



JavaScript

**Webtechnologie
Essentials**

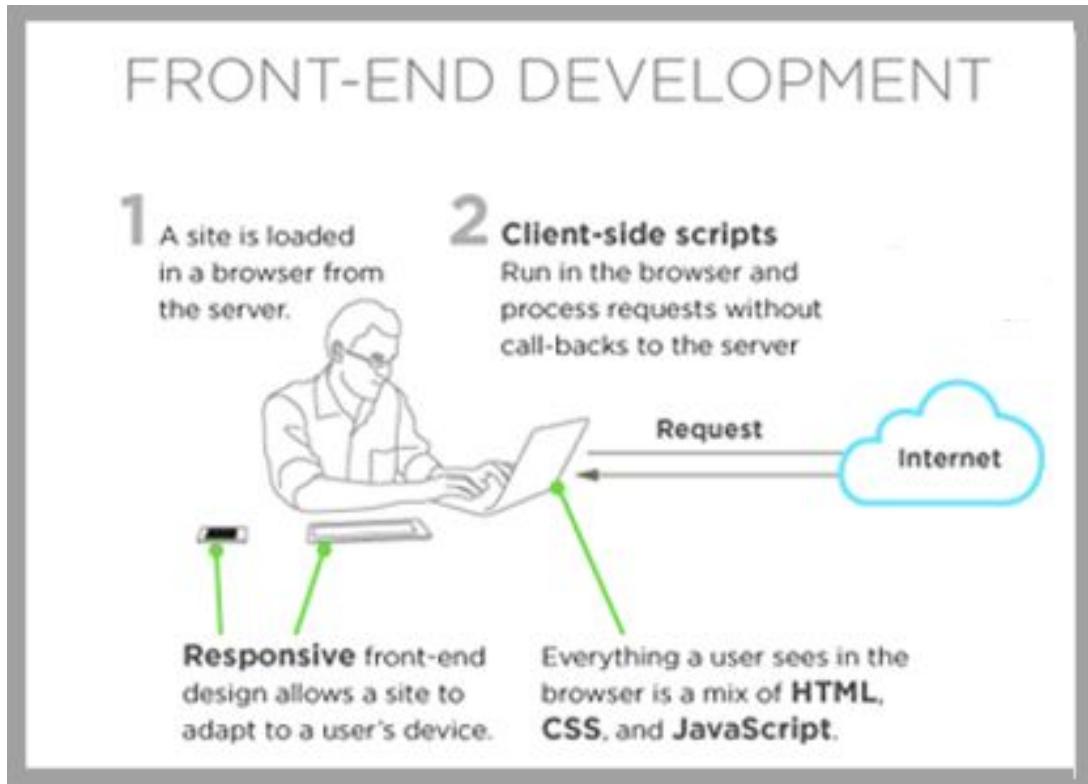
**DE HOGESCHOOL
MET HET NETWERK**

Hogeschool PXL – Elfde-Liniestraat 24 – B-3500 Hasselt
www.pxl.be - www.pxl.be/facebook



JavaScript

- Scriptingtaal: programma's hoeven niet gecompileerd te worden. De browser vertolkt (interpreter)



JavaScript

- Voordeel:
 - De taal zelf is ECMA gestandaardiseerd
 - Snel testen of een programma werkt: gewoon openen in een browser volstaat
 - Niet beperkt tot browsers of internettoepassingen alleen (ook backend: NodeJS)
- Niet gerelateerd aan de taal Java
 - Verschillend qua historiek en bedoeling

Eerste voorbeeld

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Demo 1: first javascript</title>
</head>
<body>
<output id="demo"></output>
<script>
    document.getElementById("demo").innerHTML
        = "Hello JavaScript!";
</script>
</body>
</html>
```

Voorbeeld: <https://jsfiddle.net/NiekVandael/auxa6vku/>

JavaScript in extern bestand

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Demo 2: externe bestanden</title>
    <script src="info.js" async></script>
    <script src="berekeningen.js" defer></script>
</head>
<body>
    ...

```

Strict-mode

- Strict mode: code wordt “nauwkeurig” en “strikt” uitgevoerd.
- Iets minder mogelijkheden, minder problemen
- Sneller
- Zichtbare errors i.p.v. stille mislukking.
- Opt-in:
"use strict";
- Aanrader!

