ECMAScript 2015

Top 10 features*

*by qmmr

About myself…

Twitter: @qmmr
Github: github.com/qmmr

A little bit of history…

A brief history of JavaScript

- May 1995, created in 10 days by Brendan Eich, at Netscape, named “Mocha”
- September 1995, shipped in beta of Netscape Navigator 2.0: “LiveScript”
- December 1995, Netscape 2.0b3: “JavaScript” renamed after the popular Java language

A brief history of JavaScript

- August 1996, JavaScript cloned in Microsoft IE 3.0: “JScript”
- 1996-1997, Standardization ECMA-262 Ed. 1: ”ECMAScript” aka ES1
- The standard received a significant update as ECMAScript edition 3 in 1999
- The fourth edition was abandoned.
- ECMAScript edition 5, published in December of 2009

A brief history of JavaScript

- June 2015 - ECMAScript 2015
- 2016 - ECMAScript 2016
- 2017 - ???

#10

DEFAULT/REST/SPREAD PARAMETERS

#10 - Default parameters

```
function timeout(timeout = 2000, callback = function() {}) {
    // the rest of the function
}
```

#10 - Default parameters

```
function getCallback() {
    return function() {};
}

function timeout(timeout = 2000, callback = getCallback()) {
    // the rest of the function
}
```

#10 - Rest parameters

```
var sum = function(...rest) {
    return rest.reduce(function(accu, curr) {
        return accu + curr;
    }, 0);
}
sum(1, 2, 3, 4, 5); // 15
```

#10 - Spread parameters

```
var numbers = [ 20, 22, 42 ];
Math.max(...numbers); // 42
```

#10 - Spread parameters

```
var numbers = [ 10, 12, 20 ];

var sum = function(x, y, z) {
    return x + y + z;
}

sum(...numbers); // 42
```

#9

PROMISES

#9 - Promises

```
function timeout(duration = 1000) {
    return new Promise(function(resolve, reject) {
        setTimeout(function() {
            resolve(42);
        }, duration);
    });
}

timeout(500).then(function(value) {
    console.log('Value: ' + value); // 42
});
```

#9 - Promises

```
function httpGet(url) {
    return new Promise(function(resolve, reject) {
        var request = new XMLHttpRequest();
        request.onreadystatechange = function() {
            if (this.status === 200) {
                resolve(this.response); // Success
            } else {
                // Something went wrong (404 etc.)
                reject(new Error(this.statusText));
            }
        }
        request.onerror = function() {
            reject(new Error('Error: ' + this.statusText));
        };
        request.open('GET', url);
        request.send();
    });
}
```

#9 - Promises

```
httpGet('http://example.com/file.txt')
    .then(
        function(value) {
            // fullfilment
            console.log('Contents: ' + value);
        },
        function(err) {
            // rejection
            console.error('Something went wrong', err);
        }
    );
```

#9 - Promises

```
function timeout(duration = 1000) {
    return new Promise(function(resolve, reject) {
        setTimeout(function() {
            resolve(duration + 'ms');
        }, duration);
    });
}
```

```
Promise.all([ timeout(), timeout(1500), timeout(2000) ])
    .then(function(value) {
        console.log(value); // [ "1000ms", "1500ms", "2000ms" ]
    });

Promise.race([ timeout(), timeout(1500), timeout(2000) ])
    .then(function(value) {
        console.log(value); // "1000ms"
    });
```

#8 | *OBJECT.ASSIGN*

#8 - Object.assign

```
var person = {
    firstName: 'Peter',
    lastName: 'Parker'
};

var superhero = {
    alias: 'Spider-man',
    superpowers: [ 'web', 'wall-climbing' ]
};
```

```
var spiderman = Object.assign({}, person, superhero);

/*{
    "firstName": "Peter",
    "lastName": "Parker",
    "alias": "Spider-man",
    "superpowers": [ "web", "wall-climbing" ]
}*/
```

#7

TEMPLATE LITERALS
TAGGED TEMPLATES

#7 - Template literals/tagged templates

```
// template literals
`string text line 1
string text line 2
    string text line 3`

// String interpolation
`string text ${expression} string text`
```

#7 - Template literals/tagged templates

```
var greet = function(greet, name) {
    return `${ greet } ${ name },
how are you today?`;
};
```

```
console.log(greet('Hi', ‘Marcin’));
"Hi Marcin,
how are you today?"
```

#7 - Template literals/tagged templates

```
// tagged templates
tag `string text ${expression} string text`
```

