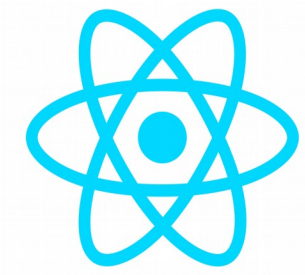
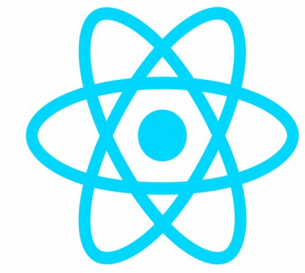


Web Developer
Scripts Clients

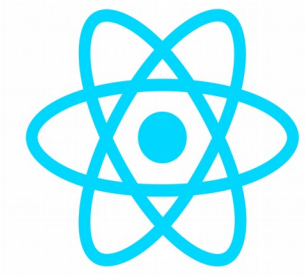
Bonus
04 – ReactJS

A. Martel 2018-2019



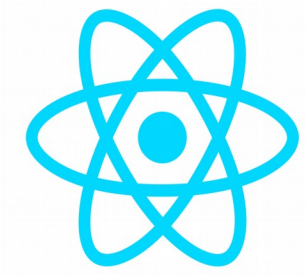
Plan du chapitre 4

1. Présentation
2. Installation



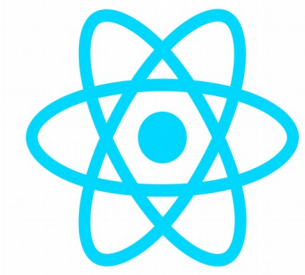
4.1. Présentation

- React (ou ReactJS) est une bibliothèque JS créée en 2011 par un ingénieur de Facebook aidé, pour la généralisation du code, par un ingénieur d'Instagram
- Application libre publiée en 2013
- Utilisable côté client et côté serveur
- Utilisée par des sites comme Facebook, Instagram, Twitter, Netflix, Airbnb, Dailymotion, Paypal, etc.



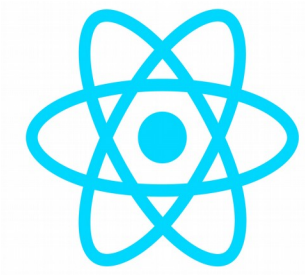
4.1. Présentation

- Il s’agit d’une bibliothèque frontend
- Les bibliothèques et les frameworks frontend les plus populaires sont React, Angular, Vue.js, Ember, etc.
- Ceux-ci nous permettent de créer plus aisément des applications riches pour les navigateurs



4.1. Présentation

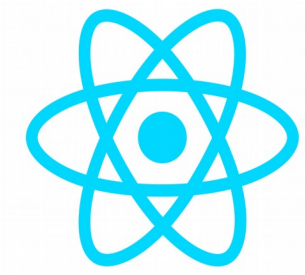
- Principe de création et d'utilisation de composants : on ne sépare plus structure, aspect et comportement mais au contraire, on les rassemble au sein d'un composant
- De base, pas de système de templates
- Full JS (ES6)
- JSX pour l'aspect « HTML »
- DOM Virtuel (manipulation du DOM en mémoire puis « injection » dans le DOM du navigateur)



4.1. Présentation

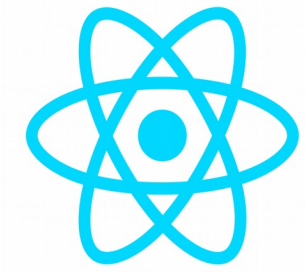
- Une application est une arborescence de composants
- Pour faciliter les échanges entre eux sans les connecter on applique le principe suivant :

  Les données descendent (vers les fils) et les états remontent (vers les parents)



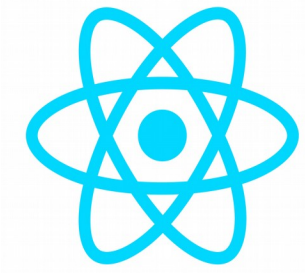
4.1. Présentation

- Versions complémentaires :
  - React Native : pour créer des applications iOS, etc.
  - React pour IoT
  - React pour VR



