

# React

1. Rappels

## **2. Syntaxes ES6+**

3. Premiers pas avec React

4. Les composants

5. Formulaires et AJAX

6. Hooks & React Router

## 2. SYNTAXES ES6+

- Les bases
- POO
- Modules

# ES6 : LET

- remplace "var"
- scopée

```
let i = 0;  
for ( let i = 1; i < tableau.length; i++ ) {  
    console.log( i ); // 1, 2, 3, ...  
}  
console.log( i ); // 0
```

---

Notes :

Fonctionne exactement comme le `var` traditionnellement employé. Cependant, le `let` a la particularité d'être scopé au niveau du bloc de code dans lequel il est déclaré, et pas forcément dans toute la fonction.

# ES6 : CONST

- déclaration de constante
- portée

```
const hello; // Erreur : une constante doit avoir une valeur !
const hello = 'Hello';

if (true) {
  const hello = 'Bonjour';
  hello = 'Hola'; // Erreur : une constante ne peut être modifiée !
  console.log(hello); // "Bonjour"
}

console.log(hello); // "Hello"
```

---

## Notes :

`const` permet de déclarer une constante. De plus, comme le `let`, une variable `const` est scopée au niveau du bloc de code et non pas de la fonction comme le `var`.

# ES6 : CONST ?

```
const ingredients; // 🚫 Erreur : une constante doit avoir une valeur
```

```
const ingredients = ['methanol', 'red phosphorus']; // 👍
```

```
ingredients = ['methanol', 'red phosphorus', 'pseudo']; // 🚫 Erreur :  
// réassignation impossible
```

```
ingredients.push('pseudo'); // 👍
```

# ES6 : TEMPLATES STRINGS

- Nouveau délimiteur de chaîne : `
- Injection d'expressions (variable ou appel de fonction)
- multiline

```
const value = `danger`;
let phrase = `I am the ${ value } !`; // I am the danger !

function myFunction(valeur) {
    return valeur;
}
phrase = `I am the ${ myFunction('danger') } !`; // I am the danger !
```

```
function renderLink( target, cssClass, href ) {
    return `<a href="${ href.toLowerCase() }"
        class="${ cssClass }"
        target="${ target }">
        Cliquez ici !
    </a>`; // multiline !
}
```

# ES6 : DESTRUCTURING

- déclarer des variables au nom d'une propriété d'objet
- ... ou qui correspondent aux valeurs d'un tableau

```
const personnage = { prenom: 'Skyler', nom: 'White' };  
// const prenom = personnage.prenom, nom = personnage.nom;  
const { prenom, nom } = personnage;  
console.log(`Salut ${prenom} ${nom} !`);
```

```
function maFonction( {prenom, nom} ) {  
    return `Salut ${prenom} ${nom} !`;  
}  
maFonction( personnage );
```

```
const tableau = [ 'Walter Jr', 'White' ];  
// const prenom = tableau[0], nom = tableau[1];  
const [ prenom, nom ] = tableau;
```

---

Notes :

Le destructuring permet de faciliter la création et l'assignation de variables à partir issues d'objets ou de tableaux.

<http://codepen.io/kumquats/pen/dXWxzW?editors=0011>

# ES6 : ARROW FUNCTIONS

1.8

- déclaration de fonction simplifiée
- return implicite
- scope préservé

```
// Fonction anonyme en ES5
var add = function( a, b ) {
    return a + b;
}

// et en ES6 (opérateur "fat arrow")
const add = ( a, b ) => a + b;
const square = x => x * x;
// si un seul paramètre, pas besoin de parenthèses
```

---

Notes :

La valeur de `this` dans une arrow function est toujours celui dans lequel elle est déclarée :

```
const character1 = {
  name: 'hank',
  link: document.querySelector( 'a.link' ),
  init: function(){
    this.link.addEventListener( 'click',
      (event)=>{
        event.preventDefault();
        console.log( this.name );
      }
    );
  }
}
character1.init();
```

cf. <https://codepen.io/uidlt/pen/wvGLwmL?editors=1001>



# PROBLÈME : SUPPORT NAVIGATEUR

PROBLÈME : SUPPORT  
NAVIGATEUR

## ECMAScript 2015 (ES6)

Support for the ECMAScript 2015 specification. Features include Promises, Modules, Classes, Template Literals, Arrow Functions, Let and Const, Default Parameters, Generators, Destructuring Assignment, Rest & Spread, Map/Set & WeakMap/WeakSet and many more.

