

SEMAINE INTENSIVE REACT



2018 | HETIC

Belle Epoque

- Agence spécialisée en développement
- Nous sommes ici car on utilise React au quotidien
- HETIC promo P2013

Vous pouvez nous contacter
et postuler hello@agence-belle-epoque.fr



Déroulement

- Théorie + pratique
- Démos et exercices pour du concret
- Interaction / poser des questions
- Mélangez-vous

Plan

- Pourquoi React
- Rappel ES6
- Installation
- Props & State
- React Lifecycle
- CSS
- React Router
- Redux
- Animations
- React avancé en fonction du temps : SSR, tests...
- Derniers jours : rappels + QCM

Pourquoi React

Commençons par la base



L'essentiel

- Librairie javascript pour créer des interfaces graphiques
- Rend des vues et répond à des événements
- Le V de MVC
- Open source depuis 2013
- Cible essentiellement des grosses applications JS avec des données qui changent dans le temps

Quelques grands principes

- Approche composant
- Virtual DOM
- Simple

Demo Performance

Une application de base pour tester le taux de repaint des frameworks JS.

@ryanflorence

Vous pouvez le tester sur <https://mathieuancelin.github.io/js-repaint-perfs/>

“

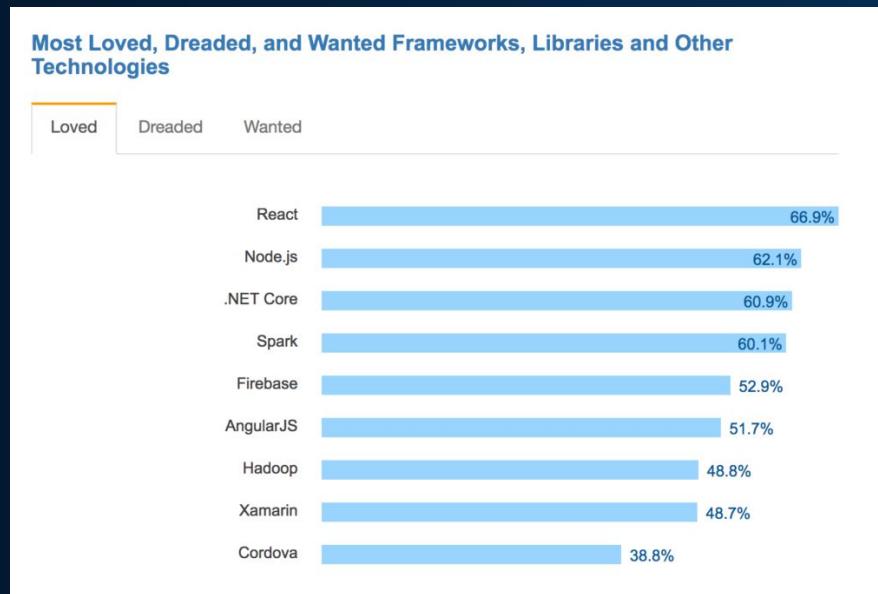
First, software ate the world, the web ate software, and JavaScript ate the web. In 2018, React is eating JavaScript.

