ReactJS

S.Gounane
ISIL 2018-2019

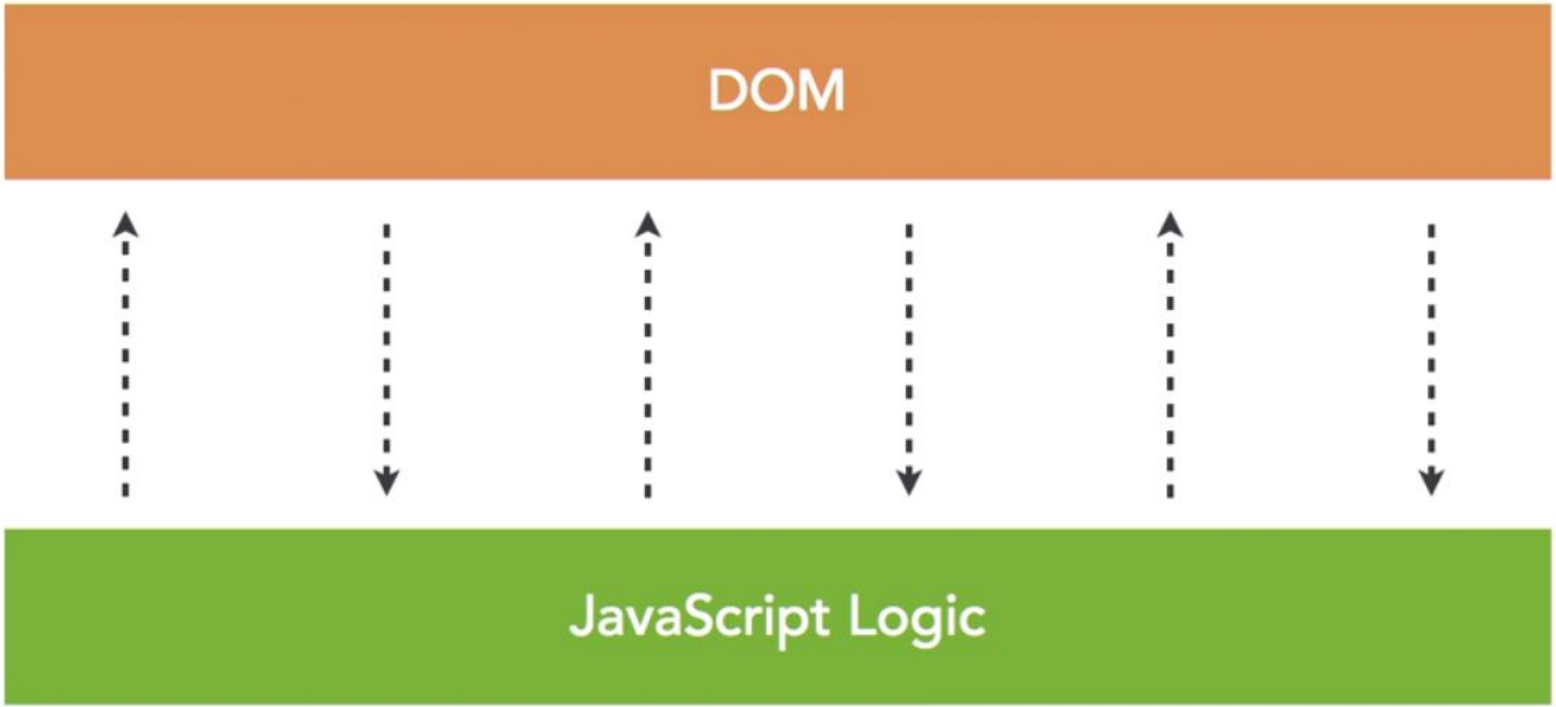
Introduction

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

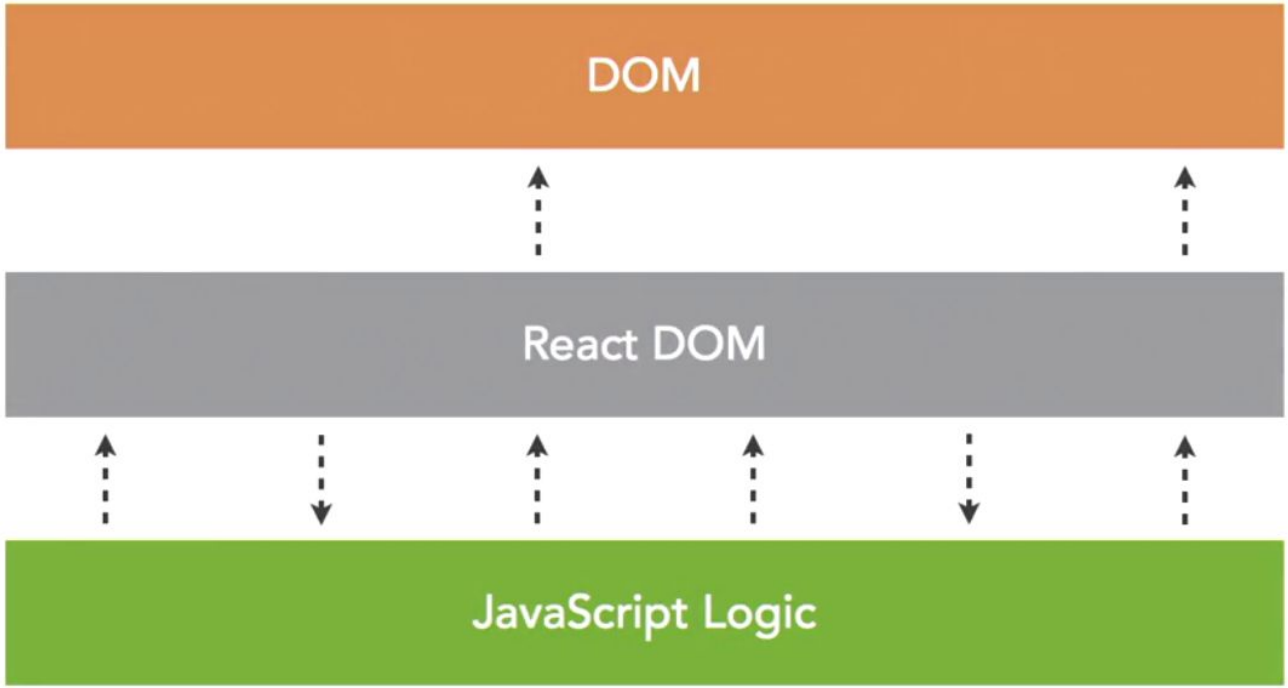
DOM diffing

- 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

```
1   const {createElement} = React;
2   const {render} = ReactDOM;
3
4   const app=createElement('h1',
5       {id: 'titre', className:'header'},
6       'Bonjour!!');
7   render(app, document.getElementById('root'));
8
```

```
3   <head>
4       <meta charset="utf-8">
5       <title>Ma premiere React app</title>
6       <script type="text/javascript" src='../react.js'></script>