<https://jsfiddle.net/NiekVandael/auxa6vku/6/>

Variabelen

- Declareren en initialiseren:

```
let getal1;  
getal1 = 5;  
let getal2 = 10;  
let getal3, getal4 = 7, getal5;
```

Variabelen

- Naamgeving:

```
let hoogte = 5;
```

```
let eersteGetal = 7;
```

```
let tweedeGetal = 8; // enkel als goed leesbaar
```

```
let zin = "dit is een string";
```

- Afgeraden:

```
let derde_getal = 5;
```

```
let $tekst = "hallo";
```

```
let _naam = "Jan"; (veel gebruikt als private variabelen in libraries)
```

```
let VALVERSNELING = 9.81; // niet een constante
```

Types van variabelen

- *Number:*

- Decimale of hexadecimaal, positief of negatief

```
var decGetal = 10; // decimale waarde is 10
```

- **Punt**, geen komma.

```
var floatGetal = 3.5; // decimale waarde is 3,5
```

Types van variabelen

- *Number:*
 - 64-bit double precision floating point
 - Geen byte, short, int, long of float

```
var x = 3;  
x = x / 2;           // x = 1.5
```

Types van variabelen

- *Number:*
 - Speciale waarden:
 - 1 - 1: 0
 - 1 * 0: -0
 - 1 / 0: Infinity
 - 1 / 0: -Infinity
 - 0 / 0: NaN

Werken met getallen

- De functie Number()
 - String omzetten naar een getal

```
var getal = Number(window.prompt("Geef een getal in: "));
```
- De functie isNaN()
 - Testen op de waarde NaN, geeft true of false terug

```
if (isNaN(getal))
```
- De functie parseInt(String)
 - String omzetten naar een **geheel** getal

```
var afrond = parseInt(getal + 0.5);  
console.log("Getal " + getal + " is afgerond " + afrond);
```
- De functie parseFloat(String)
 - String omzetten naar een reëel getal

Types van variabelen

- *String:*
 - Een reeks van UTF-16-karakters*
 - Enkele aanhalingstekens:

```
var welkomTekst = 'Hallo, wereld';
```

```
var talen = '中文 español English हिन्दी + العربية +  
'português বাংলা русский 日本語 '+  
'ਪੰਜਾਬੀ 한국어 தமிழ்';
```

* Eigenlijk UCS-2 met surrogaten. Info: <https://mathiasbynens.be/notes/javascript-encoding>

Types van variabelen

- *String:*

- Speciale karakters:

'\' '\\'

'\n' '\r'

'\uXXXX' '\0'

...

- Enkele quotes en backtick symbool:

```
var welkomTekst = 'Hallo, Wereld';
```

```
var andereTekst = `Ik zeg: ${welkomTekst}`;
```

Werken met strings

- Aaneenschakeling van karakters

```
let a = 'een testje met een string';
let b = "een andere string";
let c = "";    // de lege string
let d = "Deze string kan een 'quote' bevatten";
let e = 'Deze string kan "dubbele quotes" bevatten';
let f = "Quotes kan ook door \"escapes\" toe te passen";
let g = "Eerste lijn \nTweede lijn";
let h = new String("Nog een string");
```

Voorbeeld: <https://jsfiddle.net/NiekVandael/0395g2k7/3/>

Bewerkingen op strings

- Samenvoegen van strings (concatenatie)

```
let tekst1 = 'Hogeschool PXL';
let tekst2 = 'Ik studeer aan';
let alles = tekst2 + ' ' + tekst1;
```

- Lengte van string opvragen

```
let tekst = 'Hogeschool PXL';
let lengte = tekst.length;           // lengte = 14
```

- Karakter op bepaalde positie opvragen

```
let tekst = 'Hogeschool PXL';
let letter = tekst.charAt(0);        // letter = "H"
```

Bewerkingen op strings

- Karakter(s) zoeken in string

```
let tekst = 'Piet@myComp.be';
let i = tekst.indexOf('@');           // i = 4
let j = tekst.indexOf('.nl');         // j = -1
```