Eric Elliott



La librairie Javascript la plus plébiscitée

Enquête annuelle réalisée par le site StackOverflow en 2017



Les meilleures utilisent React



NETFLIX

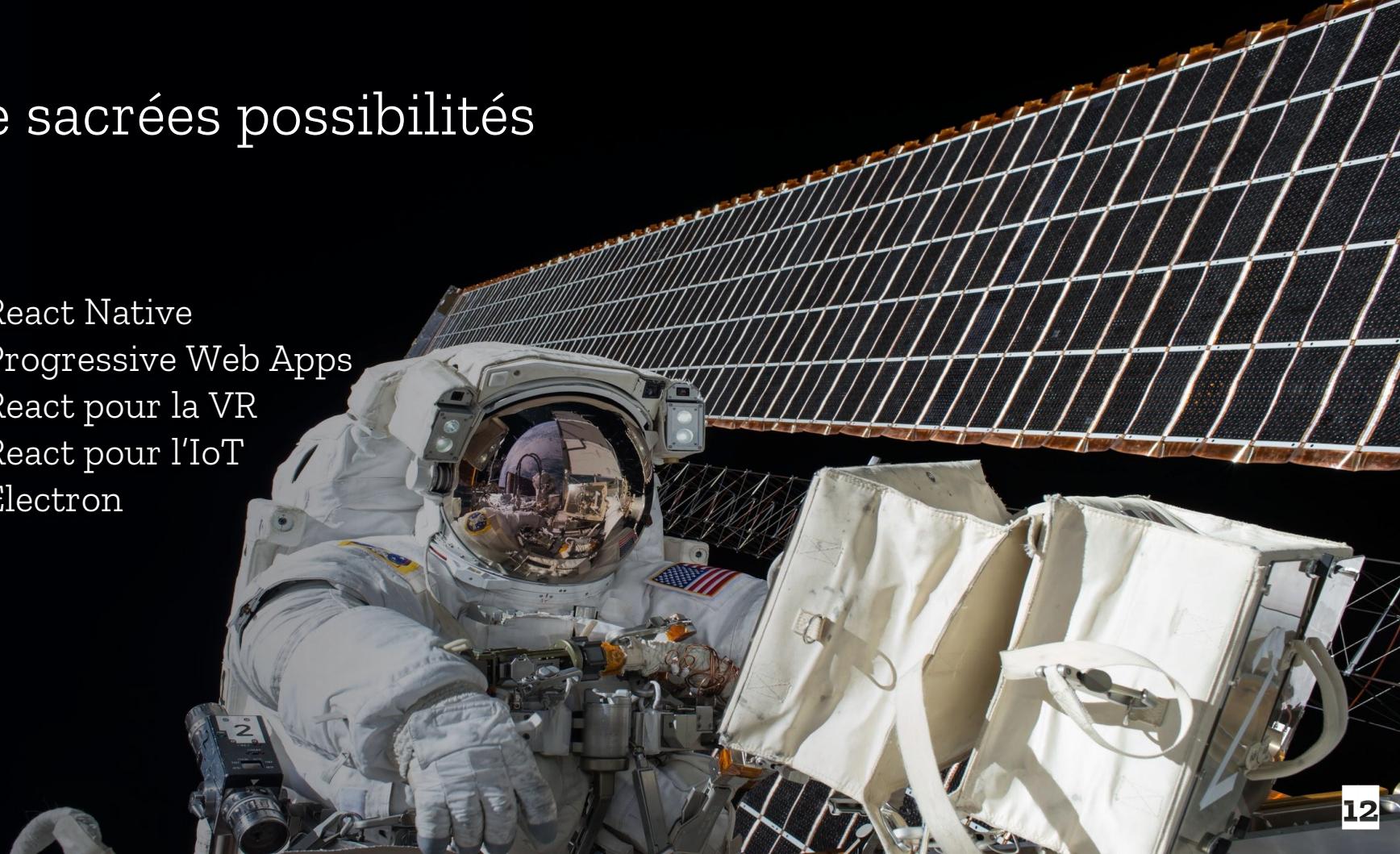


ebay

PayPal

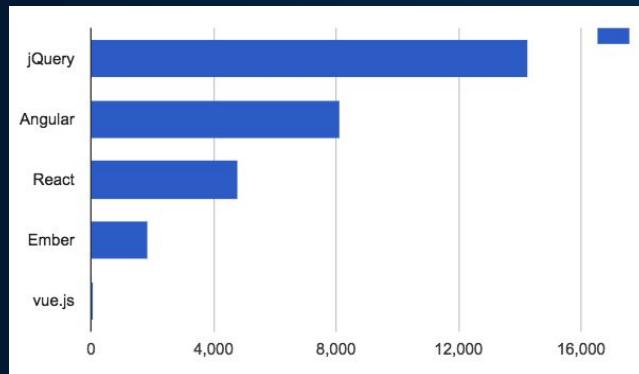
De sacrées possibilités

- React Native
- Progressive Web Apps
- React pour la VR
- React pour l'IoT
- Electron

