Next-Gen JavaScript

A Refresher

Welcome to this module in this course. I'm going to say it right away. This module is optional. If you already know next generation javascript, so things like arrow functions, let, const and so on, You may very well skip this module. This module is for you if you know javascript but haven't worked with ES6 or any other version of javascript. Then javascript as we use it in this course might sometimes look like a new language to you because of all the next generation features React apps typically use. Now I'm not using these next generation features because I want to show off or anything like that. React apps typically are built with the latest version of javascript and using the next gen features allows us to write clean and robust React apps. React itself uses a lot of Next Generation Javascript features. Therefore it's important for you to understand all these features and not be confused by this. Sometimes really strange looking version of Javascript. Javascript is evolving quickly and therefore new features can look different but really allow us as a developer to do more powerful things. This module is for you, I will walk you through the core features we use in this course so that the code thereafter hopefully looks a bit less strange to you. And again, feel free to skip this module if you already know it, feel free to always come back to it if you later in the course encounter something that looks strange to you.

let & const

var

let

variable values

const

constant values

JS Bin - Collaborative JavaSc

jsbin.com/?html,output

File Add library HTML CSS JavaScript Console Output Login or Register Blog 1 Help

HTML

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width">
  <title>JS Bin</title>
</head>
<body>

</body>
</html>
```

Output

Run with JS Auto-run JS

JS Bin - Collaborative JavaSc

jsbin.com/?js,console

File Add library HTML CSS JavaScript Console Output Login or Register Blog 1 Help

JavaScript

```
var myName = 'Max';
console.log(myName);

myName = 'Manu';
console.log(myName);
```

Console Run Clear

```
"Max"

"Manu"

>
```

JS Bin - Collaborative JavaSc

jsbin.com/?js,console

File Add library HTML CSS JavaScript Console Output Login or Register Blog 1 Help

Run Clear

```
'let' is available in ES6 (use esnext option) or Mozilla JS extensions
(use moz).
let myName = 'Max';
console.log(myName);

myName = 'Manu';
console.log(myName);
```

```
"Max"

"Manu"

>
```

JS Bin - Collaborative JavaScr

jsbin.com/?js,console

File Add library HTML CSS JavaScript Console Output Login or Register Blog 1 Help

Run Clear

⚠ 'const' is available in ES6 (use esnext option) or Mozilla JS extensions (use moz).

```
const myName = 'Max';
console.log(myName);

myName = 'Manu';
console.log(myName);
```

```
"Max"

✕ "TypeError: Assignment to constant variable.
    at null.js:4:8
    at https://static.jsbin.com/js/prod/runner-4.1.0.min.js:1:13850
    at https://static.jsbin.com/js/prod/runner-4.1.0.min.js:1:10792"

›
```

Arrow Functions

```
function myFnc() {
    ...
}
```

```
const myFnc = () => {
    ...
}
```

No more issues with the this keyword!

JS Bin - JS Bin

jsbin.com/capasenona/edit?js,console

File Add library Share HTML CSS JavaScript Console Output Login or Register Blog 1 Help

JavaScript

```
function printMyName(name) {
  console.log(name);
}

printMyName('Max');
```

Console Run Clear

```
"Max"
>
```

Bin info
just now

JS Bin - JS Bin

jsbin.com/capasenona/edit?js,console

File Add library Share HTML CSS JavaScript Console Output Login or Register Blog 1 Help

JavaScript

```
const printMyName = (name) => {
  console.log(name);
}

printMyName('Max');
```

Console Run Clear

```
"Max"
>
```

Bin info
just now

JS Bin - JS Bin

jsbin.com/capasenona/edit?js,console

File Add library Share HTML CSS JavaScript Console Output Login or Register Blog 1 Help

JavaScript

```
const printMyName = name => {
  console.log(name);
}

printMyName('Max');
```

Console Run Clear

```
"Max"
>
```

That's only valid for exactly one argument though, not for more and not for less. Here,

Bin info
just now

JS Bin - JS Bin

jsbin.com/capasenona/edit?js,console

File Add library Share HTML CSS JavaScript Console Output Login or Register Blog 1 Help

JavaScript

```
const printMyName = () => {
  console.log('Max');
}

printMyName();
```

Console Run Clear

```
"Max"
>
```

Bin info
just now

JS Bin - JS Bin

jsbin.com/capasenona/edit?js,console

File Add library Share HTML CSS JavaScript Console Output Login or Register Blog 1 Help

JavaScript

