

EcmaScript 201X

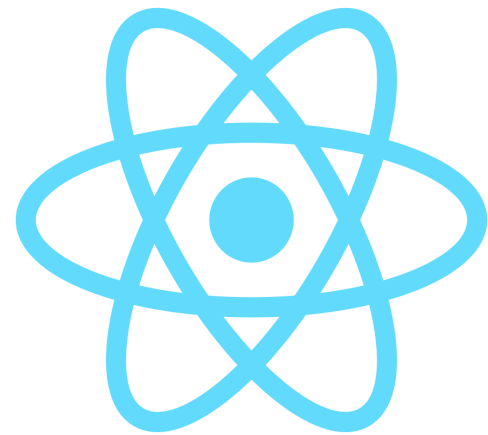


Training: rick@oblicum.com or [@rickbeerendonk](https://twitter.com/rickbeerendonk)



- **ECMAScript**

- 5, 2015, 2016, 2017, 2018...



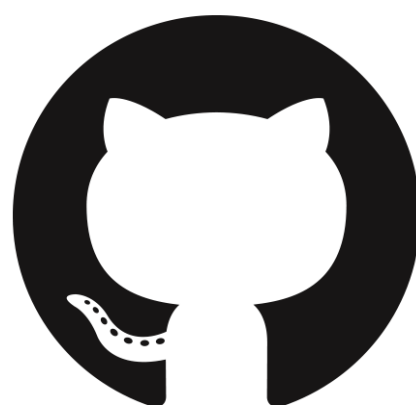
- **React**

- Components, Properties, State, Events, Virtual DOM...



- **Redux**

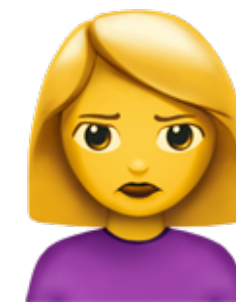
- Actions, Reducers, Stores...



- **Samples & Slides**

- <https://github.com/rickbeerendonk/ECMAScript-examples>

JavaScript 201X Quiz



1. The following code is...

```
var x = 10;
```



C#



JavaScript

1. The following code is...

```
var x = 10;
```



C# 🎉

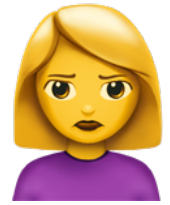


JavaScript 🎉

2. C#'s foreach in JavaScript is...





for .. in



for .. of

2. C#'s foreach in JavaScript is...

 for .. in

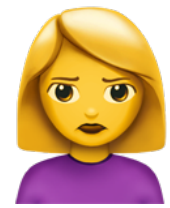
 for .. of 

3. Indeterminate Number of Parameters in JavaScript

```
C#: void Test(params int[] a)
```



```
function test([]a)
```



```
function test(...a)
```


3. Indeterminate Number of Parameters in JavaScript

```
C#: void Test(params int[] a)
```



```
function test([]a)
```



```
function test(...a)
```



4. When does calling this function throw an error?

```
function test(a, b)
```

1 finger 🙋 test(1)

2 fingers 🙋 test(1, 2)

3 fingers 🙋 test(1, 2, 3)

0 fingers 🙄 *<never>*

4. When does calling this function throw an error?

```
function test(a, b)
```

1 finger 🙋 test(1)

2 fingers 🙋 test(1, 2)

3 fingers 🙋 test(1, 2, 3)

0 fingers 🙄 <never> 🎉

5. Call constructor of the parent class

```
class Child extends Parent {  
    constructor(name, value) {  
        <??>  
        this.balance = value;  
    }  
}
```



base(name)



super(name)

5. Call constructor of the parent class

```
class Child extends Parent {  
    constructor(name, value) {  
        <??>  
        this.balance = value;  
    }  
}
```



`base(name)`



`super(name)`



ECMAScript

- 2009: 5th Edition
- 2015: 6th Edition
- Changed to:
 - ▶ Yearly releases (in June)
 - ▶ Year = version number

Ecma International, Technical Committee 39 (TC39)

- [Proposal process](#)
- [Finished proposals](#)
- [Active proposals](#)

Primitive Data Types

- String
- Number
- Bool
- Undefined
- Null

Primitive Data Types

- String
 - ▶ Single (") or Double Quotes (""")
 - ▶ C#: Char & String
- Number
 - ▶ C#: Double & Int
- Bool
- Undefined
- Null

Variable declarations

```
var a = 1;

if (true) {
    var a = 2;
    console.log(a); // 2
}

console.log(a); // 2
```

```
let a = 1;

if (true) {
    let a = 2;
    console.log(a); // 2
}

console.log(a); // 1
```

C# var scoping = JS let scoping

Constants

```
// changeable
var a = 1;

if (true) {
    var a = 2;
    console.log(a); // 2
}

console.log(a); // 2
```

```
// unchangeable
const a = 1;

if (true) {
    const a = 2;
    console.log(a); // 2
}

console.log(a); // 1
```