7       <script type="text/javascript" src='../react-dom.js'></script>
8   </head>
9   <body>
10  <div id='root'></div>
11  <script type="text/javascript" src='index.js'></script>
12  </body>
13  </html>
```

ReactJS: Example

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

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

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

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

JSX

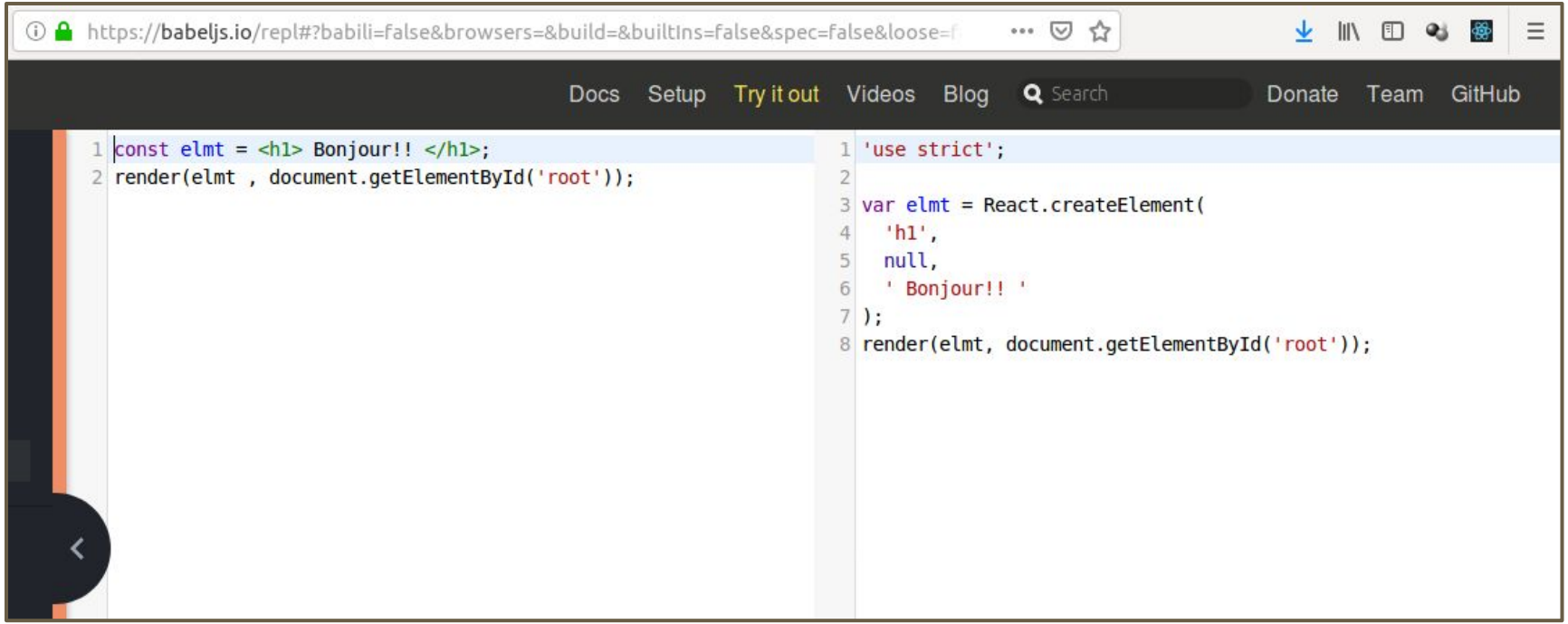
- Du “***HTML***” dans ***javascript***

```
1  const {render} = ReactDOM;
2
3  const elmt = <h1> Bonjour!! </h1>;
4  render(elmt , document.getElementById('root'));
5
```