#7 - Template literals/tagged templates

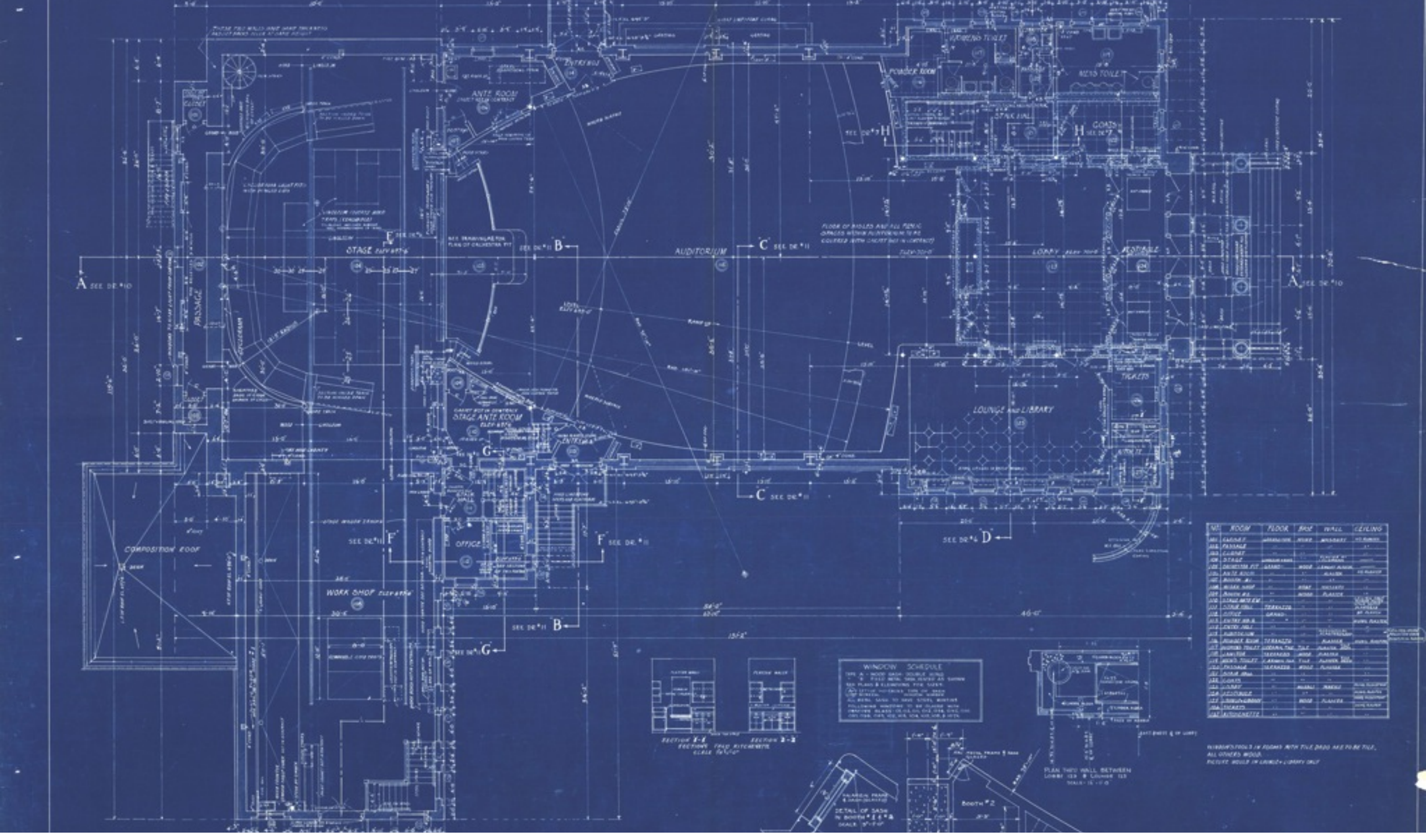
```
var lovem = function(strings, user) {
    return `${ user } <3 ${ strings[1] }!`;
};

var user = 'Marcin';

console.log(lovem `${ user }ECMAScript2015`);
// “Marcin <3 ECMAScript2015!”
```

#6

ENHANCED OBJECT LITERALS

#6 - Enhanced object literals

```
var title = 'ECMAScript 2015';
var students = [ 'Joe', 'Jane', 'Phil' ];
var id = function() { return 42 };
var course = {
    title,
    students,
    welcome() {
        return `Welcome to ${ this.title } course!`;
    },
    [ `_prop${ id() }` ]: id()
};
```