4.2. Installation

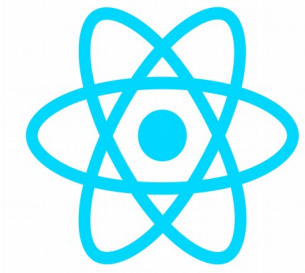
- Le plus simple pour installer React est d'utiliser un gestionnaire de paquets
- Dans la suite de cette présentation, nous utiliserons npm (Node Package Manager)



4.2. Installation

- Installation de React :

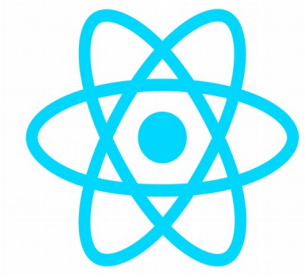
```
npm install -g create-react-app
```



4.3. Création du projet

- Commencez par vous placer dans votre répertoire de programmation
- Lancez la commande suivante pour créer le projet `first-react` :

```
create-react-app first-react
```



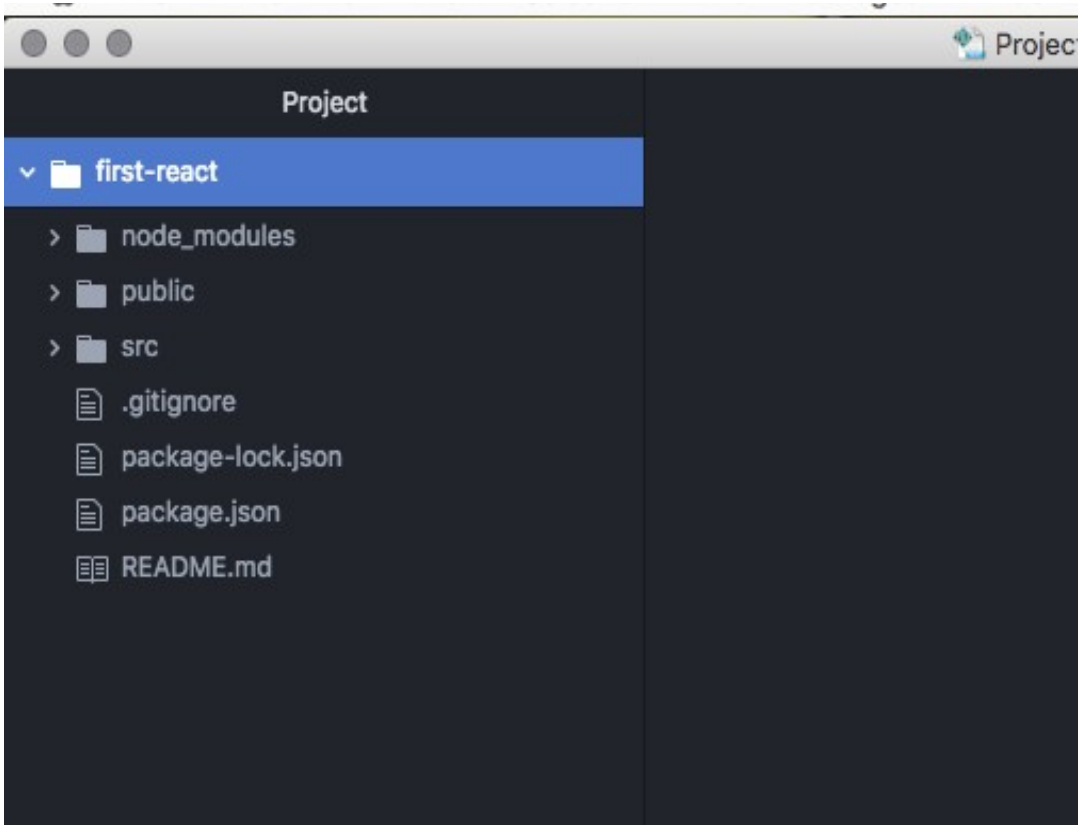
4.3. Création du projet

- Placez vous dans ce répertoire et lancez le serveur de développement :

```
cd first-react

npm start
```