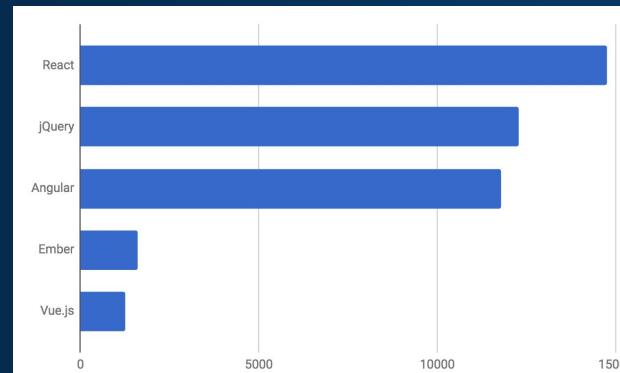


Vous êtes recherchés

Avant ☹



Aujourd'hui React détrône jQuery



Une multitude de modules et d'outils

- Redux
- Redux-Saga
- React Router
- Material UI
- Storybook
- Lodash
- Babel
- Webpack
- Node & Express
- RxJS
- Reselect
- Immutable
- Jest



Rappel ES6

Les bases



ES6 Pourquoi ?

- Plus facile
- Plus puissant
- Plus fiable
- Plus performant



Babel

Transpile ES6, ES7 en ES5

Intégré avec l'univers : Grunt, Gulp, Webpack, Node, React, IDE...

Const ou Let :

telle est la question

- Donnez du sens à vos variables
- N'utilisez plus le mot-clé "var"

```
const maConstante = 'je ne change jamais';  
  
let maVariable = 'je peux changer de valeur';  
maVariable = 'comme ceci';
```

Littéraux

- Simple quote 'pour simple texte'
- Double quote avec attr="Attribut" de balise non dynamique
- Backtick ` avec des \${variables} et des textes sur plusieurs lignes`

```
const maConstante = 'deux';
console.log(`ligne de texte 1
ligne de texte ${maConstante}`);

"ligne de texte 1
ligne de texte deux"
```

The Spread Syntax

```
const mid = [3, 4];
const arr = [1, 2, mid, 5, 6];

console.log(arr);
```

[1, 2, [3, 4], 5, 6]

```
const mid = [3, 4];
const arr = [1, 2, ...mid, 5, 6];

console.log(arr);
```

[1, 2, 3, 4, 5, 6]

The Spread Syntax Object

```
const defaults = { first: 'John', last: 'Doe', age: 42 };
let trainer = { last: 'Smith', age: 35 };
trainer = { ...defaults, ...trainer, age: 36 };

// => { fist: 'John', last: 'Smith', age: 36 };
```

React :

```
<TodoItem key={todo.id} todo={todo} {...trainer} />
```

Destructuring

```
function printBasicInfo({firstName, secondName, profession}) {  
    console.log(firstName + ' ' + secondName + ' - ' + profession)  
}  
  
var person = {  
    firstName: 'John',  
    secondName: 'Smith',  
    age: 33,  
    children: 3,  
    profession: 'teacher'  
}  
  
printBasicInfo(person);
```

Destructuring 2

```
const { filter: selectedFilter, onShow } = this.props;
```

```
=
```

```
Const onShow = this.props.onShow;  
Const selectedFilter = this.props.filter;
```

Arrow function

```
const getStuffAwesome = ({ id, name, force, verbose }) => {  
  ...do stuff  
}
```

Mot-clé "this" devient accessible sans binder

```
<TodoTextInput text={todo.text}  
  editing={(text) => this.handleSave(todo.id, text)} />
```

