

Web technology

Les 10: JS - Inleidende les

Kristof Michiels

In deze presentatie

- JavaScript: inleiding
- Variabelen
- Wiskundige bewerkingen
- Booleans: true of false
- Voorwaardelijke statements

JavaScript: inleiding

Wat is JavaScript?

- JavaScript is de programmeertaal van het web
- Samen met HTML en CSS onderdeel van de drie pijlers
- JavaScript wordt gelezen en geïnterpreteerd door de browser van de bezoeker van de website (= client-side of front-end programming)
- In 2021: zeer belangrijke programmeertaal
- Wij gaan ons de komende weken bezig houden met de grondbeginselen van JavaScript
- Heel snel knopen we terug aan met web development en gaan we zien hoe we de DOM kunnen manipuleren

Hoe ziet JavaScript eruit?

Een basisvoorbeeld dat een zinnetje in de console doet verschijnen:

```
let deel1 = "We zijn hier om ";  
let deel2 = "JavaScript te leren!";  
console.log(deel1 + deel2); // We zijn hier om JavaScript te leren!
```

Wat is een script?

Een script is een programma dat een webpagina begeleid. Het wordt uitgevoerd door de web browser en maakt je pagina's dynamischer en krachtig.

```
<!doctype html>
<html>
  <body>
    <script>
      let deel1 = "We zijn hier om ";
      let deel2 = "JavaScript te leren!";
      document.write(deel1 + deel2);
    </script>
  </body>
</html>
```

Een JavaScript-programma bestaat uit statements

- De statements volgen elkaar op en worden één voor één chronologisch uitgevoerd
- Elk statement krijgt zijn eigen lijn en eindigt telkens op een puntkomma
- Een langer statement kan op meerdere regels worden geschreven
- Ongevoelig voor witruimte, net als html en CSS

```
let naam;  
let leeftijd;  
naam = "joe";  
leeftijd = 36;  
console.log(naam);  
console.log(leeftijd);
```

Commentaar in je code

- We kennen nog het nut van commentaar: verduidelijken van je code
- Wordt genegeerd door de browser
- Commentaar op 1 regel wordt voorafgegaan door een dubbele slash (//)
- Commentaar op meerdere regels: /* Hier komt je commentaar */

```
// Script om naam af te drukken
let naam, leeftijd;
naam = "Joe";
leeftijd = "36";
console.log(naam);
/* Dit commentaar kan
doorlopen over meerdere regels */
```


JavaScript: output van resultaten

- Verschillende manieren om data te vertonen in JS. Wij zullen 2 manieren gebruiken: de browser console en uiteraard web pagina's
- De browserconsole: weergave > Ontwikkelaar > JavaScript console OF...
- Een nieuwe template bestaande uit een html-, css- en js-bestand
- De template staat op Digitap. Jij gaat op dit moment enkel in het js-bestand werken.
- Je opent in de browser en krijgt een wit scherm
- Je opent de browserconsole. console.log-statements zal je zien verschijnen in de console

```
console.log("Hallo wereld");
```

Variabelen

Variabelen

- Variabelen zijn containers die ons helpen informatie voor later te op te slaan en te onthouden
- We maken hen aan met het woord *let* gevolgd door een naam, "identifier" genoemd
- We geven hen een waarde met het gelijkheidsteken (=), dat we in deze context de toekenningoperator noemen
- We kunnen de waarde tijdens de looptijd van ons programma veranderen
- JavaScript regelt zelf welk datatype het ziet in een variabele. Je moet dat dus niet gaan declareren

```
let naam = "Ben";  
let leeftijd = 23;  
console.log(naam); // Ben
```

Variabelen

- De naam of "identifier" hoort uniek te zijn en moet starten met een kleine letter
- Mag niet starten met een getal, maar mag wel getal bevatten
- Sommige woorden zijn gereserveerd en dus niet geschikt om gebruikt te worden: if, else true, float, switch, ...
- We kunnen de variabele eerst aanmaken en pas later vullen: eigen keuze
- We kunnen ook meerdere variabelen aanmaken binnen hetzelfde statement

```
let familienaam;  
familienaam = "Van de Vijver";  
let leeftijd = 21, hobby = "programmeren";
```