AUDITORIUM
STAGE
STAGE ANTE ROOM
ANTE ROOM
LOBBY
LOUNGE and LIBRARY
TICKETS
COMPOSITION ROOF
WORK SHOP
OFFICE
MEN'S TOILET
WOMEN'S TOILET
powder room
GOATS
STALLS
WINDOW SCHEDULE

#5 | *CLASSES*

classes

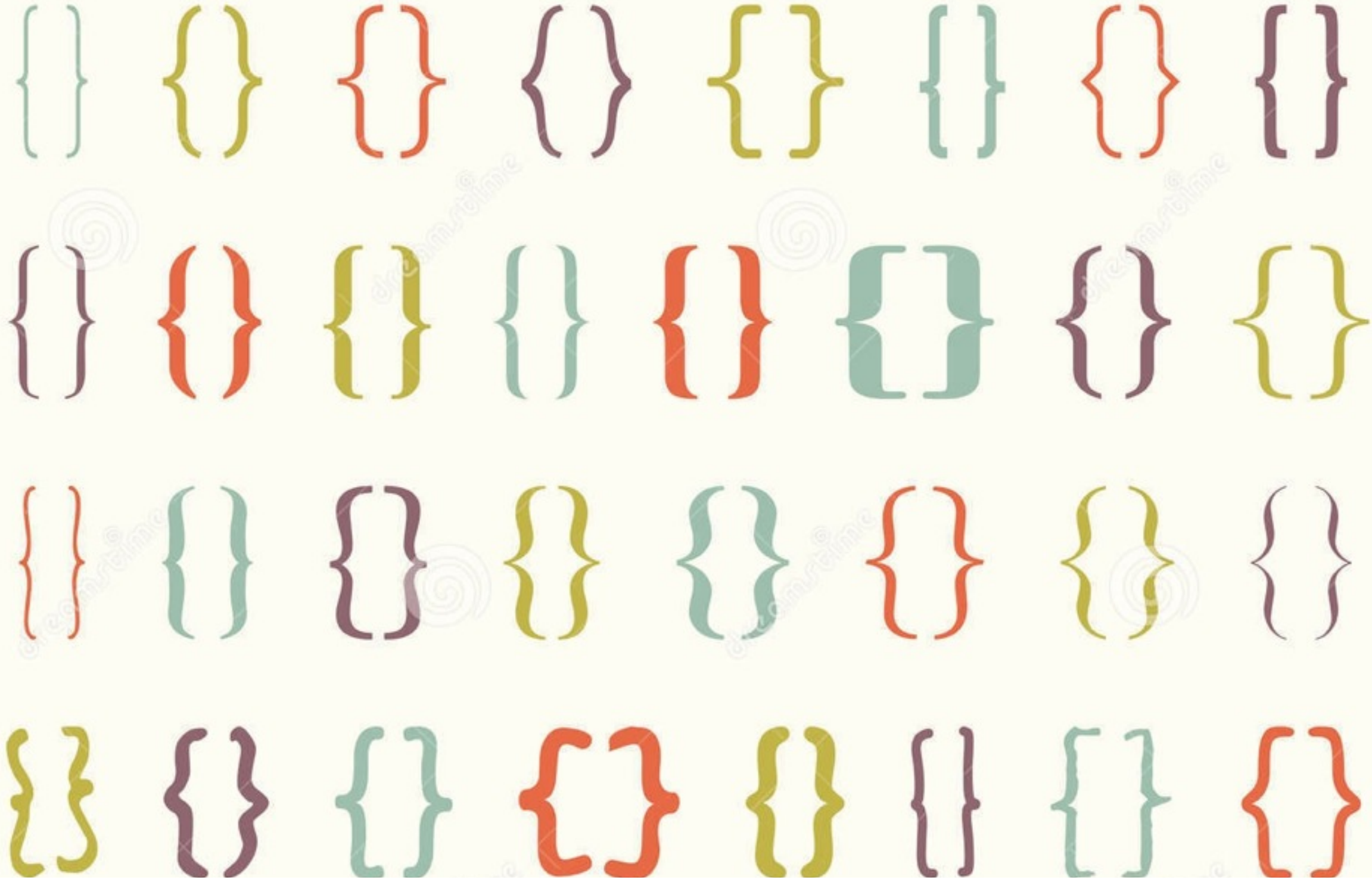
```
class Human {
    constructor(firstName = 'Joe', lastName = 'Doe') {
        this.firstName = firstName;
        this.lastName = lastName;
    }
    greet() {
        return `Hi, my name is ${ this.firstName }.`;
    }
    static type() {
        return 'human';
    }
}

var joe = new Human();
console.log(joe.greet()); // “Hi, my name is Joe.”
console.log(Human.type()); // “human”
```

