# Introduction à React.js

Said Gounane
2019/2020

## Introduction

- Une librairie pour créer des interfaces utilisateur.
- Créer par Facebook et Instagram (mars 2013)
- Suivi par React native pour les téléphones mobiles

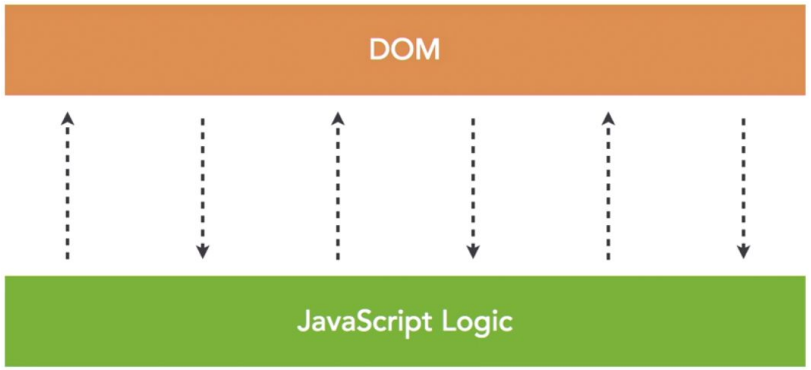


## DOM diffing

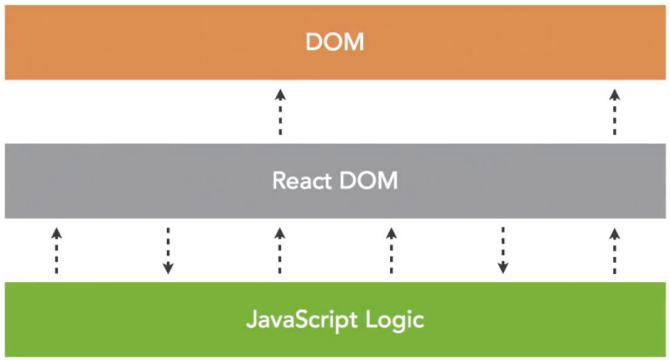
- Compare la UI actuel avec la nouvelle
- Effectue uniquement les modification nécessaires
- Une comparaison des objets javascript
- Plus rapide que les modification directes sur le DOM



## Sans ReactJS



## Avec ReactJS (Virtual DOM)






## ReactJS: Example

```
const app=React.createElement('h1',
    {id: 'titre', className:'header'},
    'Bonjour!!');

ReactDOM.render(app,
    document.getElementById('root'));
```

```
<head>
    <meta charset="utf-8">
    <title>Ma premiere React app</title>
    <script type="text/javascript" src='../react.js'></script>
    <script type="text/javascript" src='../react-dom.js'></script>
</head>
<body>
<div id='root'></div>
<script type="text/javascript" src='index.js'></script>
</body>
</html>
```



## ReactJS: Example

```
const {createElement} = React;
const {render} = ReactDOM;

const app=createElement('h1',
    {id: 'titre', className:'header'},
    'Bonjour!!');
render(app, document.getElementById('root'));
```

```
<head>
    <meta charset="utf-8">
    <title>Ma premiere React app</title>
    <script type="text/javascript" src='../react.js'></script>
    <script type="text/javascript" src='../react-dom.js'></script>
</head>
<body>
<div id='root'></div>
<script type="text/javascript" src='index.js'></script>
</body>
</html>
```



## ReactJS: Example

```
const {createElement} = React;
const {render} = ReactDOM;

const stl={
    backgroundColor: 'orange',
    color: 'white',
    fontFamily: 'Verdana'
};

const app=createElement('h1',
    {id: 'titre', className:'header', style: stl},
    'Bonjour!!');
render(app, document.getElementById('root'));
```

```
</head>
<body>
<div id='root'></div>
<script type="text/javascript" src='index.js'></script>
</body>
</html>
```