```
const printMyName = (name, age) => {
  console.log(name, age);
}

printMyName('Max', 28);
```

Console Run Clear

```
"Max"
28
>
```

Bin info
just now

JS Bin - JS Bin

jsbin.com/capasenona/edit?js,console

File Add library Share HTML CSS JavaScript Console Output Login or Register Blog 1 Help

JavaScript

```
const multiply = (number) => {
  return number * 2;
}

console.log(multiply(2));
```

Console Run Clear

```
4

>
```

Bin info
just now

JS Bin - JS Bin

jsbin.com/capasenona/edit?js,console

File Add library Share HTML CSS JavaScript Console Output Login or Register Blog 1 Help

JavaScript

```
const multiply = number => number * 2;

console.log(multiply(2));
```

Console Run Clear

```
4

>
```

Bin info
just now

JS Bin - JS Bin

jsbin.com/capasenona/edit?js,console

File Add library Share HTML CSS JavaScript Console Output Login or Register Blog 1 Help

JavaScript

⚠ 'arrow function syntax (=>)' is available in ES6 (use esnext option) or Mozilla JS extensions (use moz).

Run Clear

```
const multiply = number => number * 2;

console.log(multiply(2));
```

```
4

>
```

Bin info
just now

Exports & Imports (Modules)

person.js

```
const person = {
   name: 'Max'
}

export default person
```

utility.js

```
export const clean = () => { ... }

export const baseData = 10;
```

app.js

```
import person from './person.js'
import prs from './person.js'

import { baseData } from './utility.js'
import { clean } from './utility.js'
```

Imports default and ONLY export of the file Name in the receiving file is up to you

Exports & Imports (Modules)

default export

```
import person from './person.js'

import prs from './person.js'
```

named export

```
import { smth } from './utility.js'

import { smth as Smth } from './utility.js'

import * as bundled from './utility.js'
```

You choose the name

Name is defined by export

Exports & Imports (Modules)

default export

```
import person from './person.js'

import prs from './person.js'
```

named export

```
import { smth } from './utility.js'

import { smth as Smth } from './utility.js'

import * as bundled from './utility.js'
```

You choose the name

Name is defined by export

file as properties so that you simply have bundled.baseData, bundled.clean to access the

Classes

Property

```
class Person {
    name = 'Max'
    call = () => {...}
}
```

Method

Usage

(constructor functions anyone?)

```
const myPerson = new Person()
myPerson.call()
console.log(myPerson.name)
```

Inheritance

(prototypes anyone?)

```
class Person extends Master
```

JS Bin - JS Bin

jsbin.com/capasenona/edit?js,console

File Add library Share HTML CSS JavaScript Console Output Login or Register Blog 1 Help

'class' is available in ES6 (use esnext option) or Mozilla JS extensions (use moz).

Run Clear

```
class Person {
  constructor() {
    this.name = 'Max';
  }

  printMyName() {
    console.log(this.name);
  }
}

const person = new Person();
person.printMyName();
```

```
"Max"
>
```

Bin info
just now

JS Bin - JS Bin

jsbin.com/capasenona/edit?js,console

File Add library Share HTML CSS JavaScript Console Output Login or Register Blog 1 Help

JavaScript

```
class Human {
  constructor() {
    this.gender = 'male';
  }

  printGender() {
    console.log(this.gender);
  }
}

class Person extends Human {
  constructor() {
    this.name = 'Max';
  }

  printMyName() {
    console.log(this.name);
  }
}

const person = new Person();
person.printMyName();
person.printGender();
```

Console Run Clear

```
"error"

"ReferenceError: Must call super constructor in derived class before accessing 'this' or returning from derived constructor
    at new Person (capasenona.js:13:5)
    at capasenona.js:21:16
    at https://static.jsbin.com/js/prod/runner-4.1.0.min.js:1:13850
    at https://static.jsbin.com/js/prod/runner-4.1.0.min.js:1:10792"

>
```

Bin info
just now

JS Bin - JS Bin

jsbin.com/capasenona/edit?js,console

File Add library Share HTML CSS JavaScript Console Output Login or Register Blog 1 Help

JavaScript

```javascript
class Human {
  constructor() {
    this.gender = 'male';
  }

  printGender() {
    console.log(this.gender);
  }
}

class Person extends Human {
  constructor() {
    super();
    this.name = 'Max';

  }

  printMyName() {
    console.log(this.name);
  }
}

const person = new Person();
person.printMyName();
person.printGender();
```

Console Run Clear

```
"Max"

"male"

>
```

Bin info
just now

JS Bin - JS Bin

jsbin.com/capasenona/edit?js,console

File Add library Share HTML CSS JavaScript Console Output Login or Register Blog 1 Help

JavaScript