⇒ Erreur

Besoin d’un interpréteur ***⇒ Babel***

JSX: Babel


https://babeljs.io/repl#?babili=false&browsers=&build=&builtIns=false&spec=false&loose=
Docs
Setup
Try it out
Videos
Blog
Search
Donate
Team
GitHub
1 const elmt = <h1> Bonjour!! </h1>;
2 render(elmt , document.getElementById('root'));
1 'use strict';
2
3 var elmt = React.createElement(
4 'h1',
5 null,
6 ' Bonjour!! '
7);
8 render(elmt, document.getElementById('root'));

JSX: Babel

index.js

```
1  const {render} = ReactDOM;
2
3  const elmt = <h1> Bonjour!! </h1>;
4  render(elmt , document.getElementById('root'));
5
```

index.html

```
3   <head>
4       <meta charset="utf-8">
5       <title>First React app</title>
6       <script type="text/javascript" src='../react.js'></script>
7       <script type="text/javascript" src='../react-dom.js'></script>
8       <script type="text/javascript" src='
        https://cdnjs.cloudflare.com/ajax/libs/
        babel-core/5.8.38/browser.js'></script>
9   </head>
10  <body>
11  <div id='root'></div>
12  <script type="text/babel" src='index.js'></script>
13  </body>
14  </html>
```

JSX: Babel

```
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.js

index.html