# JSX

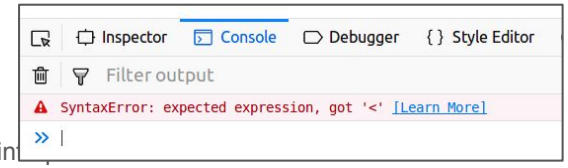
- Du “***HTML***” dans ***javascript***

```
const {render} = ReactDOM;

const elmt = <h1> Bonjour!! </h1>;
render(elmt , document.getElementById('root'));

```

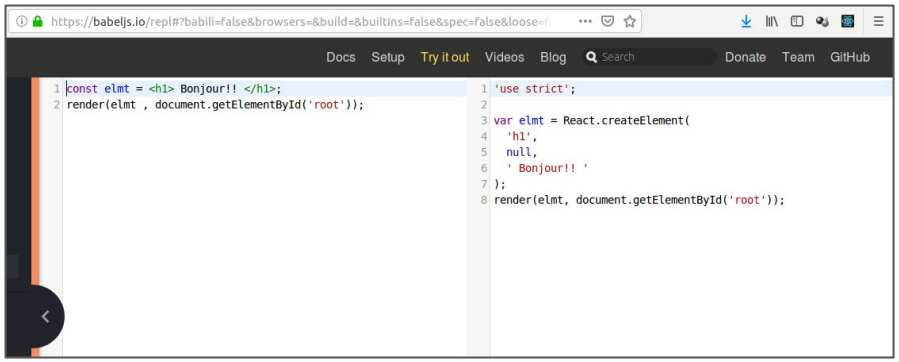
⇒ ***Erreur***




Besoin d’un in



# JSX: Babel






# JSX: Babel

index.js

```
const {render} = ReactDOM;

const elmt = <h1> Bonjour!! </h1>;
render(elmt , document.getElementById('root'));

```

index.html

```
<head>
    <meta charset="utf-8">
    <title>First React app</title>
    <script type="text/javascript" src='../react.js'></script>
    <script type="text/javascript" src='../react-dom.js'></script>
    <script type="text/javascript" src='
    https://cdnjs.cloudflare.com/ajax/libs/
    babel-core/5.8.38/browser.js'></script>
</head>
<body>
<div id='root'></div>
<script type="text/babel" src='index.js'></script>
</body>
</html>
```



# JSX: Babel

index.js

```
const {render} = ReactDOM;

const stl={
    backgroundColor: 'orange',
    color: 'white',
    fontFamily: 'Verdana'
};

const elmt = <h1 id='titre'
                 className='header'
                 style={stl}>
                 Bonjour!!
             </h1>;

render(elmt , document.getElementById('root'));
```

index.html

```
</head>
<body>
<div id='root'></div>
<script type="text/babel" src='index.js'></script>
</body>
</html>
```

## JSX: Babel

- Avec cette méthode, la transformation du jsx en javascript se fait dans le browser.
- Méthode utilisée uniquement pour les tests.
- Pour la production il faut utiliser le fichier javascript préalablement transformé

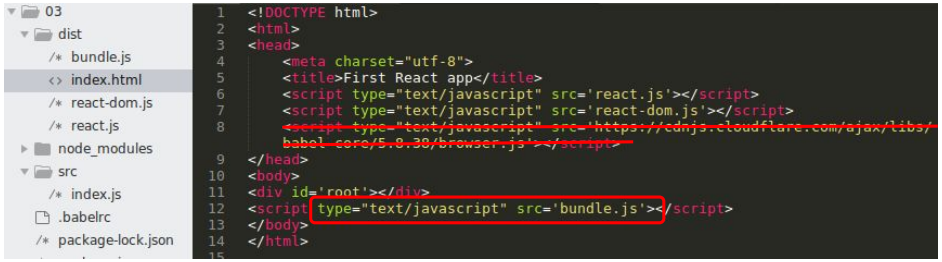


## Babel: installation

- $ ***npm init***
- $***npm install --save-dev babel-cli*** ou ***sudo npm install -g babel-cli***
- $***npm install --save-dev babel-preset-latest babel-preset-react babel-preset-stage-0***
- Dans le fichier ***.babelrc***: ***{ 'presets': ['latest', 'react', 'stage-0'] }***