```javascript
class Human {
  constructor() {
    this.gender = 'male';
  }

  printGender() {
    console.log(this.gender);
  }
}

class Person extends Human {
  constructor() {
    super();
    this.name = 'Max';
    this.gender = 'female';
  }

  printMyName() {
    console.log(this.name);
  }
}

const person = new Person();
person.printMyName();
person.printGender();
```

Console Run Clear

```
"Max"
"female"
>
```

Bin info
just now

Classes, Properties & Methods

Properties are like "variables attached to classes/ objects"

ES6

```
constructor () {
        this.myProperty = ' value'
}
```

ES7

```
myProperty = ' value'
```

Methods are like "functions attached to classes/ objects"

ES6

```
myMethod () { ... }
```

ES7

```
myMethod = () => { ... }
```

Classes, Properties & Methods

Properties are like "variables attached to classes/ objects"

ES6

```
constructor () {
        this.myProperty = ' value'
}
```

ES7

```
myProperty = ' value'
```

Methods are like "functions attached to classes/ objects"

ES6

```
myMethod () { ... }
```

ES7

```
myMethod = () => { ... }
```

JS Bin - JS Bin

jsbin.com/capasenona/edit?js,console

File Add library Share HTML CSS JavaScript Console Output Login or Register Blog 1 Help

JavaScript

```
class Human {
  gender = 'male';

  printGender = () => {
    console.log(this.gender);
  }
}

class Person extends Human {
    name = 'Max';
    gender = 'female';

  printMyName = () => {
    console.log(this.name);
  }
}

const person = new Person();
person.printMyName();
person.printGender();
```

Console Run Clear

```
"error"

"SyntaxError: Unexpected token =
    at https://static.jsbin.com/js/prod/runner-4.1.0.min.js:1:13850
    at https://static.jsbin.com/js/prod/runner-4.1.0.min.js:1:10792"

>
```

Bin info
just now

jsbin.com/capasenona/edit?js,console#javascriptprocessors

JS Bin - JS Bin

jsbin.com/capasenona/edit?js,console

File Add library Share HTML CSS JavaScript Console Output Login or Register Blog 1 Help

JavaScript

- JavaScript
- ES6 / Babel
- JSX (React)
- CoffeeScript
- Traceur
- TypeScript
- Processing
- LiveScript
- ClojureScript
- Convert to JavaScript

Console Run Clear

```
"error"

"SyntaxError: Unexpected token =
    at https://static.jsbin.com/js/prod/runner-
4.1.0.min.js:1:13850
    at https://static.jsbin.com/js/prod/runner-
4.1.0.min.js:1:10792"
```

›

Bin info
just now

jsbin.com/capasenona/edit?js,console#babel

JS Bin - JS Bin

jsbin.com/capasenona/edit?js,console

File Add library Share HTML CSS ES6 / Babel Console Output Login or Register Blog 1 Help

ES6 / Babel

```
class Human {
  gender = 'male';

  printGender = () => {
    console.log(this.gender);
  }
}

class Person extends Human {
    name = 'Max';
    gender = 'female';

  printMyName = () => {
    console.log(this.name);
  }
}

const person = new Person();
person.printMyName();
person.printGender();
```

Console Run Clear

```
"Max"

"female"

>
```

You'll still see the arrows on the left but you may ignore them.

Bin info
just now

Spread & Rest Operators

...

Spread

Used to split up array elements OR object properties

```
const newArray = [...oldArray, 1, 2]
const newObject = { ...oldObject, newProp: 5 }
```

Rest

Used to merge a list of function arguments into an array

```
function sortArgs(...args) {
    return args.sort()
}
```

JS Bin - JS Bin

jsbin.com/capasenona/edit?js,console

File Add library Share HTML CSS ES6 / Babel Console Output Login or Register Blog 1 Help

ES6 / Babel

```
const numbers = [1, 2
const newNumbers = [...numbers, 4];

console.log(newNumbers);
```

⚠ 'spread/rest operator' is available in ES6 (use esnext option) or Mozilla JS extensions (use moz).

Console

Run Clear

›

Bin info
just now

JS Bin - JS Bin

jsbin.com/capasenona/edit?js,console

File Add library Share HTML CSS ES6 / Babel Console Output Login or Register Blog 1 Help

ES6 / Babel

```
const numbers = [1, 2, 3];
const newNumbers = [...numbers, 4];

console.log(newNumbers);
```

Console

Run Clear

```
[1, 2, 3, 4]
```

Bin info
just now

JS Bin - JS Bin

jsbin.com/capasenona/edit?js,console

File Add library Share HTML CSS ES6 / Babel Console Output Login or Register Blog 1 Help

ES6 / Babel