JavaScript naamgeving

- Programmeurs gaan vaak gebruik maken van camelcasing
- Eerste woorddeel begint met een kleine letter, daarna telkens een hoofletter

```
let cirkelDiameter = 3;  
let ontstaansGeschiedenisDatum = 1830;  
console.log(cirkelDiameter);
```

Strings

- Variabelen kunnen verschillende soorten informatie bevatten
- Strings zijn opeenvolgingen van karakters = eigenlijk tekst
- Strings kunnen elke waarde bevatten, zolang we ze schrijven tussen enkele of dubbele aanhalingstekens
- Maak een keuze en doe het op dezelfde manier. Ik raad dubbele aanhalingstekens aan.

```
let naam = "joe";  
let taal = 'Nederlands';  
console.log(naam);  
console.log(taal);
```

Gehele getallen of integers

- Wij creëren onze variabelen op de juiste manier, en JavaScript maakt er het juiste datatype van
- Zo bvb een getal
- Getallen kunnen gehele getallen zijn. Gehele getallen noemen we integers

```
let getal1 = 39;  
let getal2 = 12;  
console.log(getal1 * getal2);  
let getal3 = "112"; // geen getal maar een tekst, door gebruik aanhalingstekens
```

Decimale getallen of floats

- Decimalen kunnen ook
- We noemen ze floats
- We gebruiken punten ipv komma's
- Een decimaal getal in JS kan tot 17 cijfers bevatten

```
let hoogte = 1.72;  
let pi = 3.141592653589793;  
let fontSizeInRem = 1.65;
```


typeof()

- Als we willen weten welk soort data type onze variabele bevat?
- Geeft niet de waarde terug, maar wel welk soort datatype onze variabele is

```
let naam = "Bruno";  
console.log(typeof(naam)); // geeft string terug
```

Wiskundige bewerkingen

Wiskundige bewerkingen

- We kunnen rekenen met de getalwaarden in onze variabelen
- We kunnen +, -, * en / gebruiken
- We noemen dit wiskundige operatoren
- We gebruiken ze om wiskundige operaties of bewerkingen uit te voeren

```
let r = 4 + 2;  
let pi = 3.141592653589793;  
console.log(2 + r * pi);
```

Wiskundige bewerkingen

```
let basisRechthoek = 3;  
let hoogteRechthoek = 2;  
let oppervlakteRechthoek = basisRechthoek * hoogteRechthoek;  
console.log(oppervlakteRechthoek); // 6  
basisRechthoek = 6;  
console.log(oppervlakteRechthoek); // 12
```

De incrementele operator ++

- ++ noemen we de incrementele operator
- Deze verhoogt de variabele die eraan voorafgaat met 1
- De decrementele operator -- vermindert het getal met 1
- Een getal verhogen of verlagen is eenvoudig op deze manier

```
let leeftijd = 45;  
leeftijd++;  
console.log(leeftijd); // 46  
leeftijd--;  
leeftijd--;  
console.log(leeftijd); // 44
```

De toekennings- of assignment- operator

- We zagen al = bij het toekennen van waardes aan onze variabelen
- Bepaalde assignment operatoren kunnen ook de waardes doen veranderen
- += voegt een getal toe, -= trekt een getal af
- *= en /= werken ook

```
let leeftijd = 45;  
leeftijd -= 11; // is een shortcut voor leeftijd = leeftijd - 10  
console.log(leeftijd); // 34  
leeftijd += 4;  
leeftijd /= 2;  
console.log(leeftijd); // 19
```

Strings aaneenschakelen

```
let a = "Vandaag ga ik ";  
let b = "genieten van het mooie weer";  
let c = a + b;  
console.log(c); // Vandaag ga ik genieten van het mooie weer
```

Booleans: true of false

Booleaanse operatoren of booleans

- Booleans is een belangrijk datatype
- Heeft te maken met vergelijkingen tussen waarden en zal steeds uitkomen op waar of niet waar
- Om waarden te vergelijken gebruiken we == of === (niet te verwarren met toekenningsoperator =)
- Als de waarden gelijk zijn geeft de vergelijking true terug, indien niet geeft ze false terug

```
let gelijkheid1 = 3 == 3;  
let gelijkheid2 = "Banaan" == "Sinaasappel";  
console.log(gelijkheid1); // true  
console.log(gelijkheid2); // false
```