IE	Edge	Firefox	Chrome	Safari	iOS Safari	Opera Mini	Chrome for Android	Android Browser	Samsung Internet
9	88	86	88	13.1	13.7			4.4	12.0
10	89	87	89	14	14.4			4.4.4	13.0
11	90	88	90	14.1	14.5	all	90	90	14.0
		89	91	TP					

✓ ✗ Partial Support

Global: 94.59% + 3.11% = 97.70%

Data from caniuse.com | Embed from caniuse.bitsofco.de

28 May 2021

### Notes :

Le support navigateur est très partiel, notamment sur IE. Il faut noter que chaque feature ES6 dispose de son propre support navigateur de manière indépendante. <http://caniuse.com/#search=es6>

<http://kangax.github.io/compat-table/es6/> Autre tableau détaillé, navigateur par navigateur du support des différentes fonctionnalités ES6.

Pour ES2016 et suivantes voir <http://kangax.github.io/compat-table/es2016plus/> et pour les features encore en cours de normalisation : <http://kangax.github.io/compat-table/esnext/>

**BABEL** 7.10.0 Docs Setup Try it out Videos Blog Search Donate Team GitHub 1.11

GET BABEL HOLIDAY APPAREL


# Babel is a JavaScript compiler.

Use next generation JavaScript, today.

Babel 7 is out! Please read our [announcement](#) and [upgrade guide](#) for more information.

Put in next-gen JavaScript	Get browser-compatible JavaScript out
<pre>let yourTurn = "Type some code in here!";</pre>	<pre>let yourTurn = "Type some code in here!";</pre>

Special Sponsor

 handshake

Decentralized certificate authority and naming

Notes :

BabelJS <https://babeljs.io/>

Babel est un compilateur de code ES6+ en ES5.

Cet outil permet de rendre notre application JS compatible avec les vieux navigateurs.

Le principe est le suivant :

- on code de manière élégante à l'aide des dernières features du langage (ES6+)
- Babel compile le tout en ES5 (compatible avec les anciens navigateurs),
- c'est ce fichier compilé qui sera mis en ligne et servi aux visiteurs

BABEL

DocsSetupTry it outVideosBlogSearchDonateTeam112GitHub

```
1 // Arrow function
2 let square = x => x * x;
3
4 // const
5 const i = square(42);
6 //i = 12; //erreur de compilation (constante)
7
8 // let scopé
9 for ( let i = 0; i < 12 ; i++ ){
10   console.log(i);
11   if (true){
12     let i = 5;
13     console.log(i);
14   }
15 }
16
17 // destructuring
18 o = {
19   a: "a",
20   b: "b",
21 };
22 let {a, b} = o;
23
24 // objet rest spread (stage 3)
25 const params = { lol: 'lol' };
26 const fusion = {
27   pouet: 'pouet',
28   ...params
29 }
30
```

```
1 "use strict";
2
3 // Arrow function
4 var square = function square(x) {
5   return x * x;
6 }; // const
7
8
9 var i = square(42); //i = 12; //erreur de compilation (constante)
10 // let scopé
11
12 for (var _i = 0; _i < 12; _i++) {
13   console.log(_i);
14
15   if (true) {
16     var _i2 = 5;
17     console.log(_i2);
18   }
19 } // destructuring
20
21
22 o = {
23   a: "a",
24   b: "b"
25 };
26 var _o = o,
27   a = _o.a,
28   b = _o.b; // objet rest spread (stage 3)
29
30 var params = {
31   lol: 'lol'
32 };
33 var fusion = {
34   pouet: 'pouet',
35   ...params
36 };
```

Notes :

Babel dispose d'un outil en ligne qui permet de tester la conversion de code ES6+ en ES5. Cet outil a surtout un but pédagogique : montrer comment les syntaxes ES6+ permettent d'avoir un code plus propre et plus lisible que son équivalent en ES5.

Outil de test en ligne du compilateur babel (repl)

babel repl pré-rempli avec des exemples de code ES6