Export / Import

```
export function addTodo () {  
  return { type: types.ADD_TODO,  
text }  
}  
  
export const ADD_TODO =  
'ADD_TODO';  
  
export default Footer;
```

```
// Import intégral dans une "namespace"  
import * as types from  
'../constants/ActionsTypes';  
  
// Ou juste un des deux modes :  
import Footer from './Footer';  
import { ADD_TODO } from  
'../contants/TodoFilters';
```

Classes

```
class TodoItem extends Component {  
  constructor(props, context) {  
    super(props, context);  
    this.state = {  
      editing: false  
    };  
  }  
  
  handleClick(e) {  
    this.setState({ editing: true });  
  }  
  
  render() {  
    return (  
      <div onClick={(e) =>  
        this.handleClick(e)}>  
        {this.state.editing &&  
          <p>Editing est à true</p>  
        }  
        {!this.state.editing &&  
          <p>Editing est à false</p>  
        }  
      </div>  
    );  
  }  
}
```

Async / Await

- Créer une fonction asynchrone grâce au mot-clé "async"
- Attendre une réponse grâce au mot-clé "await"
- Faire moins de Callbacks et de Promesses
- Une compréhension plus simple des appels : fonction avec un aspect procédurale

Async / Await

```
function sendEmails(query) {
  const usersP = getUsers(query);
  // On récupère le champ "email" de tous les utilisateurs
  const emailsP = usersP.then(users =>
    users.map(u => u.email));
  // Pour chaque email...
  const sentP = emailsP.then(emails =>
    emails.map(email => {
      // ... on envoie un mail
      return sendMail(email, "Bonne fête");
    })
  );
  // On attend que tous les envois soient résolus
  return Promise.all(sentP);
}

sendEmails({ firstName: "Edouard" })
  .then(() => console.log("OK"))
  .catch(() => console.error("FAIL"));
```



```
async function sendEmails(query) {
  const users = await getUsers(query);
  const emails = users.map(u => u.email);
  const sentP = emails.map(email =>
    sendMail(email, "Bonne fête"));
  return await Promise.all(sentP);
}

async function main() {
  try {
    await sendEmails({ firstName: "Edouard" });
    console.log("OK");
  } catch (e) {
    console.error("FAIL");
  }
}

main();
```

Installation

Passons au concret...



Npm & node

<https://nodejs.org/en/>

Create React App

Official. No setup. Minimal.

```
npm install -g create-react-app
```

```
npx create-react-app my-app && cd my-app
```



```
npm start
```

Architecture

```
my-app
├── README.md
├── node_modules
├── package.json
├── .gitignore
└── public
    ├── favicon.ico
    ├── index.html
    └── manifest.json
└── src
    ├── App.css
    ├── App.js
    ├── App.test.js
    ├── index.css
    ├── index.js
    └── logo.svg
        └── registerServiceWorker.js
```

ESLint config

<https://bitbucket.org/snippets/lebrenn/8eara8>

<https://www.npmjs.com/package/eslint-config-prettier>



An aerial photograph of a dense forest. The trees are mostly evergreen, with varying shades of green and some bare branches. A dark, narrow path or road winds its way through the center of the forest. The overall scene is dark and moody.

Props & State

Props & state

this.props : contient toutes les propriétés disponibles d'un composant. Elles sont modifiables par le composant parent

this.state : contient tous les états disponibles d'un composant. Ils peuvent changer au sein du composant

```
handleClick(e) {  
  e.preventDefault();  
  this.setState({ monEtat1: 'Nouvelle valeur' });  
}  
  
render() {  
  return (  
    <button onClick={(e) => this.handleClick(e)}>  
      <MonComposantEnfant  
        maProp1={this.state.monEtat1} />  
    </button>  
  );  
}
```

Props & state

- L'état qui représente l'état interne d'une instance de composants
 - L'état est mutable, idéalement seulement par le composant lui-même
- C'est le changement de l'état qui lancera la mise à jour de la vue. Le double binding est supporté mais non encouragé

```
this.setState({  
  tasks: []  
});
```

```
This.state.tasks.map(item => <Task key={task.id} task={item} />);
```

