

# ES2015

...

# Giacomo Zinetti Giko

🌐 A Nerd Frontend Developer

and...

Ducatista

Star Wars Lego builder

Board game player

Tchouker

<https://giko.it>

@giacomozinetti



# Block scoped variables

```
let a = 10;  
  
{  
  let a = 20;  
}  
  
// a === 10;
```

```
var a = 10;  
  
(function() {  
  var a = 20;  
})();  
  
// a === 10;
```

# Block scoped variables

```
for (let i = 1; i <= 5; i++) {  
  item = document.createElement('li');  
  
  item.onclick = function(ev) {  
    console.log('Item ' + i);  
  };  
  
}
```

```
for (var i = 1; i <= 5; i++) {  
  item = document.createElement('li');  
  
  (function(i) {  
  
    item.onclick = function(ev) {  
      console.log('Item ' + i);  
    };  
  
  })(i);  
}
```

# Constants

```
const PI = 3.141593;
```

```
PI = 6.022140;
```

*TypeError: Assignment to constant variable.*

```
// constant reference
```

```
const COLORS = {  
  good: 'green',  
  bad: 'red'  
}
```

```
COLORS.good = 'yellow';
```

```
Object.defineProperty(window, "PI", {
```

```
  value:      3.141593,
```

```
  enumerable: true,
```

```
  writable:   false,
```

```
  configurable: false
```

```
});
```

```
// Not block scoped
```

# Arrow functions

```
const sum = (a, b) => a + b;
```

```
const square = a => a * a;
```

```
const theAnswer = () => 42;
```

```
var sum = function(a, b) {  
  return a + b  
}
```

```
var square = function(a) {  
  return a * a;  
}
```

```
var theAnswer = function() {  
  return 42;  
}
```

# Arrow function - Lexical scope

```
function Timer() {  
  
    this.counter = 0;  
  
    setInterval(() => {  
        this.counter++;  
    }, 1000);  
}  
  
let p = new Timer();
```

```
function Timer() {  
  
    var that = this;  
  
    this.counter = 0;  
  
    setInterval(function() {  
        that.counter++;  
    }, 1000);  
}  
  
var p = new Timer();
```

# Default parameters

```
function next(x, step = 5) {  
  
    return x + step;  
  
}
```

```
function next(x, step) {  
  
    step = step || 1;  
  
    return x + step;  
  
}
```



# Rest parameters

```
function f(x, y, ...a) {  
  // a === [3, 4, 5]  
}
```

```
f(1, 2, 3, 4, 5);
```

```
function f(x, y) {  
  var a = Array  
    .prototype  
    .slice  
    .call(arguments, 2);  
};
```

# Spread operator

```
let args = [1, 2, 3, 4, 5];  
function sum(...n) {  
    return n.reduce((a, b) => a + b, 0);  
}  
sum(...args);
```

```
let arr1 = [0, 1, 2];  
let arr2 = [3, 4, 5];  
let arr = [...arr1, ...arr2, 6];
```

```
var args = [1, 2, 3, 4, 5];  
function sum() {  
    return Array.prototype.reduce  
        .call(arguments, function(a, b) {  
            return a + b;  
        }, 0);  
}  
sum.apply(null, args);
```

```
var arr1 = [0, 1, 2];  
var arr2 = [3, 4, 5];  
var arr = arr1.concat(arr2).concat(6);
```

# Template Literals

```
let message = `  
  Hello ${name.toUpperCase()}!  
  Today is your ${age}th birthday  
`;  
;
```

```
var message =  
  
"Hello " + name.toUpperCase() + "!\n" +  
"Today is your " + age + "th birthday";
```

# Object literals

```
let o = {  
  name,  
  rename(newName) {  
    this.name = newName;  
  },  
  ["__" + name]: 42,  
};
```

```
var o = {  
  name: name,  
  rename: function(newName) {  
    this.name = newName;  
  },  
};  
  
o["__" + name] = 42;
```

# Destructuring

```
let values = [1, 2];  
let [a, b] = values;
```

```
const coords = () => ({x: 40, y: -35});  
  
let {x, y} = coords();
```

```
var values = [1, 2];  
let a = values[0], b = values[1];
```

```
function coords() {  
    return { x: 40, y: -35 };  
}
```

```
var coord = coords();  
var x = coord.x;  
var y = coord.y;
```

# Classes

```
class Timer {  
  
    constructor (time) {  
        this.time = time;  
        this.start();  
    }  
  
    start () {  
        // ...  
    }  
  
}
```

```
var Timer = function (time) {  
    this.time = time;  
    this.start();  
};  
  
Timer.prototype.start = function() {  
    // ...  
};
```

# For - Of

```
let arr = [1, 2, 3, 4];  
  
for (let e of arr) {  
    console.log(e);  
}
```

```
var arr = [1, 2, 3, 4];  
  
for (var i = 0; i < arr.length; i++) {  
    console.log(arr[i]);  
}
```

# New methods

```
[1, 3, 4].find(x => x > 2)
```

```
"<el></el>".repeat(3)
```

```
"hello".startsWith("he")
```

```
"hello".endsWith("lo")
```

```
"hello".includes("el")
```

```
Math.trunc(x)
```

```
Math.sign(x)
```

```
[1, 3, 4].filter(function (x) { return x > 2; })[0]
```

```
Array(3 + 1).join("<el></el>")
```

```
"hello".indexOf("he") === 0;
```

```
"hello".indexOf("lo") === ("hello".length - 2)
```

```
"hello".indexOf("el") !== -1;
```

```
(x < 0 ? Math.ceil(x) : Math.floor(x))
```

```
(x > 0 ? 1 : -1)
```



# And so on... but we need more time

- Promises
- Generators
- Proxies
- Intl
- Symbols
- Maps and Sets
- Modules
- Tail call

# Our friends

- Babel - <https://babeljs.io/>
- Caniuse - <http://caniuse.com/>
- <http://kangax.github.io/compat-table/es6/>



**Grazie!** 

**@giacomozinetti**