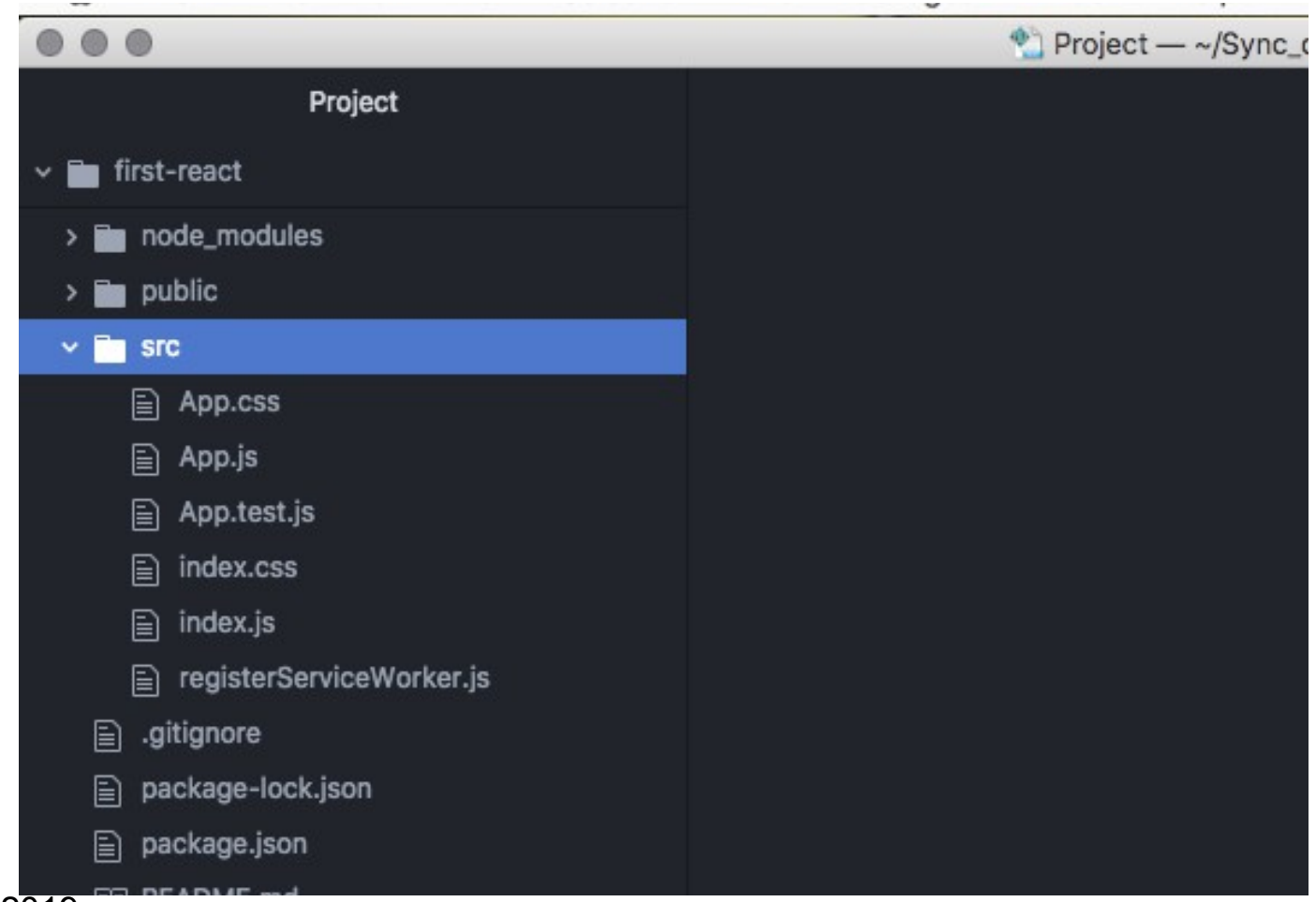
4.4. Arborescence

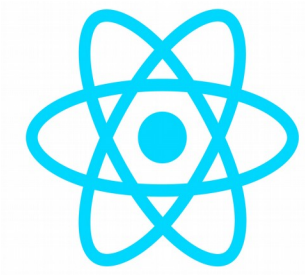
- Cette commande a créé toute l'arborescence du projet ainsi que les fichiers nécessaires :



4.4. Arborescence

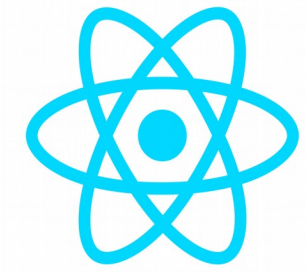
- Nous modifierons les fichiers du dossier `src` :