Using State Correctly

```
// Wrong
this.state.comment = 'Hello';

// Correct
this.setState({comment: 'Hello'});

// Correct
this.setState((prevState, props) => ({
  counter: prevState.counter + props.increment
}));
```

Demo

Création de notre premier module : "compteur de clic"

1. Créer un nouveau composant de class
2. Définir un constructeur (état initial de notre composant)
3. Ajouter une fonction de rendu avec un bouton
4. Ajouter une action qui sera appelée à chaque clic sur le bouton et qui incrémentera la valeur de notre compteur

Exercice

Number

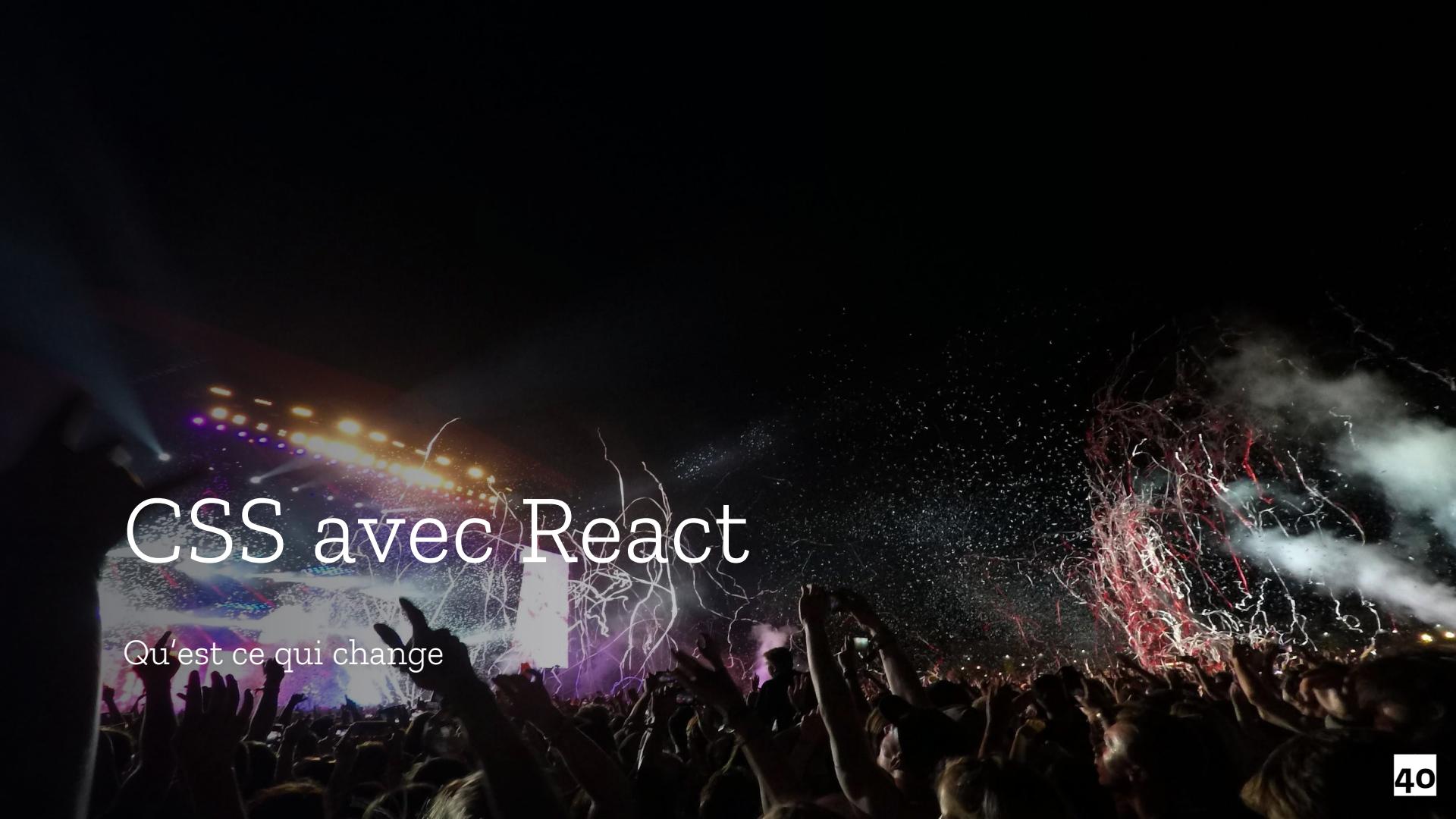
Number

Result

Créer une calculatrice simple avec les opérateurs de bases :

+ - * /





CSS avec React

Qu'est ce qui change

CSS

Vous pouvez utiliser une feuille de style classique

Ou votre préprocesseur préféré Sass, LESS, Stylus...

Le problème

Le style n'est pas localisé au niveau du composant,
il sera attaché au DOM global

- Régressions / collisions
- Difficile à maintenir
- Difficile de collaborer à plusieurs
- L'important rentre en jeu pour encore plus de problèmes



BEM

- Meilleure maîtrise de la priorisation
- Facilement modifiable
- Court et compréhensible

```
.Block
.Block--theModifier
.Block-theElement
.Block-theElement--theModifier
```

BEM

```
h1 {  
  color: $textColor;  
  
  img {  
    border: 1px solid black;  
  }  
}  
}
```

se transforme en :

```
.Title {  
  color: $textColor;  
}  
.Title-icon {  
  border: 1px solid black;  
}
```

CSS Modules

Limite le scop d'utilisation à chaque module

```
import styles from "./style.css";
return <h1 className={styles.styleName}></h1>;
```



Installation

```
npm run eject
```

Change config/webpack.config.dev.js

```
// before
{