- Aantal karakters uit string halen

```
let tekst = 'PXL Hogeschool Limburg';
let s1 = tekst.substring(0, 3);        // s1 = "PXL"
let s2 = tekst.substring(4);          // s2 = "Hogeschool Limburg"
```

- Converteren van hoofdletters naar kleine letters en omgekeerd

```
let tekst = 'Hogeschool PXL';
let hoofdLetters = tekst.toUpperCase(); //naar hoofdletters
let kleineLetters = tekst.toLowerCase(); //naar kleine letters
```

Types van variabelen

- *Boolean:*
 - let groterDanHelft = true;
 - let kleinerDanHelft = false;

Types van variabelen

- *Undefined*
 - De null uit Java/C#/...
 - Ongedefinieerde variabele:
`let geenWaarde;`
 - Kan ingesteld worden:
`let test = undefined;`
- *Null*
 - Verwarrend! Niet hetzelfde als null in Java/C#/...
 - ≈ onbekend, **leeg**
 - Sommige DOM functies geven null i.p.v. undefined

Operatoren

- Rekenkundige

operator	bewerking	voorbeeld	commentaar
+	optellen	$i = j + 3;$	
-	aftrekken	$i = j - 3;$	
*	vermenigvuldigen	$i = j * 3;$	
/	delen	$i = j / 3;$	
%	modulus	$i = j \% 3;$	rest van de deling

Operatoren

- Rekenkundige, afgeraden buiten for-lus

operator	bewerking	voorbeeld	Commentaar
++	postfix increment	res = i++;	Verhoogt i met 1, geeft oude waarde terug
++	prefix increment	res = ++i;	Verhoogt i met 1, geeft nieuwe waarde terug
--	postfix decrement	res = i--;	Verlaagt i met 1, geeft oude waarde terug
--	prefix decrement	res = --i;	Verlaagt i met 1, geeft nieuwe waarde terug

Operatoren

- Toekenning

operator	bewerking	voorbeeld	commentaar
=	toekennen	$i = j$	
+=	optellen met	$i += j$	idem als $i = i + j$
-=	aftrekken met	$i -= j$	idem als $i = i - j$
*=	vermenigvuldigen met	$i *= j$	idem als $i = i * j$
/=	delen met	$i /= j$	idem als $i = i / j$
%=	modulus met	$i \%= j$	idem als $i = i \% j$

- Tekst

operator	bewerking	voorbeeld	commentaar
=	toekenning	woord = woord2	
+	concatenatie	tekst = woord1 + woord2	
+=	concatenatie	tekst += woord3	tekst = tekst + woord3

Zwakke typering

- Type niet opgeven: let
- Afgeleid uit waarde
- Voorbeeld:

```
let counter = 0;    // counter is een Number  
counter = "test"; // counter is een String
```

- Het type kan wijzigen!

Soorten typering

- Zwak getypeerd: geen type aangeven

```
let i = 10;           // i is Number: 10  
i = "Getal " + i + i; // i is String: "Getal 1010"  
i = 10;             // i is Number: 10  
i = "Getal " + (i + i); // i is String: "Getal 20"
```

- Sterk getypeerd: voorbeeld in C#

```
int i = 10; // type = System.Int32  
i = i * 5; // enkel als int te gebruiken
```

Conversie

- Automatische conversie op basis van de context
- Number -> String
 - tekst = " + getal;
 - tekst += getal;
 - getal += tekst;
 - tekst = getal.toString(10);
- String -> Number
 - getal = 1 * tekst;
 - getal = +tekst;
 - getal = parseInt(tekst, 10);

Bereik van een variabele

- Globale variabelen:
 - Declaratie buiten alle functies
 - Overal toegankelijk.
 - Slecht idee, zeker bij grotere sites.
- Lokale variabelen:
 - Binnen een functies gedeclareerd;
 - Enkel beschikbaar binnen deze functie.

```
let tekst1 = "JavaScript"; // globale variabele
(function () {
    let tekst2 = "is een zwak getypeerde taal"; // lokale variabele
    console.log(tekst1 + " " + tekst2);
})();
```

Const en Let

```
const MINUTES_IN_HOUR = 60;
```

```
MINUTES_IN_HOUR = 40;
```

//Uncaught TypeError: Assignment to constant variable.

```
for (let teller = 0; teller <= 5; teller++) {  
    resultaat += teller + "<br>";  
}
```