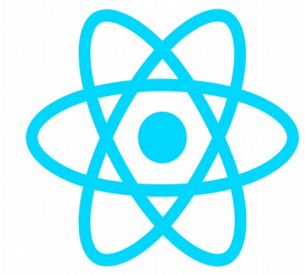
4.4. Arborescence

- Dans le dossier `public`, on trouve le point d'entrée du site : `index.html` (fichier obligatoire en React)
- Dans le dossier `src`, nous trouvons :
  - `index.js` : point d'entrée de l'application React (fichier obligatoire en React)
  - `App.js` : composant appelé « App »



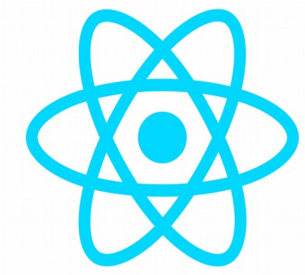
4.4. Arborescence

- Afin de tout reprendre à zéro, nous allons supprimer tous les fichiers du dossier `src` et les récréer un à un.
- Il faut préciser que tous les fichiers que nous utiliserons dans notre application devront être placés dans le répertoire `src`.



4.5. Fichiers sources

- Nous allons utiliser des instructions ES2015 (notamment pour créer les composants à l'aide de classes) que nous expliquerons au fur et à mesure
- Le premier que nous allons créer est `index.js` puisque c'est le seul fichier obligatoire

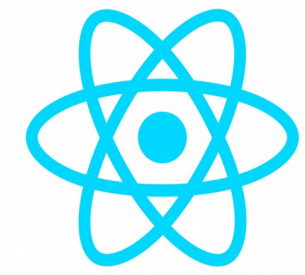


4.5. Fichiers sources

`src/index.js`

- Nous devons d'abord charger la bibliothèque React et le gestionnaire de DOM :

```
import React from 'react';

import ReactDOM from 'react-dom';
```



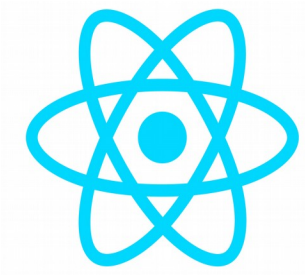
4.5. Fichiers sources

`src/index.js`

- Ensuite, nous créons un premier composant via une fonction. Ils sont appelés « composant pur fonctionnel » (attention, son nom doit débuter par une majuscule afin de signaler au moteur React qu'il s'agit d'un composant React et pas un objet natif)

```
function PremierComposant(){
  return React.createElement('p', {}, 'Bienvenue dans React !');
}
```

- `React.createElement` est une fonction de manipulation du DOM virtuel qui prend comme paramètres : l'élément à créer, un objet avec les attributs de l'élément et enfin le contenu de cet élément.



4.5. Fichiers sources

`src/index.js`

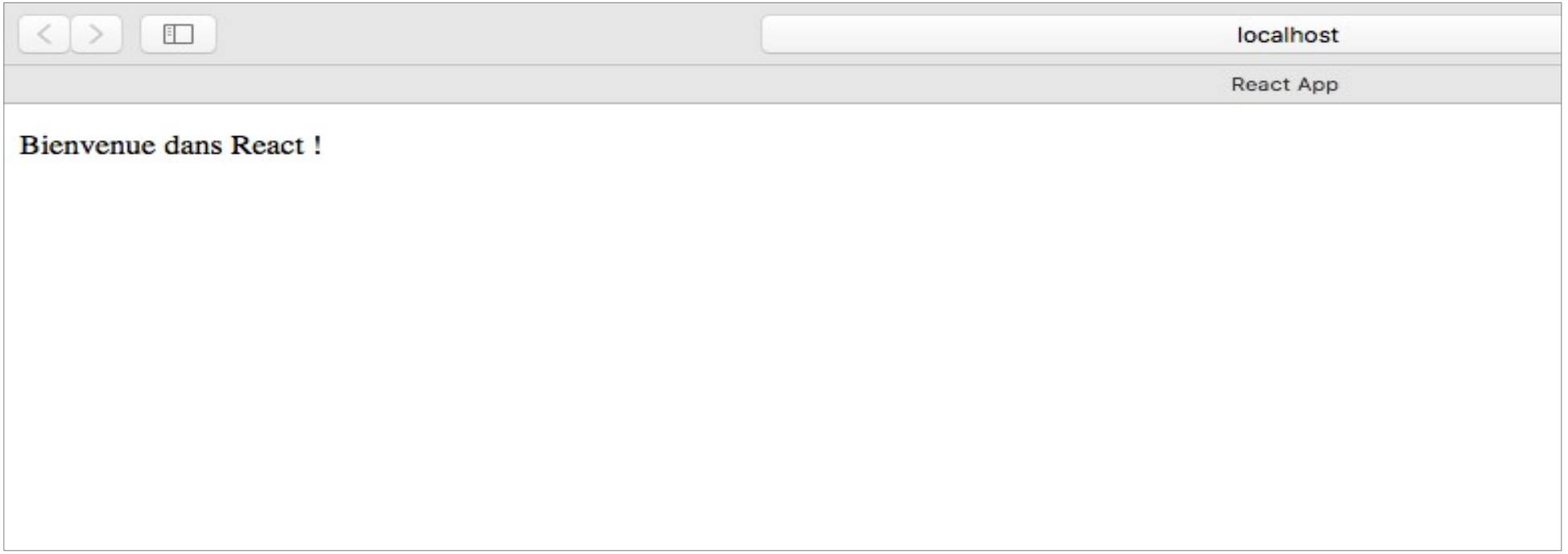
- Il nous reste à utiliser ce nouveau composant :