classes

```
class SuperHuman extends Human {
    constructor(firstName, lastName, alias = 'superhuman') {
        super(firstName, lastName);
        this.alias = alias;
    }

    greet() {
        return `Hi, I am ${ this.alias }!`;
    }

    revealIdentity() {
        return `Psst... It's me, ${ this.firstName }!`;
    }
}
new SuperHuman('Peter', 'Parker', 'Spider-man');
```

classes

```
class Person {
    constructor(firstName = 'Joe', lastName = 'Doe') {
        this.firstName = firstName;
        this.lastName = lastName;
    }

    get fullName () {
        return `${this.lastName}, ${this.firstName}`;
    }

    set occupation(occupation) {
        return this._occupation = occupation;
    }
}
var jane = new Person(‘Jane’).fullName; // “Doe, Jane”
```



#4 | *LET/CONST*

#4 - let/const

```
function foo() {
    let bar = 5;
    if (1) {
        let bar = 10; // shadows outer `bar`
        console.log(bar); // 10
    }
    console.log(bar); // 5
}
```

#4 - let/const

```
let arr = [];

for (var i = 0; i < 3; i++) {
    arr.push(function() {
        return i;
    });
}

arr.map(function(fn) {
    return fn();
}); // [ 3, 3, 3 ]
```

#4 - let/const

```
let arr = [];

for (let i = 0; i < 3; i++) {
    arr.push(function() {
        return i;
    });
}

arr.map(function(x) {
    return x();
}); // [ 0, 1, 2 ]
```

#4 - let/const

```
const ULTIMATE_NUMBER;
// SyntaxError: missing = in const declaration

const ULTIMATE_NUMBER = 42;

ULTIMATE_NUMBER = 23;
// re-assign to ULTIMATE_NUMBER, fails

console.log(ULTIMATE_NUMBER); // 42

const ULTIMATE_NUMBER = 23;
// redeclare a constant throws an error

var ULTIMATE_NUMBER = 23;
// ULTIMATE_NUMBER is reserved for constant above, fails
```

#4 - let/const

```
const myObject = { "name": “Joe” };

myObject = {}; // fails

myObject.name = “Jane”; // works

const myArray = [];

myArray.push(‘foo’);
```

#4 - let/const

```
if (true) { // enter new scope, TDZ starts
    // Uninitialized binding for `tmp` is created

    console.log(typeof tmp); // ReferenceError
    tmp = 'abc'; // ReferenceError
    console.log(tmp); // ReferenceError

    let tmp; // TDZ ends, `tmp` is `undefined`
    console.log(tmp); // undefined

    tmp = 123;
    console.log(tmp); // 123
}
```

#3

DESTRUCTURING

#3 - Destructuring

```
var a = 1;
var b = 2;
var tmp = a;

a = b;
b = tmp;

console.log(a, b); // 2, 1
```

#3 - Destructuring

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

console.log(a, b); // 1, 2

[ a, b ] = [ b, a ];

console.log(a, b); // 2, 1
```

#3 - Destructuring

```
let getColors = function() {
    return [ '#008744', '#0057e7', '#d62d20' ];
};

let [ green, blue, red ] = getColors();

console.log(green, blue, red);
// "#008744" "#0057e7" "#d62d20"
```

#3 - Destructuring

```
let getColors = function() {
    return [ '#008744', '#0057e7', '#d62d20' ];
};

let [ green, ...colors ] = getColors();

console.log(green, colors);
// “#008744”, [ “#0057e7”, “#d62d20” ]
```

#3 - Destructuring

```
const person = {
    firstName: 'Joe',
    lastName: 'Doe'
};
```

```
let { firstName, lastName } = person;

console.log(firstName, lastName);
// “Joe”, "Doe"
```

#3 - Destructuring

```
const person = {
    firstName: 'Joe',
    lastName: 'Doe'
};
```

```
let { firstName: first, lastName: last } = person;

console.log(first, last);
// “Joe”, "Doe"
```

#3 - Destructuring

```
const person = {
    firstName: 'Joe',
    lastName: 'Doe',
    email: {
        personal: 'joe.doe@gmail.com',
        work: 'joe.doe@gamesys.co.uk'
    }
};
```