Zie demo

Arrays

- Rijen van waardes

```
let leeg = [];                                // grootte 0
let weekDag = new Array(7);                   // grootte 7
let cijfers = [0,1,2,3,4,5,6,7,8,9];        // grootte 10
let getal = 5;
let tekst = 'hallo';
let allerlei = [getal, tekst, cijfers, leeg, 8, true];
                                         // grootte 6
```

Arrays

- Rijen van rijen:

```
let wall, space, player, enemy, goal;
```

```
...
```

```
var level = [  
    [wall,  wall,  wall,  wall,  wall,  wall,  wall, wall],  
    [wall, space, space,  space, space, space, goal, wall],  
    [wall, space, space, enemy, space, space, space, wall],  
    [wall, player, space,  space, space, space, space, wall],  
    [wall,  wall,  wall,  wall,  wall,  wall,  wall, wall]  
];
```

Arrays

- Aantal elementen

```
let aantal = maand.length;
```

- Indexeren van array: []

```
let elem1 = cijfers[0]; // eerste element
```

```
let elem2 = cijfers[1]; // tweede element
```

```
cijfers[2] = 'Drie'; // derde element toekennen
```

```
maand[maand.length - 1] = 'december';  
// laatste element
```

```
level[3][1] = player;
```

Arrays

- Te grote indexen gebruiken geeft geen fout!

```
maand[12] = "Leve de dertiende maand!";
// nu 13 elementen
```

```
reeks[reeks.length] = "een extra element";
reeks[reeks.length] = "nog één";
```

- Opgelet:

```
maand[20000] = "De super maand";
console.log(maand.length); // 20001
console.log(maand[16534]); // undefined
```

Vergelijkende operatoren

operator	bewerking	voorbeeld
==	is gelijk aan	i == 5
!=	is niet gelijk aan	i != 5
>	is groter dan	i > 5
<	is kleiner dan	i < 5
>=	is groter of gelijk aan	i >= 5
<=	is kleiner of gelijk aan	i <= 5

Operatoren

- Vergelijkend, **sterk afgeraden!**

operator	bewerking	voorbeeld
==	is ietwat gelijk aan	i == 5
!=	is ietwat niet gelijk aan	i != 5

- Vergelijkt zonder het type van de variabele.

```
"\n" == 0      // true  
5  != "5"     // false  
[]  == false   // true
```

```
'.' == '0'    // false  
0  == '.'     // true  
0  == '0'     // true
```

Logische operatoren

operator	bewerking	voorbeeld	commentaar
&&	and	i === 0 && j < 5	Eerste waarde als false, anders tweede waarde
	or	i === 0 i >= 100	Eerste waarde als true, anders tweede waarde
!	not	!(i === j)	Een negatie (i !== j kan ook)
? :	inline if	i >= j ? "i" : "j"	Test met true- en false-waarde

Conversie

- Naar Boolean: truthy/falsy
- Falsy:
 - false
 - 0
 - ""
 - null
 - undefined
 - NaN
- Truthy:
 - alle andere waarden

Conversie

- Naar Boolean: truthy/falsy

```
!true      // false
```

```
!!true     // true
```

```
!!0        // false
```

```
!![5, 3]   // true
```

```
!!undefined // false
```

Operatoren

||: geeft default waarde

Situatie A

```
let reeks;  
reeks = [4, 5];  
reeks = reeks || [1, 2];
```

Resultaat: reeks is [4, 5]

Situatie B

```
let reeks;  
// niks  
reeks = reeks || [1, 2];
```

Resultaat: reeks is [1, 2]

<https://www.w3schools.com/code/tryit.asp?filename=FQMLWU6ES4MS>

TypeError

- Error als het type fout is of niet kan geconverteerd worden.