Strikte gelijkwaardigheid

- Verschil tussen == en === ?
- === staat voor strikte gelijkwaardigheid
- Strikte gelijkwaardigheid wil zeggen: waarde gelijk EN datatype gelijk
- Je zal mij vooral de strikte gelijkwaardigheid zien gebruiken

```
let gelijkheid1 = 3 == "3";  
let gelijkheid2 = 3 === "3";  
console.log(gelijkheid1); // true  
console.log(gelijkheid2); // false
```

Kleiner dan, groter dan

- We kunnen ook testen of een bepaalde waarde kleiner dan, groter dan, kleiner dan of gelijk aan (\leq), groter dan of gelijk aan (\geq) is
- Ook het resultaat van deze vergelijkingen is een boolean

```
let waarde1 = 3 > 6;  
let waarde2 = 3 < 6;  
console.log(waarde1); // false  
console.log(waarde2); // true
```

Verschillend van...

- Het != of !== staat voor "niet gelijk aan" of "niet strikt gelijk aan"
- Als de waarden verschillend zijn dan wordt true teruggegeven
- Als de waarden hetzelfde zijn dan wordt false teruggegeven

```
let waarde = 3 != 1;  
console.log(waarde); // true  
waarde = 5 != "5";  
console.log(waarde); // false  
waarde = 5 !== "5";  
console.log(waarde); // true
```

De logische en (&&)

- Als we willen dat meerdere voorwaarden waar zijn vooraleer we een waar kunnen teruggeven dan gebruiken we de logische en operator
- De logische en gebruikt een dubbele ampersand als symbool
- Elke voorwaarde moet waar zijn om als geheel in waar te resulteren

```
let waarde = 5 < 10 && 10 < 15;  
console.log(waarde); // true  
waarde = 5 < 10 && 10 < 15 && 10 < 8;  
console.log(waarde); // false
```

De logische of (||)

- Als het enkel noodzakelijk is dat één van twee vergelijkingen waar is, dan kunnen we de logische of operator gebruiken
- Van zodra één van beiden waar is geeft de operator waar terug
- Pas indien er geen enkele waar is krijgen we false
- Je kan zoveel vergelijkingen als je wil in de logische en/of operator gebruiken

```
let waarde = 5 > 10 || 10 > 15;  
console.log(waarde); // false  
waarde = 5 < 10 || 10 > 15;  
console.log(waarde); // true  
console.log(true || false || false) // true
```

De logische not (!)

- De ! operator verandert de waarde van een boolean in zijn tegengestelde: true wordt false en omgekeerd: false wordt true

```
let waarde = true;  
let tegengestelde = !waarde;  
console.log(tegengestelde); // false
```

De voorwaardelijke operator (ternary operator)

- De voorwaardelijke operator kent een waarde toe aan een variabele gebaseerd op het true of false zijn van een voorwaarde
- De vorm: (voorwaarde) ? waarde indien true : waarde indien false

```
let uurVanDeDag = 7;  
let slaapAdvies = uurVanDeDag >= 11 ? "Ga best slapen" : "Blijf gerust nog wakker";  
console.log(slaapAdvies); // "Blijf gerust nog wakker"
```


Strings vergelijken

- Strings met elkaar vergelijken met `===` en `!==` is perfect zinvol
- Als je gaat gebruik maken van andere operatoren (bvb `<`) zijn de resultaten minder zinvol en voorspelbaar
- In het geval van kleiner dan gaat JavaScript bvb een alfabetische volgorde hanteren. We gebruiken het beter niet

```
let waarde = "4" > "33";  
console.log(waarde); // true  
let waarde = "aap" > "noot"; // false
```