```
let {
    firstName,
    lastName,
    email: { personal: email }
} = person;

console.log(firstName, lastName, email);
// “Joe”, “Doe”, "joe.doe@gmail.com"
```

#3 - Destructuring

```
const person = {
    firstName: 'Joe',
    lastName: 'Doe',
    emails: [
        ‘joe.doe@gmail.com',
        ‘joe.doe@gamesys.co.uk'
    ]
};
```

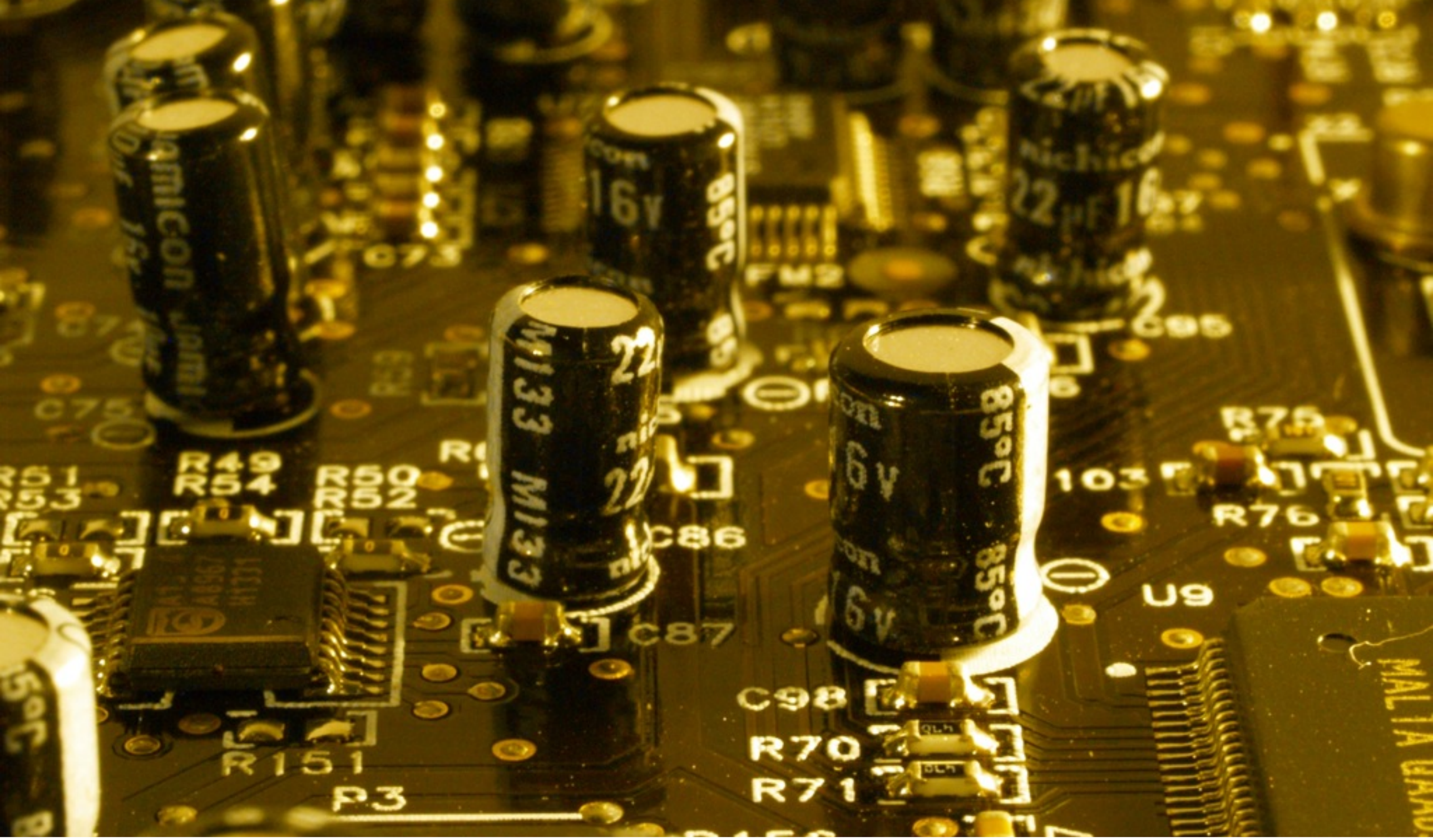
```
let {
    firstName,
    lastName,
    emails: [ personalEmail, workEmail ]
} = person;

console.log(firstName, lastName); // “Joe”, "Doe"
console.log(personalEmail, workEmail);
// “joe.doe@gmail.com”, "joe.doe@gamesys.co.uk"
```

#3 - Destructuring