- Typisch:

```
let mijnData; // leeg!
```

```
let waarde = mijnData[3]; // TypeError,  
                           // mijnData is undefined
```

- Vergelijk met NullPointerException (Java),
NullReferenceException (C#)

Operatoren

&&: beschermt tegen TypeError

Situatie A

```
let reeks;  
reeks = [1, 2];  
let getal = reeks && reeks[1];
```

Resultaat: getal === 2

Situatie B

```
let reeks;  
// niks  
let getal = reeks && reeks[1];
```

Resultaat: getal === undefined
Geen TypeError

Functies

```
<script>
  "use strict";
  document.getElementById("square").innerText = square(4);
  // bovenstaande regel geeft: "Uncaught TypeError: square is not a function"
  document.getElementById("cube").innerText = cube(4);

  var square = function(x) {
    return x * x;
  };

  function cube(x) {
    return x * x * x;
  }

  document.getElementById("square").innerText = square(5);
  document.getElementById("cube").innerText = cube(5);
</script>
```

<https://www.w3schools.com/code/tryit.asp?filename=FQMLMWXMPQI9>

Functies

```
var landscape = function () {  
    var result = "";  
    var flat = function (size) {  
        for (let count = 0; count < size; count++)  
            result += "_";  
    };  
    var mountain = function (size) {  
        result += "/";  
        for (let count = 0; count < size; count++)  
            result += "";  
        result += "\\";  
    };  
  
    flat(3); mountain(4); flat(6); mountain(1); flat(1);  
    return result;  
};
```

<https://www.w3schools.com/code/tryit.asp?filename=FQMLSICMEHWI>

Functies

- Return waarde array:

```
let keerOm = function (a, b, c) {  
    return [c, b, a];  
};
```

- Uitvoeren:

```
let reeks = keerOm(1, 2, 3);  
let laatst = keerOm(1, 2, 3)[2];  
console.log(keerOm(1, 2, 3)[2]);
```

Controlestructuren

- if-else

```
if (voorwaarde) {  
    statements;  
} else {  
    statements;  
}
```

- Voorbeeld:

```
if (voornaam === null) {  
    alert("Geef je voornaam op");  
}
```

Controlestructuren

- if-else

```
if (voorwaarde) {  
    statements;  
} else if (voorwaarde) {  
    statements;  
} else if (voorwaarde) {  
    statements;  
} else {  
    statements;  
}
```

Controlestructuren

- **switch**

```
switch(n) {  
    case 1:      //voer code uit horende bij n == 1  
        statements1;  
        break;      //spring uit het switch statement  
    case 2:      //voer code uit horende bij n == 2  
        statements2;  
        break;      //spring uit het switch statement  
    case 3:      //voer code uit horende bij n == 3  
        statements3;  
        break;      //spring uit het switch statement  
    default:     //voer code uit horende bij alle andere gevallen  
        statementsN;  
        break;      //spring uit het switch statement  
}
```

Controlestructuren

- **while**

```
while (voorwaarde) {  
    statement(s);  
}
```

```
function printEven() {  
    let tekst = "Even getallen: ";  
    let i = 0;  
    while (i < 10) {  
        if (i % 2 === 0) {  
            tekst += i + " ";  
        }  
        i++;  
    }  
    alert(tekst);  
}
```

Controlestructuren

- **for**

```
for (initialisatie; voorwaarde; stappen) {  
    statement(s);  
}
```

- Voorbeeld

```
function printEven() {  
    let tekst = "Even getallen: ";  
    for (let i = 0; i < 10; i++) {  
        if (i % 2 === 0) {  
            tekst += i + " ";  
        }  
    }  
    alert(tekst);  
}
```