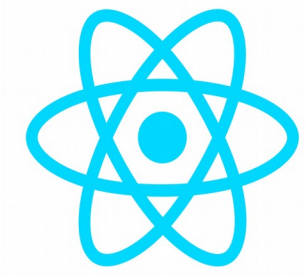
```
ReactDOM.render(
  React.createElement(PremierComposant),
  document.getElementById('root')
);
```

- ReactDOM.render sert à « injecter » le DOM virtuel dans le DOM du navigateur (la deuxième instruction précise l'élément dans lequel on va placer ce nouvel élément)

4.5. Fichiers sources

- On teste avec un `npm start` :



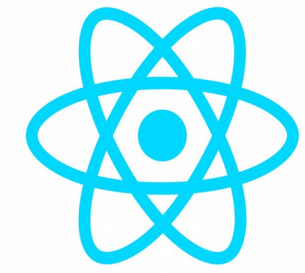


4.6. JSX

`src/index.js`

- Remplaçons maintenant l'appel à la fonction par du code JSX (dans un premier temps très proche du HTML) :

```
function PremierComposant(){
  return <p>Bienvenue dans React JSX !</p>
}
```

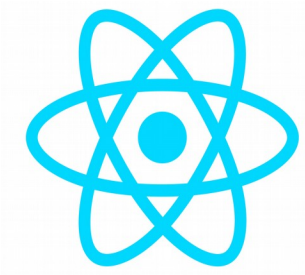


4.6. JSX

src/index.js

- Et l'appel simplifié à ce composant :

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

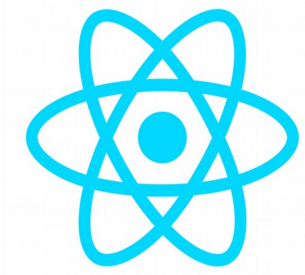


4.7. Props

- Afin de passer des paramètres à nos composants, nous utiliserons des attributs appelés `props` avec le JSX en React

```
function PremierComposant({nom='John Doe'}){
  return <p>{nom}, bienvenue dans React !</p>
}
```

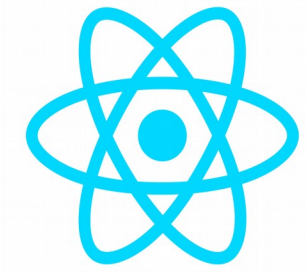
- Comme on le voit, on peut définir une valeur par défaut pour les props



4.7. Props

- Afin de passer des paramètres à nos composants, nous utiliserons des attributs appelés `props` en React

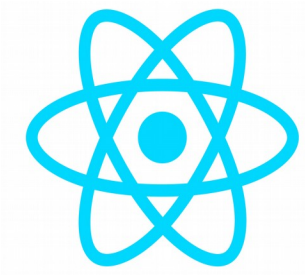
```
ReactDOM.render(
  <PremierComposant nom="Bill Boule"/>,
  document.getElementById('root')
);
```