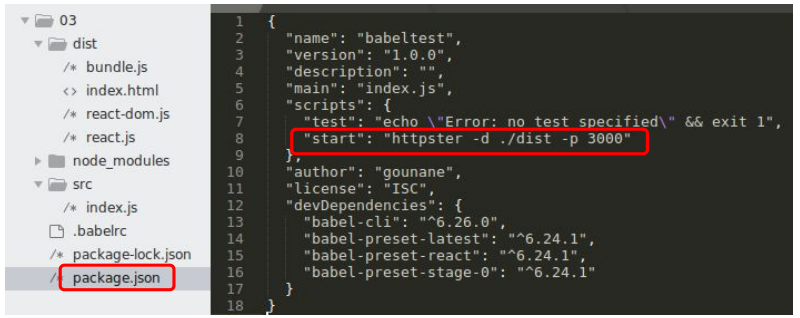
## JSX: Babel




- ***$./node_modules/.bin/babel ./src/index.js --out-file ./dist/bundle.js*** ou
- $***babel ./src/index.js --out-file ./dist/bundle.js***



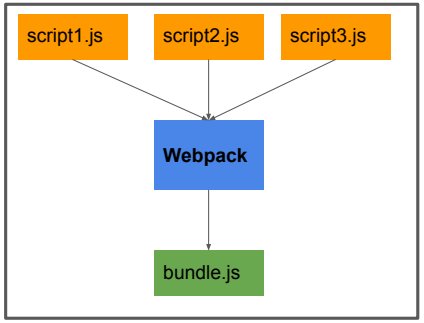
## JSX: Babel




- $***npm start***

## Webpack

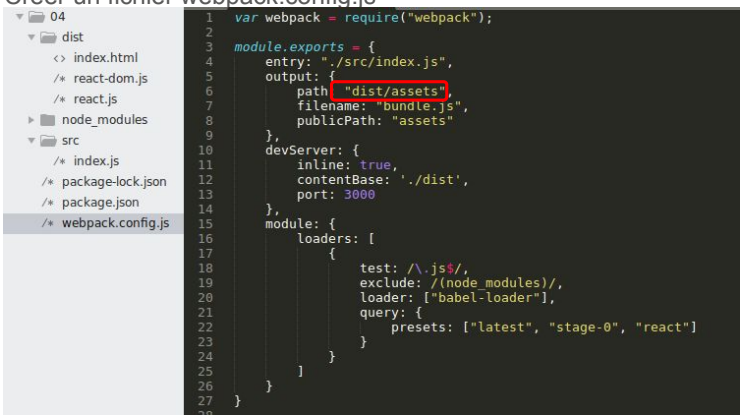
- Est un module bundler
- Création des fichiers statiques
- Automatisation des tâches