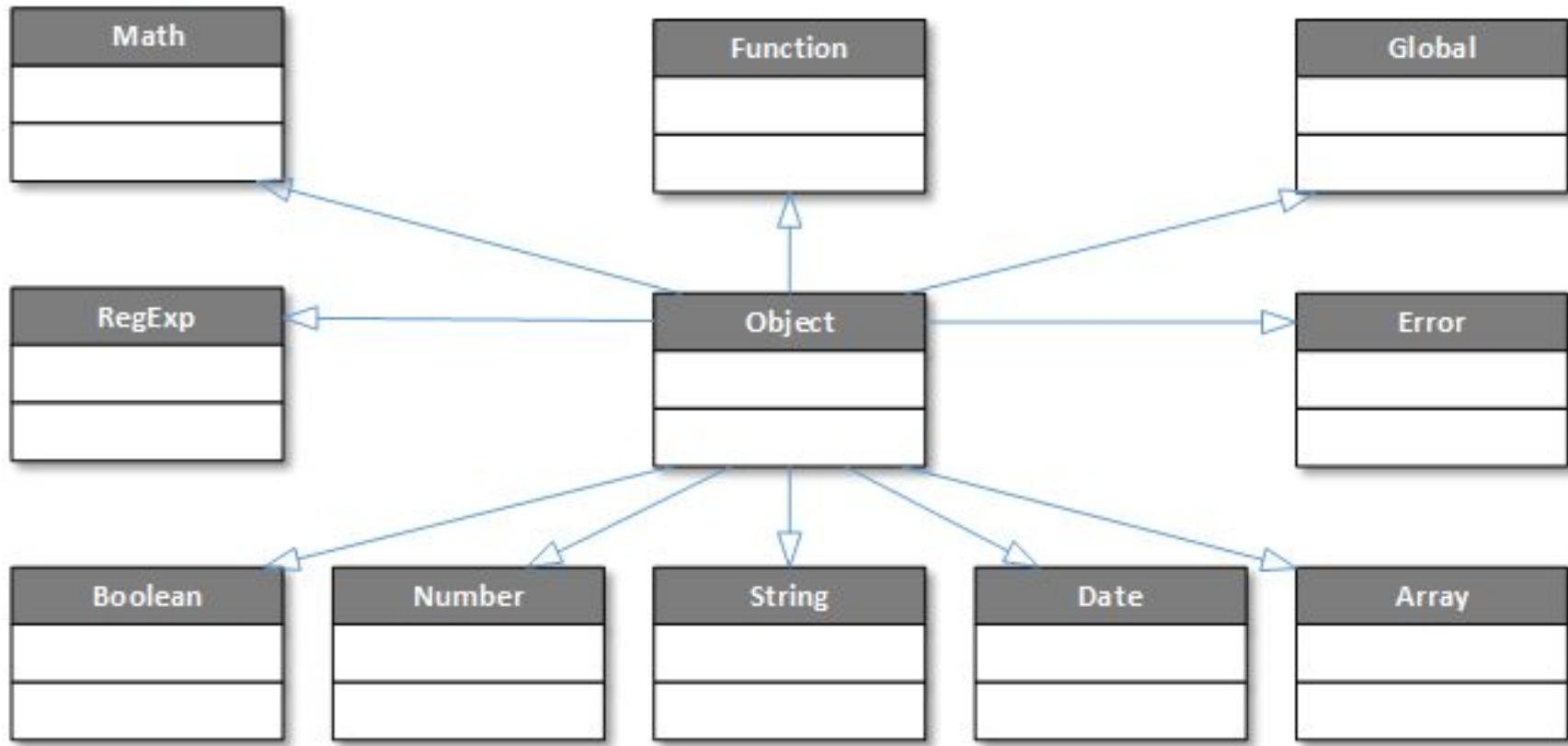
Objecten

- Een object heeft properties of eigenschappen die je kunt lezen of schrijven.
- Een object heeft methodes die je kan oproepen en die de toestand van dat object kunnen wijzigen.
`object.property`
`object.method()`

Objecthiërarchie

- Een String, een Array, een Window, een Frame, een Form, functies, ...
Alles is een object.
- Illustratie: 13,0 is een object.
 - 13.0.toString(2)
// 1101

Objecthiërarchie



Objecten maken

- Objecten maken:

```
var vandaag = Date.now();
```

```
var kerstmis = new Date(2016, 11, 25);
```

- Meestal beschikbare objecten gebruiken.

```
bv. document.getElementById("output")
```