Same as C#

Template Strings C#

```
const name = "EcmaScript";  
const version = 2015;  
Func<string> x = () => "hi!";  
  
var result = $"This is about:  
{name} {version + 1} {x()}";  
  
Console.WriteLine(result);  
// This is about:  
// EcmaScript 2016 hi!
```

Template Strings JavaScript

```
const name = 'EcmaScript';  
const version = 2015;  
const x = () => 'hi!';  
  
var result = `This is about:  
${name} ${version + 1} ${x()}`;   
  
console.log(result);  
// This is about:  
// EcmaScript 2016 hi!
```

C# `$"{}"` = JS ``${}``

Equality: == vs ===

```
const i = 1;  
const s = '1';  
  
console.log(i == s);  
// true  
// (value)
```

```
const i = 1;  
const s = '1';  
  
console.log(i === s);  
// false  
// (value + type)
```

C# == is the same as JS ===

Conditional Statements

- If

- ▶ `if (true || false) {
 console.log('positive');
} else {
 console.log('negative');
}`

- Inline

- ▶ `console.log(true || false ? 'positive' : 'negative');`

Same as C#

Loops

- for
 - ▶ `for (let i = 0; i < 2; i++) { console.log(i)}`
- forEach
 - ▶ `[1, 2, 3].forEach((element, index, array) => console.log(`a[${index}] = ${element}`))`
- for .. in
 - ▶ Iterates over object properties
- for .. of
 - ▶ Iterates over iterable object (Array, String, Map, Set, etc.)
- for await .. of (ES 2018)
 - ▶ Iterates over async iterable object (returning array of promises)

C# for = JS for

C# foreach = JS for .. of

Generators / Iterators

```
const test = {  
  [Symbol.iterator]: function*() {  
    let current = 1;  
    while (true) {  
      yield current++;  
    }  
  }  
}
```

```
for (let n of test) {  
  console.log(n);  
  if (n >= 10) {  
    break;  
  }  
}
```

C# IEnumerable = JS Iterator

C# function + yield + foreach = JS function*
+ yield + for .. of

Functions: Overloads

```
function test(a, b) {  
  console.log(a);  
  console.log(b);  
}
```

```
test(1); // a = 1, b = undefined
```

```
test(1, 2, 3, 4); // a = 1, b = 2, 3 & 4 = ignored
```

C# overload = JS one function

Functions: Default parameters

```
function test(a = 11, b = '22') {  
    console.log(a);  
    console.log(b);  
}
```

```
test(); // a = 11, b = '22'
```

```
test(1, 2, 3, 4); // a = 1, b = 2, 3 & 4 = ignored
```

C# = JS

Functions: Rest parameters

```
function test(a, b, ...c) {  
  console.log(a);  
  console.log(b);  
  console.log(c);  
}
```

```
test(1, 2, 3, 4); // a = 1, b = 2, c = [3, 4]
```

C# params [] = JS ...

Spread operator

```
function test(a, b, ...c) {  
  console.log(a);  
  console.log(b);  
  console.log(c);  
}
```

```
test(...[1, 2, 3, 4]); // a = 1, b = 2, c = [3, 4]
```


```
test(...'pqrs'); // a = 'p', b = 'q', c = ['r ', 's']
```

JS Only (C# only for params)

Arrow functions

```
const a = () => 'EcmaScript';
```

```
const b = (x) => x;
```



Omit braces

```
const c = x => x * x;
```

```
const d = x => { const y = x * x; return y; };
```



{ } when multiple statements

```
const e = (x, y) => x * y;
```

C# lambda = JS arrow

Arrow function options

- Default values

- ▶ `const f = (x = 10) => x * x;`
`console.log(f());` // 100

- Rest parameters

- ▶ `const x = (a, b, ...rest) => [a, b, rest];`
`console.log(x(1, 2, 3, 4));` // [1, 2, [3, 4]]

- Return object literal


- ▶ `const a = x => ({value: x});` // Must use ()
`console.log(a(123));` // { value: 123 }

Arrow function: this

- **this** is not bound (when called)

Classes

```
class Account extends Base {  
  constructor(name, amount) {  
    super(name);  
    this.balance = amount;  
  }  
  deposit(amount) {  
    this.balance += amount;  
  }  
};
```



No
function
keyword

```
const acc = new Account('Bill', 0);  
acc.deposit(100);  
console.log(acc); // { name: 'Bill', balance: 100 }
```

JS still prototype inheritance & different
syntax than C#

Shorthand properties

```
const name = 'Hillegom';  
const age = 1000;  
const dutch = true;  
  
const town = {  
  name,  
  age,  
  dutch  
};
```

JS Only

Method definitions

```
const obj = {  
  echo1: function (value) { return value; },  
  echo2(value) { return value; }  
};
```