```
t="utf-8">
: React app</title>
="text/javascript" src='../react.js'></script>
="text/javascript" src='../react-dom.js'></script>
="text/javascript" src='
s.cloudflare.com/ajax/libs/
.8.38/browser.js'></script>
</head>
<body>
<div id='root'></div>
<script type="text/babel" src='index.js'></script>
</body>
</html>
```

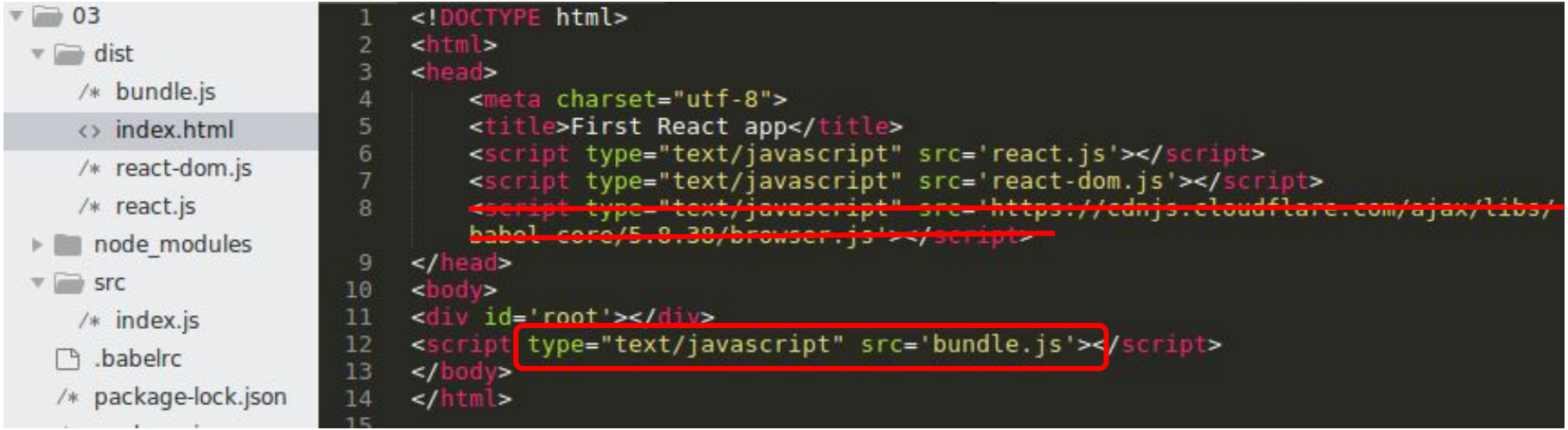
JSX: Babel

- Avec cette méthode, la transformation du jsx en javascript se fait dans le browser.