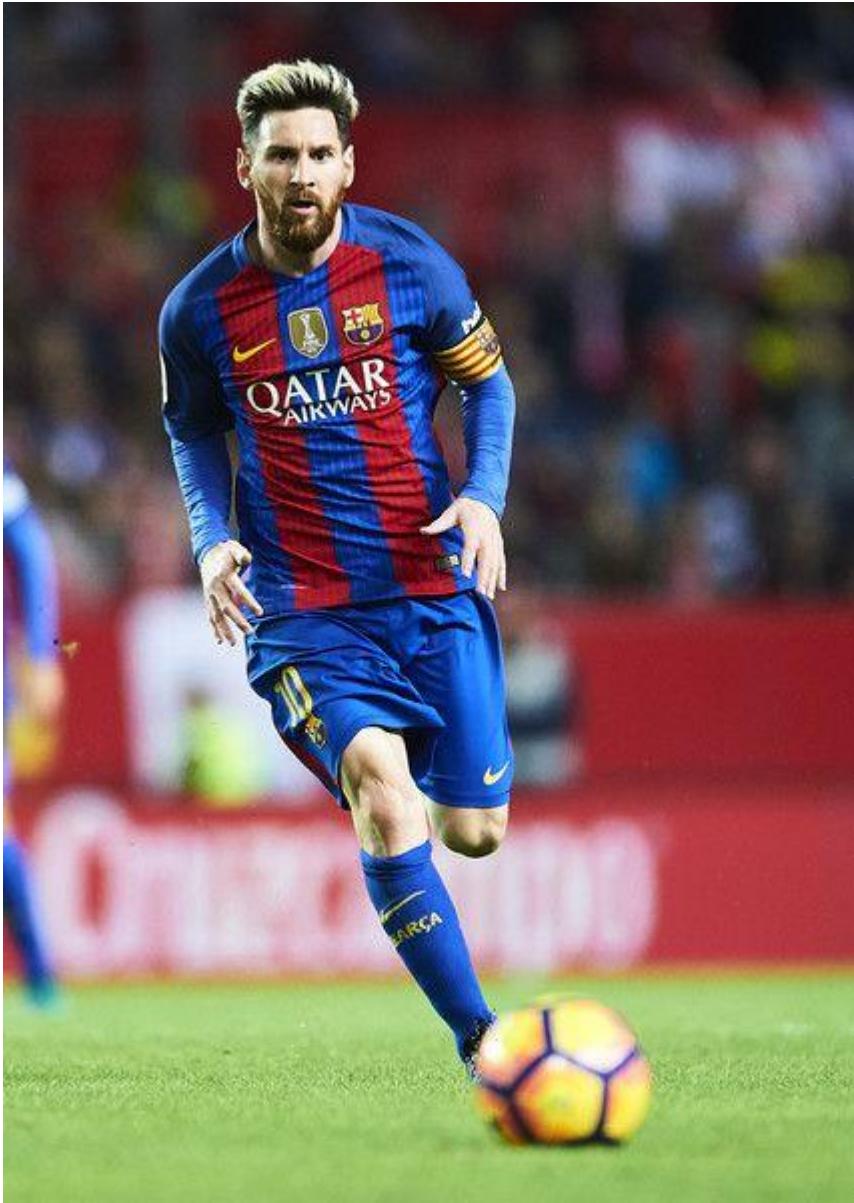
ES in WebStorm

The screenshot shows the WebStorm settings interface. The left sidebar has a search bar at the top with "JavaScript". Below it is a tree view of settings categories:

- Editor**
 - Code style
 - JavaScript
 - Inspections
 - Live Templates
 - Copyright**
 - Formatting
 - JavaScript
 - JavaScript
 - JavaScript
 - Intentions
- Plugins**
- Build, Execution, Deployment**
 - Debugger
 - Data Views
 - Stepping
- Languages & Frameworks**
 - JavaScript**
 - Libraries
 - Code Quality Tools

JavaScript

OBJECT LITERALS



Eigenschappen

voetballer.voornaam = "Lionel"

voetballer.naam = "Messi"

voetballer.geboortedatum = ""

voetballer.lengte = 170

voetballer.been = "Links"

Object literal

A JavaScript object literal is a comma-separated list of name-value pairs wrapped in curly braces. Object literals encapsulate data.

```
let voetballer = {  
    voornaam: 'Lionel',  
    naam: 'Messi',  
    lengte: 170,  
    geboortedatum: new Date(1987, 5, 24),  
    been: 'Links'  
};
```

Object literal – toegang properties

- *objectName.propertyName*
 - bijv. *voetballer.naam*
- *objectName["propertyName"]*
 - bijv. *voetballer["lengte"]*

```
let voetballer = {  
    voornaam: 'Lionel',  
    naam: 'Messi',  
    lengte: 170,  
    geboortedatum: new Date(1987, 5, 24),  
    been: 'Links'  
};
```