Modules (direct)

file-name =
module name

```
// my-export.js
export function
square(x) {
  return x * x;
}
```

```
export let pi = 3.14;
```

```
// my-import.js
import { square, pi } from './my-export';
console.log(square(3)); // 9
console.log(pi); // 3.14
```

Modules (default)

```
// my-export.js  
function square(x) {  
    return x * x;  
}  
  
const pi = 3.14;  
  
export default {square, pi};
```

```
// my-import.js  
import my_export from './my-export';  
console.log(my_export.square(3)); // 9  
console.log(my_export.pi); // 3.14
```

C# namespaces look like JS modules

Destructuring: List matching

```
const data = [1, 22, 333, 4444, 55555];  
const [a, , b, ...rest] = data;  
  
console.log(a);           // 1  
console.log(b);           // 333  
console.log(rest);        // [4444, 55555]
```

Destructuring: Object matching

```
const obj = {  
  committee: 'TC39',  
  name: 'EcmaScript',  
  edition: { version: 6, year: 2015 },  
  website: 'https://github.com/tc39'  
};  
  
const { committee, name: officialName, edition: { year } } = obj;  
  
console.log(committee); // 'TC39'  
console.log(officialName); // 'EcmaScript'  
console.log(year); // 2015
```

Destructuring: Parameters, nested & defaults

function
parameter

nested

default

```
function test([value, {name}, year = 2017]) {  
  console.log(value);    // 1  
  console.log(name);     // EcmaScript  
  console.log(year);     // 2017  
}
```

```
test([1, {name: 'EcmaScript', year: 2015}]);
```


Async & Await (ES 2017)

```
async function write() {  
  const txt = await read();  
  console.log(txt);  
}
```

Trailing commas

- array

- ▶ `[1, 2,]`

- object

- ▶ `{
 one: 1,
 two: 2,
}`

- function (ES 2017)

- ▶ `function test(one, two,) { }`

- ▶ `test(1, 2,);`

JS Only

Rest/Spread properties (ES 2018)

old:

```
let x = Object.assign({}, {a: 1, b: 2}, {c: 3})
```

new:

```
let x = {...{a: 1, b: 2}, c: 3}
```

(where {a: 1, b: 2} can also be a variable)

“LINQ” functions on arrays

```
let people = [  
    { name: "Alice", age: 35 },  
    { name: "Ben", age: 40 },  
    { name: "Charlotte", age: 15 },  
];  
  
let where = people.filter(x => x.age >= 18); // adults only  
  
let select = people.map(x => x.name); // names only  
  
let all = people.every(x => x.age >= 18); // false  
  
let any = people.some(x => x.age >= 18); // true  
  
// Warning: In place, so methods also update people array!  
let orderBy = people.sort((a, b) => a.age > b.age); // by age
```

C# LINQ can be simulated by JS array methods

Compatibility ES 2015 and higher..

- ES 2015:
<http://kangax.github.io/compat-table/es6/>
- ES 2016, 2017, 2018+:
<http://kangax.github.io/compat-table/es2016plus/>

Compiler: Transpile ES201X to ES5

- [Babel](#)
- [Traceur](#)
- [TypeScript](#)

Babel

- Install npm (by installing [NodeJS](#))
- Command line:
 - ▶ `npm init`
 - ▶ `npm install babel-cli babel-polyfill babel-preset-env babel-preset-stage-3 --save-dev`
- Create file “.babelrc”
 - ▶

```
{  
  "presets": ["env", "stage-3"]  
}
```
- Command line (transpile all js-files in src-folder into the lib-folder):
 - ▶ `babel src --out-dir lib`

Polyfills

- <https://babeljs.io/docs/usage/polyfill/>

Packaging / Bundling + Minifying

- Why?

Packaging / Bundling + Minifying

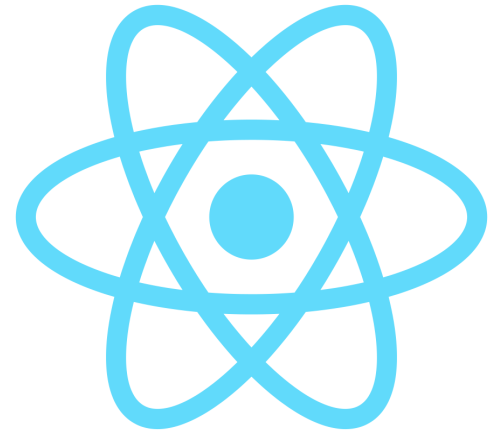
- Bundling:
 - ▶ Browsers can download max. 6 files at the same time
- Minifying:
 - ▶ Minimize download time

Training: rick@oblicum.com or [@rickbeerendonk](https://twitter.com/rickbeerendonk)



- **ECMAScript**

- 5, 2015, 2016, 2017, 2018...



- **React**

- Components, Properties, State, Events, Virtual DOM...



- **Redux**

- Actions, Reducers, Stores...



- **Samples & Slides**

- <https://github.com/rickbeerendonk/ECMAScript-examples>