## Webpack

- Créer un fichier webpack.config.js



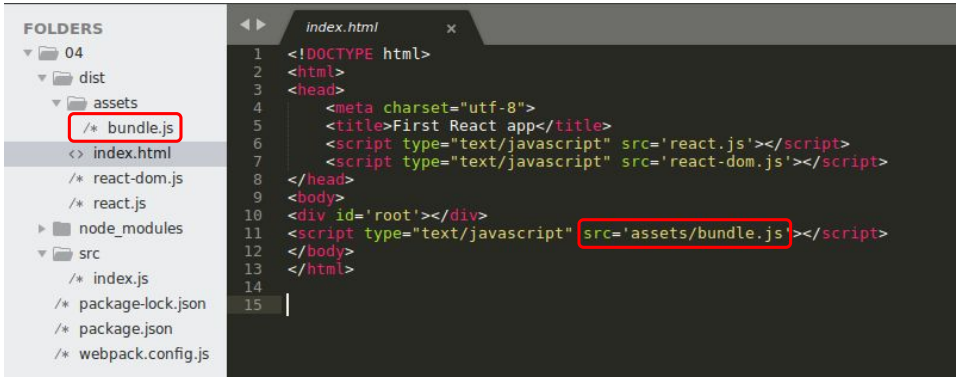



## Webpack

- $ ***npm install webpack babel-loader webpack-dev-server --save-dev***
- $***./node_modules/.bin/webpack***
- Dans le fichier ***package.json*** modifier la ligne “start”:
  - ***“start”: “./node_modules/.bin/webpack-dev-server”***
- ***$ npm start***




## Webpack



## Webpack

- $ *npm install --save react react-dom*
- $*./node_modules/.bin/webpack*
- *$ npm start*

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>First React app</title>
    <script type="text/javascript" src='react.js'></script>
    <script type="text/javascript" src='react-dom.js'></script>
</head>
<body>
<div id='root'></div>
<script type="text/javascript" src='assets/bundle.js'></script>
</body>
</html>
```

```
const {render} = ReactDOM;
import React from "react";
import ReactDOM from "react-dom";
const stl={
    backgroundColor: 'orange',
    color: 'white',
    fontFamily: 'Verdana'
};

const elmt = <h1 id='titre'
                 className='header'
                 style={stl}>
                 Bonjour!!
             </h1>;

ReactDOM.render(elmt , document.getElementById('root'));
```



## Créer une application react sans configuration

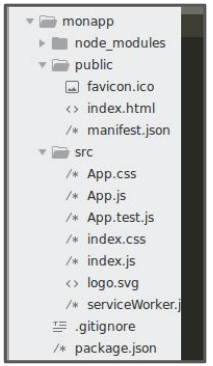
Pour créer une application react, sans passer par les étapes précédentes, la commande ***create-react-app*** se charge de toutes les configurations nécessaires.

Pour installer create-react-app:

$***sudo npm install -g create-react-app***

Pour créer une application "***monapp***":

$ ***create-react-app monapp***






## Lancer une application react

$ ***cd monapp***

$ ***npm start***



## Principe

Index.html

```
  <body>
    <noscript>
      You need to enable JavaScript to run this app.
    </noscript>
    <div id="root"> </div>
  </body>
</html>
```

index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';

ReactDOM.render(<App />, document.getElementById('root'));
```

## React Component

1- Étendre la classe Component

2- Redefinir render()

3- JSX

4- export pour l'utiliser dans un autre fichier js

```
import React, { Component } from 'react';
import logo from './logo.svg';
import './App.css';

class App extends Component {
  render() {
    const titre="MonApp React";
    return (
      <div className="App">
            <h1>{titre}</h1>
            <p>Voice ma premiere application ReactJS</p>
      </div>
    );
  }
}

export default App;
```



## Les Props

Pour passer des arguments à un ***Component*** on utilise les ***props***

***<Moncomponent arg1="val1" arg2="val2" arg3={js expression} />***

Pour accéder à ces props on a accès à l'objet props dans la méthode render de moncomponent.js

- ***this.props.arg1***
- ***this.props.arg2***
- ***this.props.arg3***



## Les props

app.js

```
class App extends Component {
  render() {
    const titre=this.props.titre;
    return (
      <div className="App">
            <h1>{titre}</h1>
            <p>Une premiere application ReactJS par {this.props.nom}</p>
      </div>
    );
  }
}
```

index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';

ReactDOM.render(<App titre="ReactJS" nom="Yasser" />, document.getElementById('root'));
```



## Le state

1. L'objet ***state*** contient des données utilisées par le ***Component***
2. Les changement dans cet objet seront répercutés automatiquement sur ce ***component***.
3. La modification de l'objet state ne se fait que par la méthode ***setStat(newstate)***;