  loader: require.resolve('css-loader'),
  options: {
    importLoaders: 1,
  },
},
```

```
// after
{
  loader: require.resolve('css-loader')
  ,
  options: {
    importLoaders: 1,
    modules: true,
    localIdentName:
      "[name]__[local]__[hash:base
64:5]"
  },
},
```

Installation

```
// before
{
  loader:
require.resolve('css-loader'),
options: {
  importLoaders: 1,
  minimize: true,
  sourceMap: true,
},
},
// after
{
  loader:
require.resolve('css-loader'),
options: {
  importLoaders: 1,
  modules: true,
  minimize: true,
  sourceMap: true,
},
},
```

Quelques changements

1. `import SomeComponent from './SomeComponent.css' .`
2. Remplacer les className `className={styles.myClass}`.
3. CSS Modules n'autorise pas les tirets (" - ")

CSS Modules

Génération automatique des noms de classes CSS

Une fois compilé, ce code générera :

```
.styleName {  
  color: $textColor;  
}  
  
import styles from "./style.css";  
`<h1 className=${styles.styleName}></h1>`;
```

```
.styleName__abc5436 {  
  color: #333333;  
}  
  
<h1 class="styleName__abc5436"></h1>
```

CSS Modules

Composition des styles

```
.titleColor {  
  color: #333333;  
}  
  
.bigTitle {  
  composes: titleColor;  
  font-size: 24px;  
}  
  
import styles from "./style.css";  
`<h1 className=${styles.bigTitle}></h1>`;
```

Va générer :

```
.titleColor__abc5436 {  
  color: #333333;  
}  
  
.bigTitle__def6547 {  
  font-size: 24px;  
}  
  
<h1 class="titleColor__abc5436  
bigTitle__def6547"></h1>
```



Global CSS

A utiliser avec parcimonie

```
:global .foo {  
}  
}
```

PostCSS

- Simplicité
- Modulable
- S'utilise avec des pré-processeurs, css modules..
- De nombreux modules :
 - Autoprefixer
 - CSS next
 - Nested
 - Variables
 - ...

