoorbeeld "a", "een", en "één 2 drie"; strings zijn reeksen karakters tussen enkelvoudige of dubbele aanhalingstekens.
* **boolean**: waar of onwaar t.t.z. true of false;
* **undefined**: als je de naam van een variabele gebruikt die niet bestaat, dit wil zeggen niet geïnitialiseerd is en ook niet gedeclareerd, krijg je de speciale waarde undefined. Als je een variabele declareert zonder er een aan toe te kennen initialiseert JavaScript de variabele achter de schermen met de waarde undefined.
* **null**: Dit is een ander speciaal gegevenstype met slechts één mogelijke waarde, namelijk de null-waarde. Het betekent geen waarde, een lege waarde of niets. Het verschil met undefined is dat als een variabele een waarde null heeft het nog steeds gedefinieerd is. De waarde ervan is gewoon weg niets. Om expliciet aan te geven dat een variabele geen waarde heeft, kunnen we dus null gebruiken.

Elke waarde die niet tot één van de vijf hier boven vermelde primitieve typen is een object.

Als we een variabele declareren maar niet initialiseren, zijn het type en de waarde van deze variabele undefined, wat we in een geheugendiagram aangeven met een ?:



*schema geheugen undefined value in variabele*

Typeof

Gebruik de typeof operator als je het het type van een variabele of een waarde wilt te weten komen.

Deze operator retourneert een tekenreeks die het gegevenstype vertegenwoordigt. De retourwaarden van typeof:

1. "number"
2. "string"
3. "boolean"
4. "undefined"
5. "object"
6. "function"

Waar zit de null? En wat doet function hier?

Merk op dat de eenvoudige primitieven (string, boolean, nummer, null en undefined) zelf geen objecten zijn. null wordt soms aangeduid als een object, maar dat is een misvatting die voortkomt uit een bug in de taal die ervoor zorgt dat typeof null de string "object" foutief retourneert. In feite, is null een eigen primitief type.

**De laatste ECMAScript standaard definieert zeven data types:**

1. **6 primitieve:**
   1. **Boolean**. true and false.
   2. **null**. A special keyword denoting a null value. Because JavaScript is case-sensitive, null is not the same as Null, NULL, or any other variant.
   3. **undefined**. A top-level property whose value is undefined.
   4. **Number**. 42 or 3.14159.
   5. **String**. "Howdy"
   6. **Symbol** (new in ECMAScript 6). A data type whose instances are unique and immutable.
2. **Object**

Object subtypen of complexe primitieven

Dat "alles in JavaScript een object is" is een veel voorkomende misvatting. Dat is duidelijk niet waar. Wel zijn er enkele bijzondere object subtypes, ,namelijk de *complexe primitieven*.

Het type function is een subtype van het object (technisch gezien een "callable object"). Van functies in JS wordt gezegd dat ze "first class" zijn, d.w.z. objecten met oproepbare gedragssemantiek op vastgeschroefd. Dat zorgt er wel voor dat ze kunnen worden gebruikt als elk ander object. (Kyle Simpson, 2014, p. 38)

Arrays zijn ook een subtype van objecten, met extra gedrag. De organisatie van de inhoud van arrays is iets meer gestructureerd dan voor algemene objecten.

Er zijn ook nog object subtypen die ingebouwde objecten worden genoemd.

* String
* Number
* Boolean
* Object
* Function
* Array
* Date
* RegExp
* Error
* Math

Afgaand op sommige namen zou je kunnen denken dat ze direct verband houden met hun eenvoudige primitieve tegenhangers, maar het is ingewikkelder dan dat zoals we zullen zien.