```
import React, { Component } from 'react';
import './style/bootstrap.min.css';
import './style/App.css';
import {textVal} from "./textVal";

class App extends Component {
  state={
    text: textVal
  }
  render() {
    const titre=this.props.titre;
    return (
      <div className="container">
            <div className="titre">
              <h1>{titre}</h1>
              <p>Par: {this.props.nom}</p>
            </div>
            <div className="row">
              <div className="col-sm-6">
                <textarea rows="20" className="form-control"
                value={this.state.text}></textarea>
              </div>
              <div className="col-sm-6">
                <h1>Resultats</h1>
              </div>
            </div>
      </div>
    );
  }
}
```

## Le state

1. Lors de l'édition dans le textArea
2. Les modification sont automatiquement apporté au div

```
class App extends Component {
  state={
    text: ""
  }
  handleEdite=(e)=>{
    const t=e.target.value;
    const newState={
      text: t
    };
    this.setState(newState);
  }
  render() {
    const titre=this.props.titre;
    return (
      <div className="container">
        <div className="titre">
          <h1>{titre}</h1>
          <p>Par: {this.props.nom}</p>
        </div>
        <div className="row">
          <div className="col-sm-6">
            <textarea rows="20" className="form-control"
            onChange={(e)=> this.handleEdite(e)}></textarea>
          </div>
          <div className="col-sm-6">
            <h1>{this.state.text}</h1>
```



## Le cycle de vie d'un Component

- Mounting
- Updating
- Unmounting



## Le cycle de vie d'un Component

- Mounting
  - constructor()
  - componentWilmount()
  - render()
  - ComponentDidMount()



## Le cycle de vie d'un Component

- Updating
  - getDerivedStateFromProps()
  - shouldComponentUpdate()
  - render()
  - getSnapshotBeforeUpdate()
  - componentDidUpdate()

## Le cycle de vie d’un Component

- Unmounting
  - componentWillUnmount()



## Exercice

Ecrire une application **React** pour :

1. Générer une page contenant une  liste d’utilisateurs avec des images de profile issus de (https://robohash.org/. Les détails des utilisateur est préalablement stocké dans un Array d’objet ***User={firstName, lastName, email, img}***
2. Un bouton pour changer la couleur des cartes
3. Une zone de recherche pour filtrer ces utilisateur par nom.

Robohash (https://robohash.org/) est un service Web simple qui permet de fournir facilement des images uniques pour n'importe quel texte.




# React Router



## Introduction

- React Router est une librairie qui affiche vos composants en fonction de l'URL de votre navigateur.
- Indispensable pour faire une SPA et que vous voulez que l'utilisateur puisse croire que c'est un site normal qu'il peut partager, bookmarker, etc.

## React-router

- Il existe deux version de react-router
  - React-router-dom : Pour l'utilisation dans un browser
  - React-router-native : pour react native
- Dans les diapos suivants on traitera react-router-dom
- Pour installer react-router-dom
  - $npm install react-router-dom
- Ce module comporte plusieurs composantes
  - BrouserRouter
  - Routes
  - Route
  - Link
  - ….



## Configuration du Routeur

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import {BrowserRouter} from "react-router-dom"
import App from './App';
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
 <React.StrictMode>
   <BrowserRouter>
   <App />
   </BrowserRouter>
 </React.StrictMode>
);
```



## Définition des routes

```
import {Routes, Route} from "react-router-dom"
function App() {
 return(
   <div className="App">
     <Nvbar/>
     <Routes>
       <Route path="/" element={<Home/>}/>          /* http://localhost:3000/home */
       <Route path="/articles" element={<About/>}/> /* http://localhost:3000/about */
       <Route path="/*" element={<Home/>}/>
     </Routes>
   </div>
 );
}
```