```
const person = {
    firstName: 'Joe',
    lastName: 'Doe'
};

function getFullname({ firstName, lastName }) {
    return `${ lastName }, ${ firstName }`;
}

getFullname(person); // "Doe, Joe"
```

#3 - Destructuring

```
const person = {
    firstName: 'Joe',
    lastName: 'Doe'
};

function getFullname({ firstName='Joe', lastName='Doe' } = {}) {
    return `${ lastName }, ${ firstName }`;
}

getFullname(person) // "Doe, Joe”
getFullname({ firstName: 'Joe' }) // "Doe, Joe"
getFullname({ lastName: 'Doe' }) // "Doe, Joe"
getFullname() // "Doe, Joe"
```



#2 | *MODULES*

#2 - Modules

```
var galaxy = ‘Milky Way’;

// export variables
export var planet = 'Earth';
export let distanceFromSun = 149600000;

// define constants
const MINI_PI = 3.14;
const EARTH_RADIUS = 6378;

// export constants
export MINI_PI;
export EARTH_RADIUS;
```

#2 - Modules

```
// this function is private to the module
function toFixed(num) {
    return Math.ceil(num * 100) / 100;
}

// export function
export function circumference(radius) {
    return toFixed(MINI_PI * (2 * radius));
}

// export class
export class Galaxy {
    constructor(sun, planets) {
        this.sun = sun;
        this.planets = planets;
    }
}
```

#2 - Modules

```
import { circumference, Galaxy } from './circumference';
import { EARTH_RADIUS } from './constants';

let earthCircum = circumference(EARTH_RADIUS);
let milkyWay = new Galaxy(‘sun’, []);

circumference = 40075; // error
```

#2 - Modules

```
// math.js
export default function sum(num1, num2) {
    return num1 + num2;
}

// main.js
import sum from "./math";
```

#2 - Modules

```
// math.js
export MINI_PI = 3.14;
export default function sum(num1, num2) {
    return num1 + num2;
}
```

```
// main.js
import sum, { MINI_PI } from "./math";

console.log(sum(1, 2)); // 3
console.log(MINI_PI); // 3.14
```

#1

=> ARROW FUNCTION

#1 => arrow functions

```
var reflect = function(value) {
    return value;
};
```

#1 => arrow functions

```
var reflect = (value) => {
    return value;
}
```

#1 => arrow functions

```
var reflect = (value) => value;
```

#1 => arrow functions

```
var reflect = value => value;
```

#1 => arrow functions

```
let reflect = value => value
```

#1 => arrow functions

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

#1 => arrow functions

```
var getName = () => 'Marcin';
```

#1 => arrow functions

```
let numbers = [ 0, 1, 2, 3, 4, 5 ];
let doubles = numbers.map(n => n * n);
// [ 0, 1, 4, 9, 16, 25 ]
```

#1 => arrow functions

```
var sum = function(...rest) {
    return rest.reduce(function(accu, curr) {
        return accu + curr;
    }, 0);
}
sum(1, 2, 3, 4, 5); // 15
```

#1 => arrow functions

```
let sum = function(...rest) {
    return rest.reduce((accu, curr) => accu + curr, 0);
}
```

#1 => arrow functions

```
let sum = ...rest => rest.reduce((accu, curr) => accu + curr, 0);
```

#1 => arrow functions

```
let points = [ 100, 50, 42 ];

let getCoords = (x, y, z) => ({ x, y, z });

let coords = getCoords(...points);

// { "x": 100, "y": 50, "z": 42 }
```

#1 => arrow functions

```
const person = {
    name: 'Joe',
    getName() {
        setTimeout(() => console.log(this.name), 1000);
    }
};
```

Links

- ★ http://www.ecma-international.org
- ★ http://www.ecma-international.org/publications/standards/Ecma-262.htm
- ★ https://leanpub.com/understandinges6
- ★ https://leanpub.com/exploring-es6
- ★ https://github.com/lukehoban/es6features
- ★ http://babeljs.io/
- ★ https://github.com/google/traceur-compiler
- ★ http://kangax.github.io/compat-table/es6/
- ★ http://www.es6fiddle.net

Questions?

Twitter: @qmmr
Github: github.com/qmmr

Thank you.