4.7. Props

- Appels multiples à notre composant :

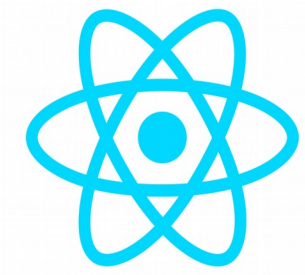
```
ReactDOM.render(
  <div>
    <PremierComposant />
    <PremierComposant nom="Bill Boule"/>
  </div>,
  document.getElementById('root')
);
```



4.7. Props

- Les `props` peuvent prendre deux types de valeurs :
  - String : valeurs entourées de double-quotes
  - Toutes les autres valeurs (y compris du JSX) : valeurs entourées d'accolades

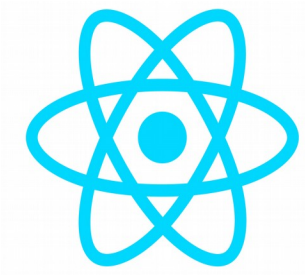
Rem : il existe un raccourci pour `true` car si on ne précise pas de valeur pour une `prop` (par exemple `ma_prop`) cela équivaut à écrire `ma_prop={true}`



4.7. Props

- Exemple :

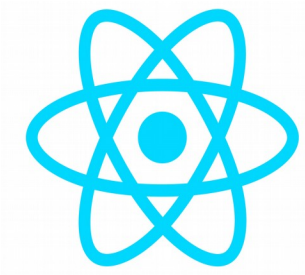
```
function PremierComposant({nom='John Doe', age}){
  return <p>{nom}, bienvenue dans React ! Vous avez {age}
ans</p>
}
ReactDOM.render(
  <div>
    <PremierComposant age={32} />
    <PremierComposant nom="Bill Boule" age={26} />
  </div>,
  document.getElementById('root')
);
```



4.7. ES2015

- Exemple :

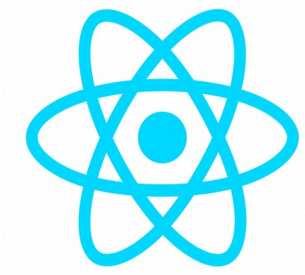
```
const PremierComposant = ({nom='John Doe', age})=>(
  <p>{nom}, bienvenue dans React ! Vous avez {age} ans</p>
)
ReactDOM.render(
  <div>
    <PremierComposant age={32}/>
    <PremierComposant nom="Bill Boule" age={26}/>
  </div>,
  document.getElementById('root')
);
```



4.8. Découpe en fichiers

- L'idéal est de créer un fichier par composant
- Nous donnerons le nom du composant au fichier (majuscule comprise)
- Créons donc le fichier PremierComposant.js dont le contenu est le suivant :

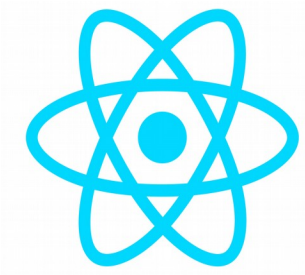
```
import React from 'react';

const PremierComposant = ({nom='John Doe', age})=>(

  <p>{nom}, bienvenue dans React ! Vous avez {age} ans</p>

)

export default PremierComposant;
```



4.8. Découpe en fichiers

- La dernière ligne de code (`export`) permet d'utiliser le composant dans d'autres fichiers
- Modifions maintenant le fichier index.js :

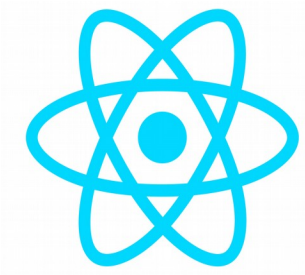
```
import React from 'react';

import ReactDOM from 'react-dom';


import PremierComposant from './PremierComposant';


ReactDOM.render(…
```



4.9. Événements

- Ajoutons un événement qui affichera le nom dans la console lorsqu'on cliquera sur un paragraphe :

```
const PremierComposant = ({nom='John Doe', age})=>(

  <p onClick={()=>console.log("Hello {nom}")}>{nom},
bienvenue dans React ! Vous avez {age} ans</p>

)
```