## Prise en charge de la navigation

```
import { Link } from "react-router-dom"
export default function Navbar(){
   return(
       <div>
           <Link to="/"><button>Home</button></Link>
           <Link to="/articles"><button>About</button></Link>
           <Link to="/contact"><button>Contact</button></Link>
       </div>
   )
}
```

## Les routes dynamiques

```
import {Routes, Route} from "react-router-dom"
function App() {
 return(
   <div className="App">
     <Nvbar/>
     <Routes>
       <Route path="/" element={<Home/>}/>        /* http://localhost:3000/home */
       <Route path="/articles" element={<About/>}/> /* http://localhost:3000/about */
       <Route path="/articles/:id" element={<Article/>}/>
     </Routes>
   </div>
 );
}
```



## Les routes dynamiques

```
import { useParams } from "react-router"
function Article(){
   let {id}=useParames()
   return(
        <>
            <h1>Titre id:{id}</h1>
        </>
   )
}
```



## La priorité des routes

```
import {Routes, Route} from "react-router-dom"
function App() {
 return(
   <div className="App">
     <Nvbar/>
     <Routes>
       <Route path="/" element={<Home/>}/>           /* http://localhost:3000/home */
       <Route path="/articles" element={<Articles/>}/> /* http://localhost:3000/about*/
       <Route path="/articles/:id" element={<Article/>}/>
      <Route path="/articles/add" element={<AddArt/>}/> /*plus prioritaire que :id*/
     </Routes>
   </div>
 );}
```



## Les routes imbriquées

```
function App() {
 return(
   <div className="App">
     <Nvbar/>
     <Routes>
       <Route path="/" element={<Home/>}/>
       <Route path="/articles">
          <Route index element={<Articles/>}/>
          <Route path=":id" element={<Article/>}/>
          <Route path="add" element={<AddArt/>}/>
       </Route>
       <Route path="/*" element={<PageNotFound/>}/>
     </Routes>
   </div>
 );}
```

## Outlet

```jsx
function App() {
 return(
   <div className="App">
     <Nvbar/>
     <Routes>
          <Route path="/" element={<Home/>}/>
          <Route path="/articles" element={<ArticleLayout/>}>
               <Route index element={<Articles/>}/>
               <Route path=":id" element={<Article/>}/>
               <Route path="add" element={<AddArt/>}/>
          </Route>
          <Route path="/*" element={<PageNotFound/>}/>
     </Routes>
   </div>
 );}
```

```jsx
import { Outlet } from
"react-router-dom"
function ArticleLayout(){
   return(
        <>
        <h1>Articles</h1>
        <hr/>
        <Outlet/>
        </>
   )
}
```



## Outlet context

```jsx
function App() {
 return(
   <div className="App">
     <Nvbar/>
     <Routes>
          <Route path="/" element={<Home/>}/>
          <Route path="/articles" element={<ArticleLayout/>}>
               <Route index element={<Articles/>}/>
               <Route path=":id" element={<Article/>}/>
               <Route path="add" element={<AddArt/>}/>
          </Route>
          <Route path="/*" element={<PageNotFound/>}/>
     </Routes>
   </div>
 );}
```

```jsx
import { Outlet } from "react-router-dom"
function ArticleLayout(){
 return(
   <>
    <h1>Articles</h1><hr/>
    <Outlet context={{key:"value"}}/>
   </>
 )}
```

```jsx
import { useParams, useOutletContext} from "react-router"
function Article(){
   let {id}=useParames()
   let context=useOutletContext()
   return(
       <>
           <h1>Titre id:{id}</h1>
           <h1>Context:{context.key}</h1
       </>
   )
```



## useRoute Hook

```jsx
import { Route, Routes, useRoutes } from "react-router-dom"
export function App() {
 const element = useRoutes([
   {
     path: "/",
     element: <Home />
   },
   {
     path: "/articles",
     children: [
       { index: true, element: <ArticlesLayout /> },
       { path: ":id", element: <Articles /> },
        { path: "add", element: <AddArt /> }

     ]
   }
 ])
```