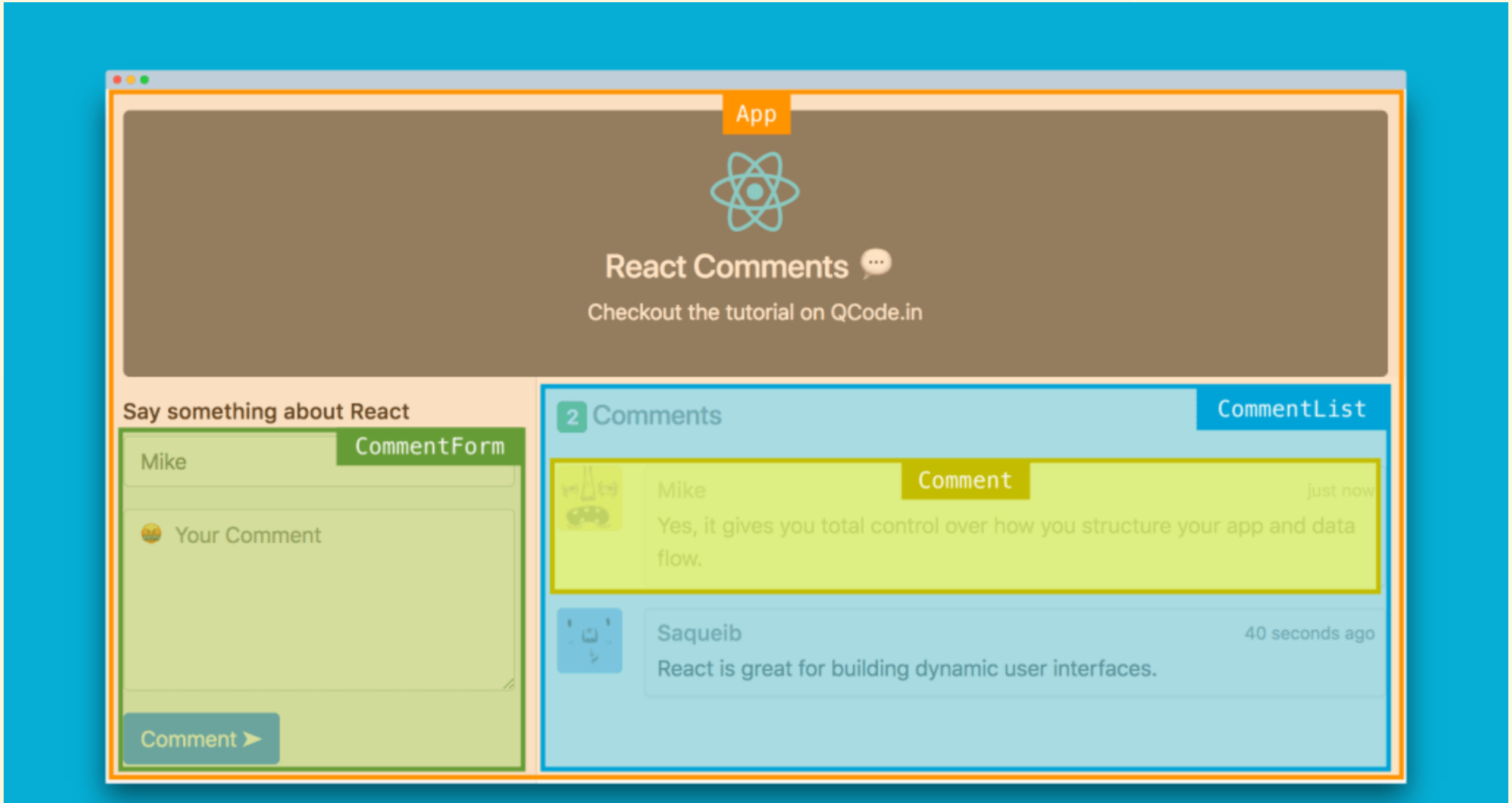
ReactJS - Les fondamentaux

Pourquoi apprendre ReactJS

- Bibliothèque JavaScript créée par Facebook
- Framework JS avec le plus d'offres d'emploi
- Sert à concevoir des applications web complexes
- Utilisé par: Airbnb, Codecademy, Dailymotion, Discord, Dropbox, **Facebook**, IMDb, Instagram, Netflix, Paypal, Tesla, Twitter, Wordpress, Yahoo!