- Methode valable pour le teste
- Pour la production il faut utiliser le fichier js préalablement transformé

Babel: installation

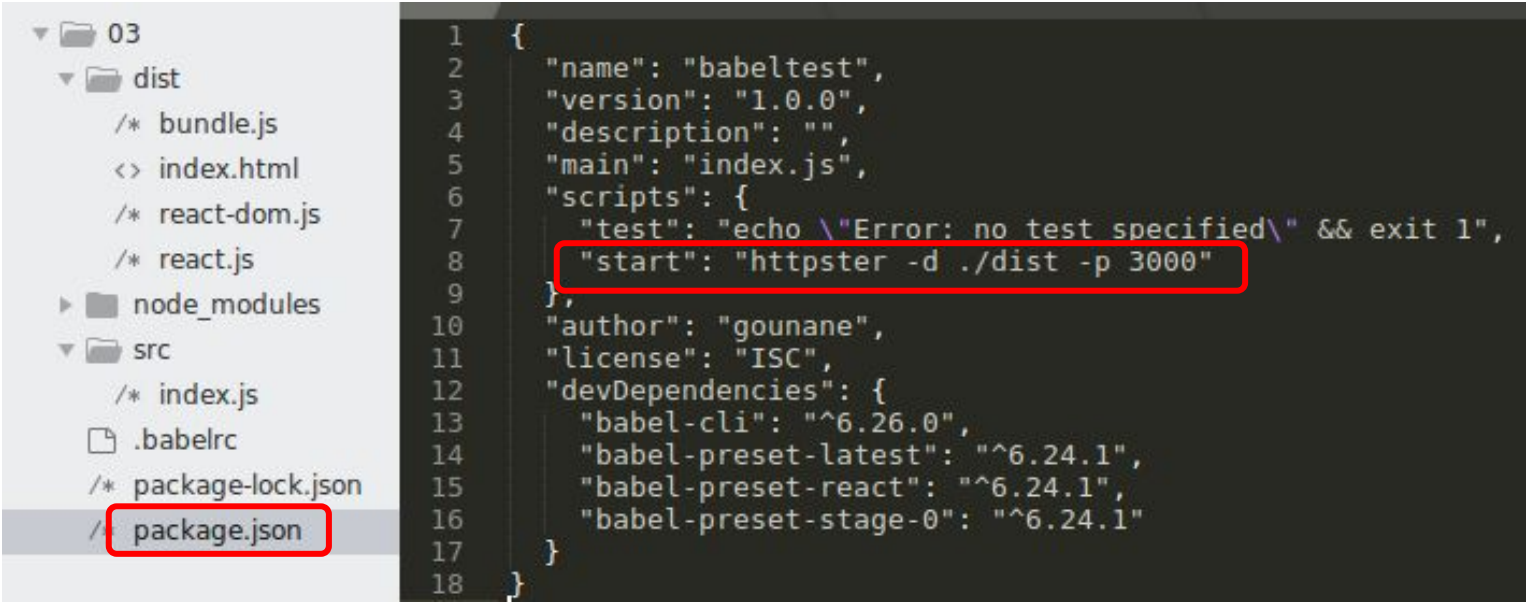
- $ ***npm init*** ⇒ package.json
- $***npm install --save-dev babel-cli*** ou ***sudo npm install -g babel-cli***
- $***npm install --save-dev babel-preset-latest babel-preset-react babel-preset-stage-0***
- Dans le fichier ***.babelrc***: ***{ 'presets': ['latest', 'react', 'stage-0'] }***