An aerial photograph of a dense forest. The trees are mostly evergreen, with varying shades of green and some bare branches. A dark, narrow path or road winds its way through the center of the forest. The overall scene is dark and moody.

React Lifecycle

Les méthodes



React fournit des méthodes ou "hooks". Ces méthodes sont appelées pendant le cycle de vie d'un composant

- Permet de mettre à jour l'UI et l'état de l'application

Les plus utilisés

- constructor()
- render()
- componentDidMount()

Component Lifecycle

Initial Render	Props Change	State Change	Component Unmount
Default props	componentWillReceiveProps()		componentWillUnmount()
constructor()	shouldComponentUpdate()	shouldComponentUpdate()	
componentWillMount()	componentWillUpdate()	componentWillUpdate()	
render()	render()	render()	
componentDidMount()	componentDidUpdate()	componentDidUpdate()	



render()

```
class Clock extends React.Component {  
  constructor(props) {  
    super(props);  
    this.state = { date: new Date() };  
  }  
  
  render() {  
    return (  
      <div>  
        <h1>Hello, world!</h1>  
        <h2>It is {this.state.date.toLocaleTimeString()}.</h2>  
      </div>  
    );  
  }  
}
```

componentDidMount()

```
componentDidMount() {  
  this.timerID = setInterval(  
    () => this.tick(),  
    1000  
  );  
}  
}
```

componentWillUnmount()

```
componentWillUnmount() {  
  clearInterval(this.timerID);  
}  
  
```

Longueur / Largeur

```
constructor(props) {
  super(props);
  this.state = {
    w: null,
    h: null
  };
}

componentWillMount() {
  // Force to get first document size.
  this.onResizeHandler();
}

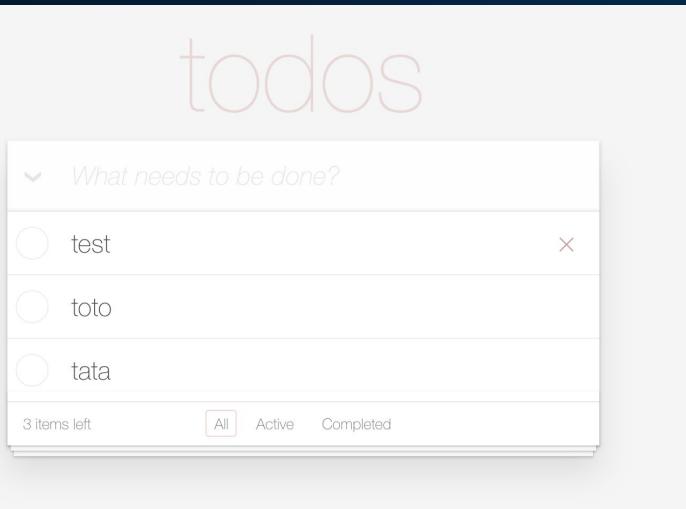
// when the component is added to the DOM...
componentDidMount() {
  window.addEventListener("resize", () =>
this.onResizeHandler());
}
```

```
// when the component is removed from the DOM...
componentWillUnmount() {
  window.removeEventListener("resize", () =>
this.onResizeHandler());
}

onResizeHandler() {
  const w = document.documentElement.clientWidth;
  const h = document.documentElement.clientHeight;
  console.log("The window has been resized!", {
    w,
    h
  });
  this.setState({ w, h });
}

render() {
  const { w, h } = this.state;
  return (
    <ul>
      <li>Width: {w}</li>
      <li>Height: {h}</li>
    </ul>
  );
}
```

Demo



Créer une TodoList avec :

- 1 formulaire d'ajout de tâche dont le champ se vide une fois l'insertion effectuée
- 1 compteur de tâches
- 1 listing des tâches
- La possibilité de supprimer une tâche
- La possibilité de barrer une tâche au clic sur son label
- La possibilité de supprimer l'ensemble des tâches depuis le bouton "Reset task manager"