```
const numbers = [1, 2, 3];
const newNumbers = [numbers, 4];

console.log(newNumbers);
```

Console Run Clear

```
[[1, 2, 3], 4]
```

Bin info
just now

JS Bin - JS Bin

jsbin.com/capasenona/edit?js,console

File Add library Share HTML CSS ES6 / Babel Console Output Login or Register Blog 1 Help

ES6 / Babel

```
const person = {
  name: 'Max'
};

const newPerson = {
  ...person,
  age: 28
}

console.log(newPerson);
```

Console

Run Clear

>

And now if I console log newPerson you may ignore this error which is thrown by jsbin, the syntax will be supported in our course project.

will be supported in our course project.

Bin info
just now

JS Bin - JS Bin

jsbin.com/capasenona/edit?js,console

File Add library Share HTML CSS ES6 / Babel Console Output Login or Register Blog 1 Help

ES6 / Babel

```
const person = {
  name: 'Max'
};

const newPerson = {
  ...person,
  age: 28
}

console.log(newPerson);
```

Console Run Clear

```
[object Object] {
  age: 28,
  name: "Max"
}
```

Bin info
just now

JS Bin - JS Bin

jsbin.com/capasenona/edit?js,console

File Add library Share HTML CSS ES6 / Babel Console Output Login or Register Blog 1 Help

ES6 / Babel

```
const filter = (...args) => {
  return args.filter(el => el === 1);
}
```

Console

Run Clear

>

Three equals signs checks for type and value equality so that el also has to be a number.

Bin info

just now

JS Bin - JS Bin

jsbin.com/capasenona/edit?js,console

File Add library Share HTML CSS ES6 / Babel Console Output Login or Register Blog 1 Help

ES6 / Babel

```
Mozilla JS extensions (use moz).
⚠ 'spread/rest operator' is available in ES6 (use esnext option) or
Mozilla JS extensions (use moz).
```

Run Clear

```
const filter = (...args) => {
  return args.filter(el => el === 1);
}

console.log(filter(1, 2, 3));
```

```
[1]

>
```

Bin info
just now

Destructuring

Easily extract array elements or object properties and store them in variables

Array Destructuring

```
[a, b] = ['Hello', 'Max']
console.log(a) // Hello
console.log(b) // Max
```

Object Destructuring

```
{name} = {name: 'Max', age: 28}
console.log(name) // Max
console.log(age) // undefined
```

JS Bin - JS Bin

jsbin.com/capasenona/edit?js,console

File Add library Share HTML CSS ES6 / Babel Console Output Login or Register Blog 1 Help

ES6 / Babel

```
const numbers = [1, 2, 3];
[num1, num2] = numbers;
console.log(num1, num2);
```

Console Run Clear

```
1
2
>
```

Bin info
just now

JS Bin - JS Bin

jsbin.com/capasenona/edit?js,console

File Add library Share HTML CSS ES6 / Babel Console Output Login or Register Blog 1 Help

ES6 / Babel

```
const numbers = [1, 2, 3];
[num1, , num3] = numbers;
console.log(num1, num3);
```

Console Run Clear

```
1
3
>
```

Bin info
just now

File Add library Share HTML CSS ES6 / Babel Console Output Login or Register Blog Help

ES6 / Babel

```
const number = 1;
const num2 = number;

console.log(num2);
```

Console

Run Clear

```
1
>
```

If I create a number like this, then this is a primitive type.
That means if I create a second number num2 and set it equal to this number then it will actually create a real copy of number so num2 of course.

Now if I log this, it will also be one.
But it will have copied that value one into num2.

Bin info
just now

JS Bin - JS Bin

jsbin.com/capasenona/edit?js,console

File Add library Share HTML CSS ES6 / Babel Console Output Login or Register Blog 1 Help

ES6 / Babel

```
const number = 1;
const num2 = number;

console.log(num2);
```

Console Run Clear

```
1
>
```

Now numbers, strings, booleans, these are so-called primitive types whenever you reassign or you store a variable in another variable. It will copy the value, objects and arrays are reference types.

Though let me show you what I mean.

Bin info
just now

JS Bin - JS Bin

jsbin.com/capasenona/edit?js,console

File ▾ Add library Share HTML CSS ES6 / Babel Console Output Login or Register Blog 1 Help

ES6 / Babel ▾