JSX: Babel



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

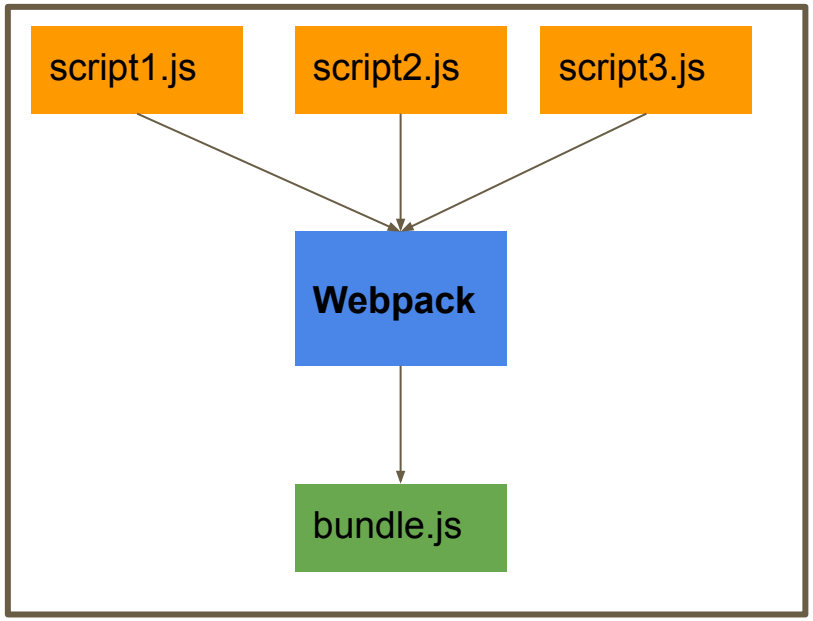
JSX: Babel



- $***npm start***

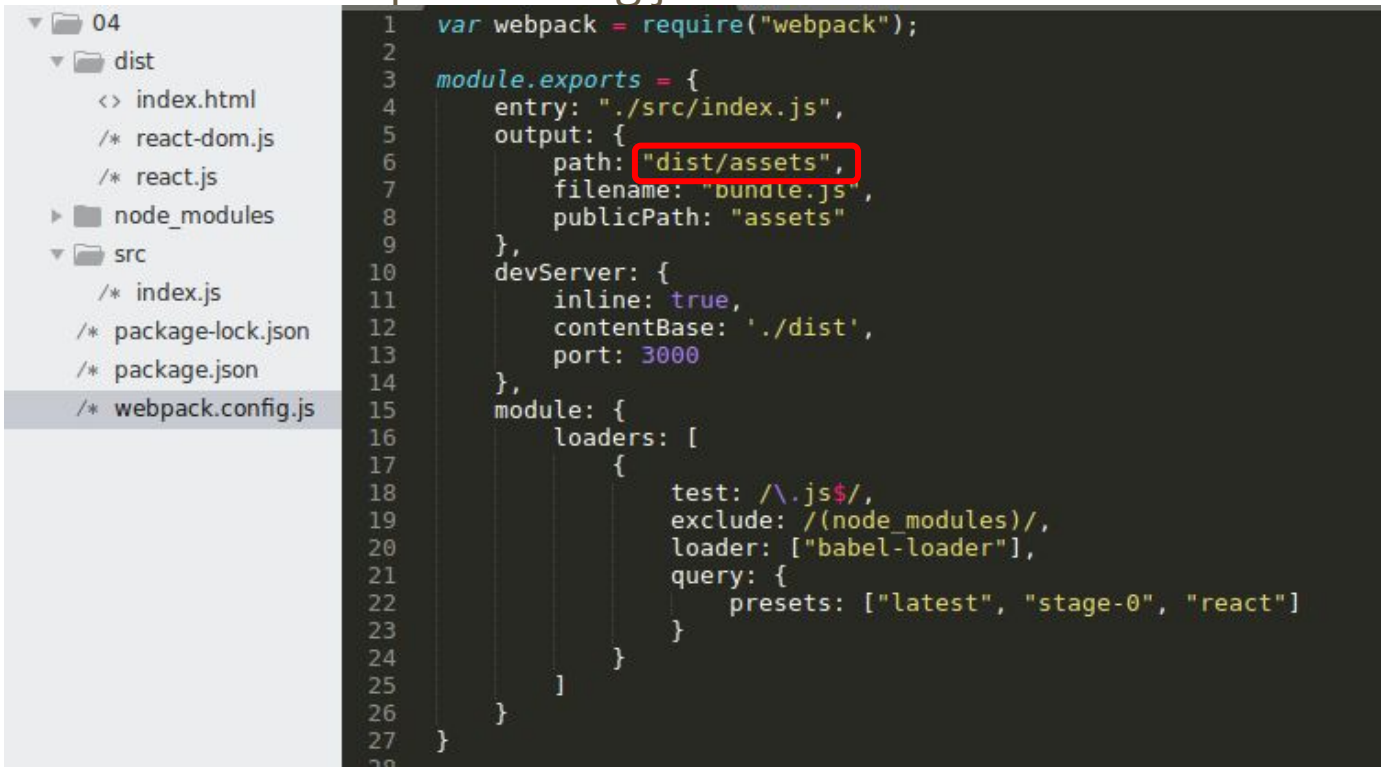
Webpack

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



Webpack

Créer un fichier webpack.config.js

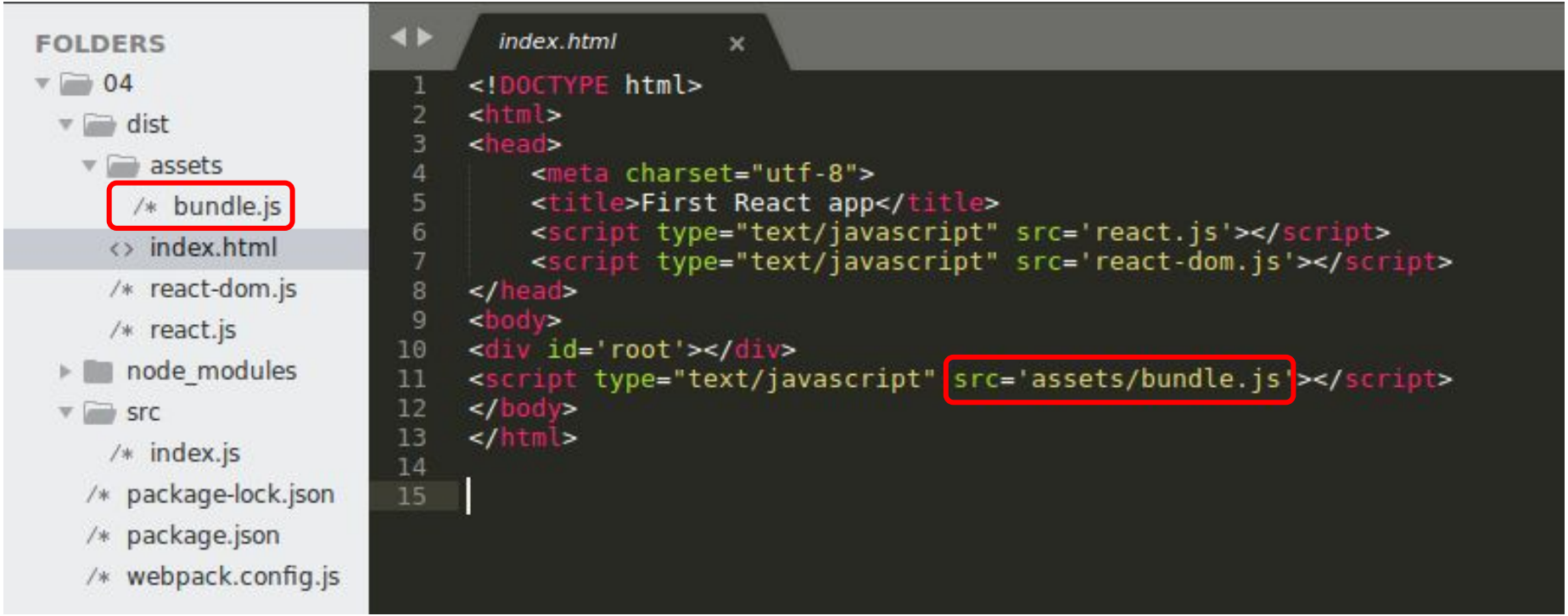


Webpack

- $ ***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***