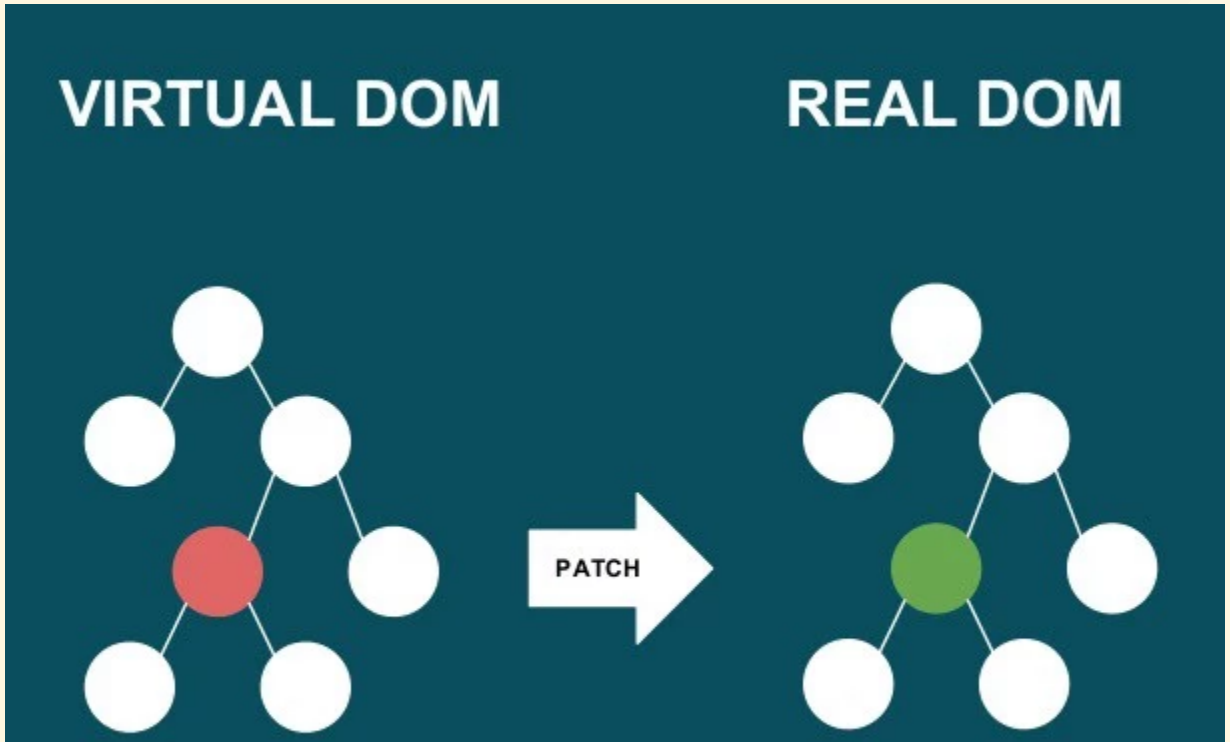
https://fr.reactjs.org/

La logique des composants



Le DOM Virtuel

Modification du DOM: Opération coûteuse
DOM Virtuel: Copie du DOM dans un objet JS. ReactJS n'ajustera le DOM que quand c'est nécessaire.



Compilateur JS

Chaque année, de nouvelles fonctionnalités sont ajoutées au JavaScript.