Voorwaardelijke statements

if-statement

- We zijn nu vertrouwd met variabelen en wiskundige operaties. tijd om ze écht aan het werk te zetten...
- Vaak zal je variabelen testen: hoe groot/klein zijn ze? Hoe verhouden ze zich tot een andere variabele?
- Met de "if"-constructie ga je voorwaarden stellen. De uitkomst zal dan waar zijn of niet waar
- Enkel indien de uitkomst waar is worden de statements die volgen uitgevoerd
- De if-constructie ziet er uit als volgt:

```
let temperatuur = 11;  
if (temperatuur > 25) {  
    console.log("Wat is het warm vandaag!");  
}
```

if-else-statement

- Indien we ook code willen uitvoeren als de uitkomst niet waar is, dan kunnen we gebruik maken van de if-else-constructie

```
let temperatuur = 23;
if (temperatuur > 25) {
    console.log("Wat is het warm vandaag!");
    // hier kunnen meerdere statements staan...
} else {
    console.log("Normaal weer vandaag");
    // Hier ook mogelijk nog statements...
}
```

if-else if-else-statement

- En indien dit nog niet genoeg is...
- Met if-else if-else kunnen we meerdere voorwaarden creëren

```
let temperatuur = 20;  
if (temperatuur > 25) {  
    console.log("Wat is het warm vandaag!");  
} else if (temperatuur < 0) {  
    console.log("Brr, het is koud vandaag!");  
} else {  
    console.log("Normaal weer vandaag");  
}
```

if-else if-else-statement

- Je kan hier gebruik maken van zoveel else if-voorwaarden als je nodig hebt
- Deze constructie moet wel starten met "if" en eindigen met "else". Tussenin gebruik je zoveel "else if" constructies als je nodig hebt

```
let naam = "John";  
if (naam === "Ringo") {  
    console.log("Goeie dag Ringo!");  
} else if (naam === "George") {  
    console.log("Alles goed George?");  
} else if (naam === "Paul") {  
    console.log("Hallo Paul!");  
} else {  
    console.log("Hey John!");  
}
```

Nesten van if-statements

- If-statements kunnen ook binnen andere if-statements gestopt worden
- We noemen dit "nesten"
- In onderstaand voorbeeld krijgen we niets in de console te zien. Het getal is immers kleiner dan 0
- Pas mocht het wel groter dan 0 geweest zijn werd het getest of het kleiner was dan 10

```
let getal = -5;
if (getal >= 0) {
  if (getal < 10) {
    console.log("Het is een cijfer!");
  } else {
    console.log("Het is 10 of groter en dus geen cijfer")
  }
}
```

Eenvoudiger testen met logische operatoren

- Zoals hierboven reeds aangegeven: logische en
- && gebruik je als de eerste als de tweede voorwaarde beiden waar moeten zijn voordat de statement wordt uitgevoerd

```
let getal = 5;  
if (getal >= 0 && getal < 10) {  
  console.log("Het is een cijfer");  
} else {  
  console.log("Het is geen cijfer");  
}
```


Eenvoudiger testen met logische operatoren

- Zoals hierboven reeds aangegeven: logische of
- `||` gebruik je als slechts één van beide voorwaarden waar moet zijn voordat de statement wordt uitgevoerd

```
let dier = "hond";  
if (dier === "hond" || dier === "kat") {  
  console.log("Het dier is een hond of een kat");  
}
```

Het switch-statement

```
let naam = "Ringo";
switch (naam) {
case "John":
    console.log("John Lennon");
    break;
case "Ringo":
    console.log("Ringo Starr");
    break;
case "Paul":
    console.log("Paul McCartney");
    break;
default:
    console.log("George Harrison");
}
```

Het switch-statement

- Soms heb je veel voorwaarden. In dat geval is het zinvol gebruik te maken van het switch-statement
- Met elke case ga je testen op een mogelijke match. Op het einde van elke case volgt een break statement
- Op het einde hebben we een default (vergelijkbaar met de else). Hier hoeft geen break toegevoegd worden

Voorwaardes die niet resulteren in een boolean?

- Ook deze voorwaarden worden automatisch omgezet in een boolean
- Als er een reële waarde in zit dan wordt de voorwaarde automatisch waar
- Ook de vaak gemaakte fout `if (waarde = "Batman")` ipv `if (waarde === "Batman")` leidt hiertoe...

```
let waarde = "Batman";  
if (waarde) {  
    console.log("Een reële waarde zal dit resultaat weergeven");  
}
```

Webtech - JS - les10 - kristof.michiels01@ap.be