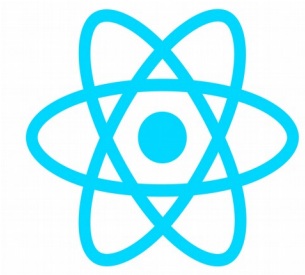


4.9. Événements

- Lorsqu'on clique, un texte apparaît dans la console mais le nom n'est pas affiché
- En effet, pour remplacer le contenu d'une variable au sein d'une chaîne de caractères en JSX, nous devons utiliser la syntaxe suivante :

```
const PremierComposant = ({nom='John Doe', age})=>(

  <p onClick={()=>console.log(`Hello ${nom}`)}>{nom},
bienvenue dans React ! Vous avez {age} ans</p>

)
```

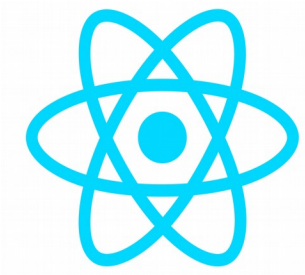
- Il faut entourer la chaîne de `backquotes` et préfixer la variable d'un `$`



4.10. Classes

- Réécrivons maintenant notre composant sous la forme d'une classe :

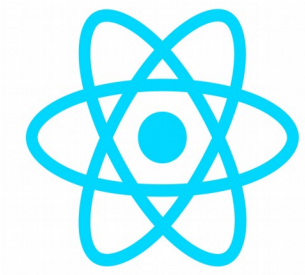
```
import React, { Component } from 'react';

class PremierComposant extends Component {

  render(){

     return <p onClick={()=>console.log(`Hello ${nom}`)}>{nom}, bienvenue dans React ! Vous avez {age} ans</p>

  }

}

export default PremierComposant;
```



4.10. Classes

- Ce code ne fonctionne pas car les `props` ne sont plus accessibles directement. Nous devons passer par l'objet `this` et son sous-objet `props` :

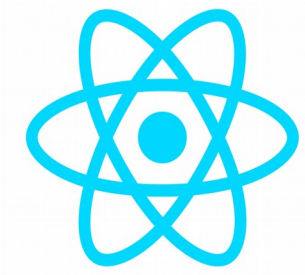
```
import React, { Component } from 'react';

class PremierComposant extends Component {

  render(){

     return <p onClick={()=>console.log(`Hello ${
{this.props.nom}`)}>{this.props.nom}, bienvenue dans React
! Vous avez {this.props.age} ans</p>

  }

}

export default PremierComposant;
```



4.10. Classes

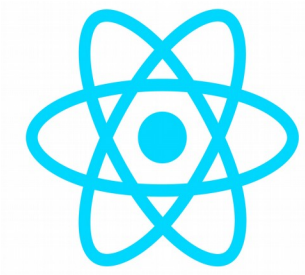
- Notre code fonctionne, mais nous avons perdu nos valeurs par défaut :

```
import React, { Component } from 'react';
class PremierComposant extends Component {
  render()…
}
PremierComposant.defaultProps={
  nom:'John Doe'
};
export default PremierComposant;
```



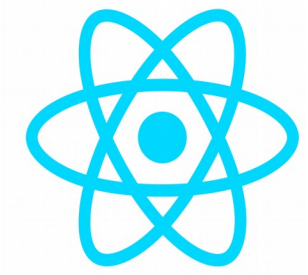
4.10. Classes

- Nous pourrions également utiliser des propriétés statiques directement dans le composant mais cela nécessite un traducteur ES2015 de type Babel ce qui compliquerait encore un peu cette introduction à ReactJS



4.11. Conditions

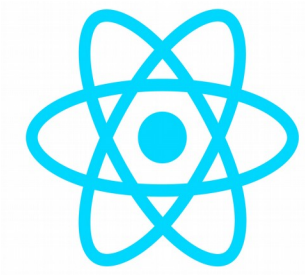
- Dans le JSX, si nous voulons utiliser des conditions, nous n’avons pas accès aux if...else du JS. Nous pouvons en revanche utiliser directement les opérateurs logiques && et || ou l’opérateur ternaire ? :



4.11. Conditions

- Par exemple, affichons « Monsieur » ou « Madame » selon l'état de la prop `sexe` :