```
5       <title>First React app</title>
6       <script type="text/javascript" src='react.js'></script>
7       <script type="text/javascript" src='react-dom.js'></script>
8   </head>
9   <body>
10  <div id='root'></div>
11  <script type="text/javascript" src='assets/bundle.js'></script>
12  </body>
```

Webpack


FOLDERS
04
dist
assets
bundle.js
index.html
react-dom.js
react.js
node_modules
src
index.js
package-lock.json
package.json
webpack.config.js
index.html
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="utf-8">
5 <title>First React app</title>
6 <script type="text/javascript" src='react.js'></script>
7 <script type="text/javascript" src='react-dom.js'></script>
8 </head>
9 <body>
10
11 <script type="text/javascript" src='assets/bundle.js'></script>
12 </body>
13 </html>
14
15

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'));
```

Creer une application react sans configuration

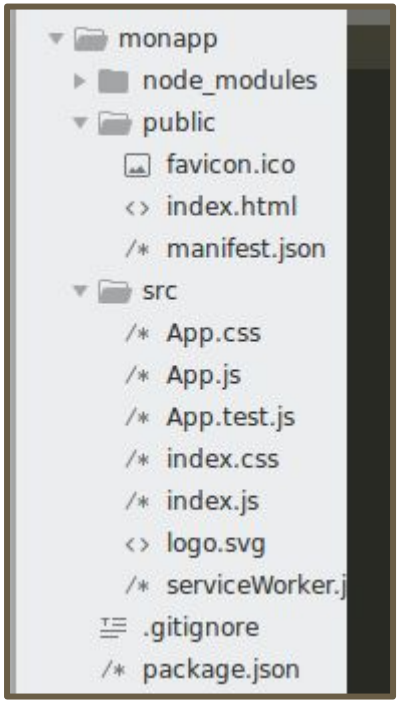
Pour créer une application reactjs sans passer par les étapes précédentes, la commande create-react-app se charge de toutes les configurations nécessaires.

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

```
11    <body>
12      <noscript>
13        You need to enable JavaScript to run this app.
14      </noscript>
15      <div id="root"> </div>
16    </body>
17  </html>
```

index.js

```
1  import React from 'react';
2  import ReactDOM from 'react-dom';
3  import './index.css';
4  import App from './App';
5
6  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

```
1  import React, { Component } from 'react';
2  import logo from './logo.svg';
3  import './App.css';
4
5  class App extends Component {
6    render() {
7      const titre="MonApp React";
8      return (
9        <div className="App">
10           <h1>{titre}</h1>
11           <p>Voice ma premiere application ReactJS</p>
12       </div>
13     );
14   }
15 }
16
17 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

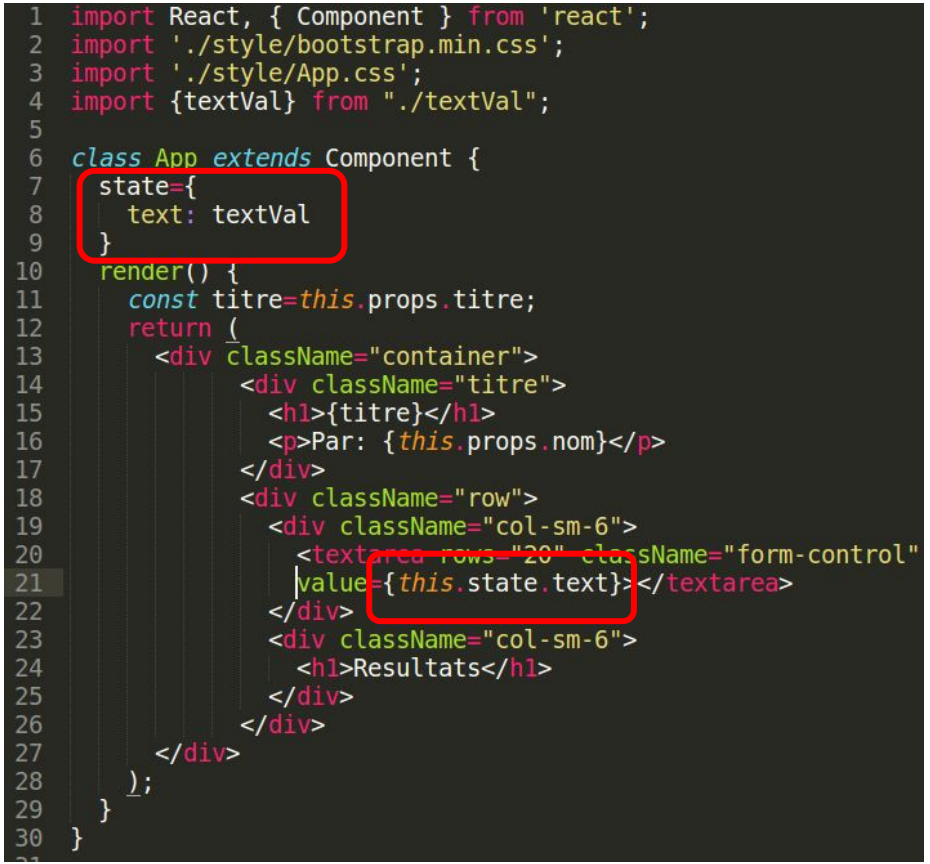
```
5  class App extends Component {
6    render() {
7      const titre=this.props.titre;
8      return (
9        <div className="App">
10             <h1>{titre}</h1>
11             <p>Une premiere application ReactJS par {this.props.nom}</p>
12       </div>
13     );
14   }
15 }
```

index.js

```
1  import React from 'react';
2  import ReactDOM from 'react-dom';
3  import './index.css';
4  import App from './App';
5
6  ReactDOM.render(<App titre="ReactJS" nom="Yasser" />, document.getElementById('root'));
7
```