```
const person = {
  name: 'Max'
};

const secondPerson = person;

console.log(secondPerson);
```

Console Run Clear

```
[object Object] {
  name: "Max"
}

›
```

I create my personal object which just has a name here and I now create a secondPerson and assigned person as a value here.
If I console log. secondPerson and hit run, it will print the same value as the first person

but it will not actually have copied the person instead. Person the object is stored in memory and in the constant person we store a pointer to that place in memory. And if we then assign person to secondPerson that pointer will be copied.

Bin info
just now

JS Bin - JS Bin

jsbin.com/capasenona/edit?js,console

File ▾ Add library Share HTML CSS ES6 / Babel Console Output Login or Register Blog 1 Help

ES6 / Babel ▾

```
const person = {
  name: 'Max'
};

const secondPerson = person;

person.name = 'Manu';

console.log(secondPerson);
```

Console Run Clear

```
[object Object] {
  name: "Manu"
}

>
```

We can see that this is the case if we changed person.name after having it copied. With that you would expect to print Max here still a person with name Max because we copied person, stored it in secondPerson and thereafter changed the name. However if I clear and run you will see name. Manu here even though I'm printing the secondperson so for secondPerson the name also changed the reason for it is that it just copied the pointer and points to the exact same object in memory as person does. So if we change name on person we automatically change it for secondPerson. Now that's important. Keep in mind and it's the same for arrays. If you copy in quotation marks. An array like this. And you then change an array element. It will all change in the so-called copied array. This will become important in React because it can lead to unexpected behaviors. If you copy objects or arrays like this because you then may manipulate one object in one place in the app and accidentally manipulate another usage of the same object in another place of the app. Therefore we will learn techniques to copy this in an immutable way which means we copy that by really copying the object and not just a pointer

Bin info
just now

JS Bin - JS Bin

jsbin.com/capasenona/edit?js,console

File Add library Share HTML CSS ES6 / Babel Console Output Login or Register Blog 1 Help

ES6 / Babel

```
const person = {
  name: 'Max'
};

const secondPerson = {
  ...person
};

person.name = 'Manu';

console.log(secondPerson);
```

Console Run Clear

```
[object Object] {
  name: "Max"
}

>
```

Therefore we will learn techniques to copy this in an immutable way which means we copy that by really copying the object and not just a pointer for that we can use this spread operator. Now we can simply create a new person object here and spread the person properties. This will pull out the properties and the values of the properties from the object and add it to this newly created object here and we do create a new one with the curly braces. Now if I hit clear and run we still print an object with name Max even though we changed the name to Manu here because now we really created a real copy. This is a technique I will also come back to later in this course. It's just important to realize and to keep in mind that objects and arrays are reference types. If you reassign them you're copying the pointer not the value. Therefore if you want to do this in a real copy way, you will have to create a new object and just copy the properties and not the entire object. That's something very important to keep in mind for this course.

Bin info
just now

JS Bin - JS Bin

jsbin.com/capasenona/edit?js,console

File ▾ Add library Share HTML CSS ES6 / Babel Console Output Login or Register Blog 1 Help

ES6 / Babel ▾

```
const numbers = [1, 2, 3];

const doubleNumArray = numbers.map((num) => {
  return num * 2;
});

console.log(numbers);
console.log(doubleNumArray);
```

Console Run Clear

```
[1, 2, 3]
[2, 4, 6]
>
```

in the map function we have to return the new value we want to turn the old one into so we could return num * 2 and since this is executed on every element here. It will return 2, 4 and 6 and conveniently map all the returns a new array. So a real new array which is then stored in doubleNumArray. So now if I output numbers and thereafter doubleNumArray like this and then I'll hit run you'll see the old one is unchanged but the new one holds double the values and I will explain what these functions do when we use them in the course. I just want to bring them to your attention right now. Explain that they always have this function which gets executed on each element and that they are not next generation javascript but normal javascript actually, be prepared to meet them. I will explain what they do when we see them and always feel free to dive into docs like the Mozilla developer network to learn more about them.

Bin info
just now

Next-Gen JavaScript

A Refresher

With that I want to conclude this module. We've learned a lot about next generation javascript and some important javascript concepts. In general you will meet a lot of the things you'll learn about in this module throughout the course. Don't be confused by it. It's still javascript. Just using some more modern features. That's just how we write React apps. The next module will now start with React. Now I will show you how to quickly create a project set up where we can use all these features while still shipping code that works in older browsers too. With that you're well prepared. Always feel free to go back to this module and have a look at a given feature if you forgot how it worked and you meet it in the course and want to refresh your knowledge. And with that let's dive into React.