Pour les exploiter, un outil: **Babel**

Babel va transformer notre utilisation des dernières spécificités en code compréhensible par les navigateurs actuels (ES6===ES2015)

NB: Nécessaire pour ReactJS et son extension JSX

JSX

```
const title = <h1>Mon titre principal</h1>;
```

> Cette drôle de syntaxe n'est ni une chaîne de caractères ni du HTML. Ça s'appelle du JSX, et c'est une extension syntaxique de JavaScript. [...] JSX produit des « éléments » React

https://fr.reactjs.org/docs/introducing-jsx.html

Liens CDN

https://fr.reactjs.org/docs/cdn-links.html

```
<script crossorigin src="https://unpkg.com/react@16/umd/react.development.js"></script>
<script crossorigin src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"></script>
```

https://babeljs.io/

```
<script src="https://unpkg.com/babel-standalone@6/babel.min.js"></script>

<script type="text/babel">
    // Our script
</script>
```

Composants & Templates

```
<div id="app"></div>

<script type="text/babel">
    class App extends React.Component {
    render(){
        return(
            <div className="app__content">
                <h1>Youhou la famille!</h1>
                <p>Comment ça va?</p>
            </div>
        )}
    }

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

Indications sur les composants

- 1 seul élément "racine" -> root element

```
// !THIS CODE DOES NOT WORK!
return(
    <h1>Youhou la famille!</h1>
    <p>Comment ça va?</p>
)
```

```
// THIS CODE WORKS
return(
    <div class="text">
        <h1>Youhou la famille!</h1>
        <p>Comment ça va?</p>
    </div>)
```

- `render()` est une méthode implémentée dans la classe `React.Component` obtenue en héritage grâce à `extends`
- Le `return` de la méthode `render()` doit renvoyer du JSX
- Au lieu de l'attribut `class`, c'est `className`

On peut déclencher des expression JS grâce aux `{}`

```
return(
    <div className="text">
        <p>Nombre aléatoire: {Math.random()}</p>
    </div>
)
```

Les états de composant

Les états de composant (**state**) sont regroupés dans un objet JS.

Précisent l'état actuel du composant: ses données, son affichage dynamique (ouvert/fermé, visible, etc...)

Les états peuvent être changés au cours du temps.
Exemples: Une fenêtre modale qui s'ouvre/se ferme, Un élément qui est ajouté/retiré d'une liste, etc...

```
{
    cities: [
        { name: "New-York", population: 862288 },
        { name: "Paris", population: 2187526 },
        { name: "Madrid", population: 3266126 }
    ]
}
```

```
{
    showSidebar: true
}
```

Déclaration d'états

```
state = {
    familyName: "Lavisse",
    nbChild: 2
}

render(){
    return(
    <div className="app__content">
        <h1>Youhou la famille {this.state.familyName}!</h1>
        <p>Comment ça va avec {this.state.nbChild} enfants?</p>
    </div>
    )
}
```

React Developer Tools

Suite d'outils pour débugger une app React.JS
Fait apparaître deux nouveaux onglets près de la console:
Components et **Profiler**

Pour Google Chrome:
https://chrome.google.com/webstore/detail/react-developer-tools/fmkadmapgofadopljbjfkapdkoienihi

Pour Firefox:
https://addons.mozilla.org/fr/firefox/addon/react-devtools/

Les événements du DOM

```
clickButton(e) {
    console.log("Cliqué sur le bouton");
    console.log(e.target); // <button>Envoyer</button>
}

render(){
    return(
    <div className="app__content">
        <button onClick={this.clickButton}>Envoyer</button>
    </div>
    )
```

`onClick` mais aussi `onChange`, `onSubmit`, etc...

L'événement peut être récupéré en paramètre

Les événements et this

Utilisation d'une fonction fléchée pour récupérer le `this` du composant

```
clickButton = (e) => {
    console.log(this.state.nbChild); //2
}

render(){
    return(
    <div className="app__content">
        <button onClick={this.clickButton}>Envoyer</button>
    </div>
    )
```

Changer un état

La mauvaise pratique

Attention: Ne jamais modifier un état directement sur this.state! Le DOM serait potentiellement mal adapté, ou pas rechargé du tout!

```
// VERY VERY BAD PRACTICE
this.state.name = "Jean-Jacques";
```

La bonne pratique

`setState` : méthode de la class `React.Component` qui met à jour intelligemment le DOM au changement d'un état

```
// Good Practice
this.setState({
    name: "Jean-Jacques"
})
```