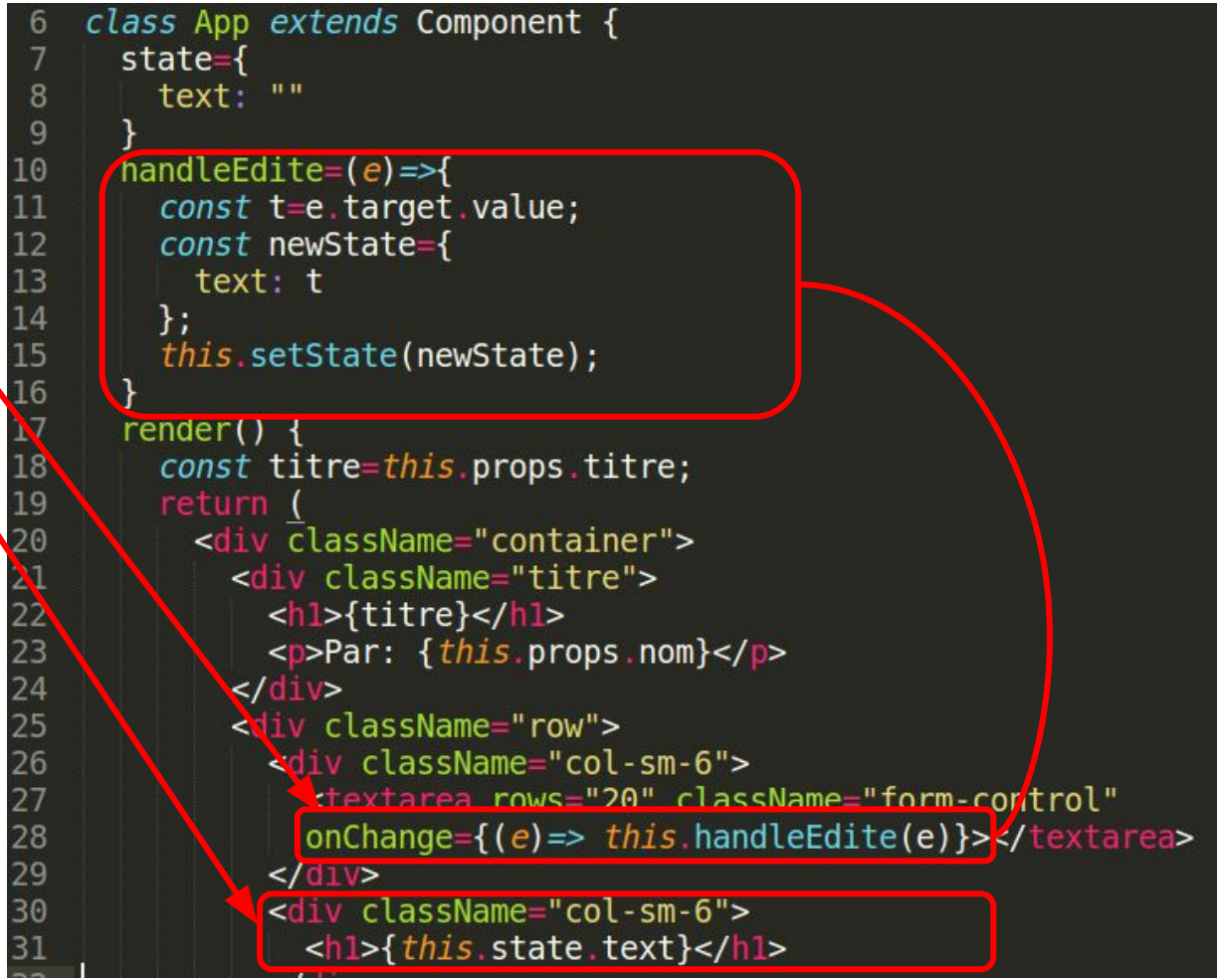
Le state

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)***;



Le state

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



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.