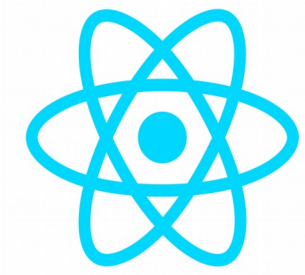
```
…
   render(){
      return
<p ...>{this.props.sexe==='M'?"Monsieur ":"Madame
"}{this.props.nom}, bienvenue dans React ! Vous
avez {this.props.age} ans</p>
   }
```



4.11. Conditions

- Pour rendre le code plus lisible, nous pouvons évidemment utiliser des variables :

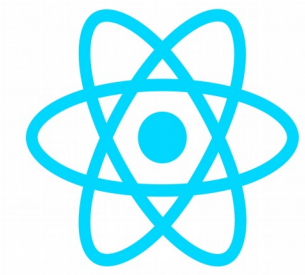
```
class PremierComposant extends Component {

  render(){

    const sexe = this.props.sexe==='M'?"Monsieur ":"Madame ";

    const texte = `${sexe}${this.props.nom}, bienvenue dans
React ! Vous avez ${this.props.age} ans`;


    return <p onClick={()=>console.log(`Hello $
{this.props.nom}`)}>{texte}</p>

  }

}
```



4.11. Conditions

- Pour s'assurer qu'une `prop` soit du bon type, soit obligatoire ou dont la valeur appartienne à une liste, nous pouvons utiliser les `propTypes` (peut nécessiter une installation `npm install --save prop-types`) :

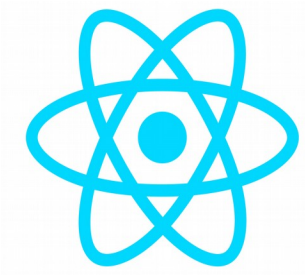
```
import PropTypes from 'prop-types'
…
PremierComposant.defaultProps={
  nom: 'John Doe',
  sexe : 'M'
};
PremierComposant.propTypes={
  nom: PropTypes.string,
  sexe : PropTypes.oneOf([
    'M', 'm', 'F', 'f'
  ])
};
```



4.11. Conditions

- On peut également utiliser nos méthodes comme Event Handler. Pour utiliser les props au sein de ces méthodes, il faut lier le contexte de l'objet :

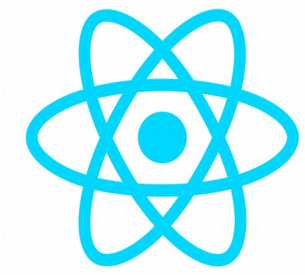
```
class PremierComposant extends Component {
  paragrapheClick(){
    console.log(`Hello ${this.props.nom}`);
  }
  render(){
    …
    return <p onClick={this.paragrapheClick.bind(this)}>{texte}</
p>
  }
}
```



4.12. Itérer des listes

- Soit une liste d'élèves à afficher dans une table
- En JSX, nous devons utiliser une fonction d'itération, ici, map (filter, keys, some,etc.) :

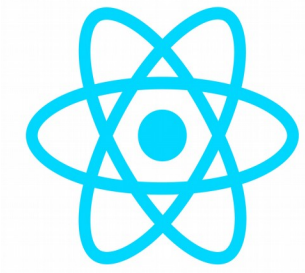
```
class PremierComposant extends Component {
  render(){
    return (
      <table>
      <thead><tr><th>Nom</th><th>Prénom</th><th>Date de naissance</th></tr></thead>
      <tbody>
        {this.props.eleves.map((eleve)=>(
          <tr><td>{eleve.nom}</td><td>{eleve.prenom}</td><td>{eleve.date_naissance}</
td></tr>
        ))}
      </tbody></table>
    )
  }
}
```



4.12. Itérer des listes

- Avec comme objet props :

```
PremierComposant.defaultProps={

  …

  eleves:[

    {nom:"Gomez", prenom:"Mario",
date_naissance:"10/10/1987"},

    {nom:"Sanchez", prenom:"Gino",
date_naissance:"10/10/1988"},

    {nom:"Gutierrez", prenom:"Pepe",
date_naissance:"10/10/1989"},

    {nom:"Pez", prenom:"Guido", date_naissance:"10/10/1990"}

  ]

};
```



4.13. Exemple

- Essayons d'implémenter la liste d'élèves vue en jQuery
- Créons une nouvelle application : `gestion-classe`