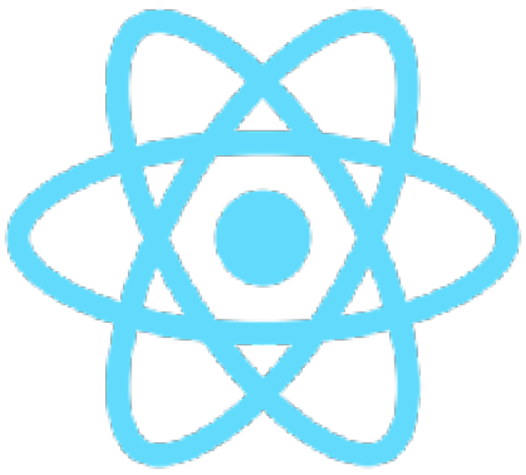
Javascript / React

Rappels et bases



Yohan Giarelli

Architecte logiciel @ Un zéro un

Enseignant IUT Reims et Web School Factory Paris

9 ans d’expérience dans le développement Web

Javascript

« JavaScript est un langage de programmation de scripts principalement employé dans les pages web interactives mais aussi pour les serveurs »

Wikipedia

Javascript

- Créé en 1995
- Standardisé en 1997 (ECMAScript)
- Diverses versions par le passé (JavaScript, JScript, ActionScript)

Javascript

- Récemment renormalisé : ECMAScript 6 (Juin 2015)

Javascript

- Supporté (plus ou moins bien) par tous les navigateurs
- Transpilé (réduit en version) pour les anciens navigateurs
- Disponible comme langage serveur (Node.js)

React

Bibliothèque Javascript pour construire des interfaces utilisateur

React

- Permet de manipuler le DOM (= la représentation de la page web) en Javascript

- Permet de décomposer une application en composants

React

- SPA (Single Page Application)

- PWA (Progressive Web Application)

React

- ReactDOM
- ReactNative
- … ou d’autres (react-hardware)

React

```
<!doctype html>
<html>
<body>
<div id="app"></div>
<script>
    ReactDOM.render(
        <h1>Coucou !</h1>,
        document.getElementById('app')
    );
</script>
</body>
</html>
```

Composants

```
class Hello extends React.Component {
    render() {
        return (
            <div>
                Hello World !
            </div>
        );
    }
}

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

Composants

- Un composant React = un fragment d’application chargé de restituer du DOM ou d’autres composants

- Application = assemblage de composants React

Composants

- Un composant doit retourner un « single root element »
- Balise HTML
- Composant react

JSX

- React utilise une variante de Javascript : JSX
- JSX ≠ JS + HTML
- Nécessite d’être compilé (webpack + babel)

JSX

- Pas de « magie »

- JSX est transformé en Javascript pur

JSX

```
class Hello extends React.Component {
    render() {
        return (
            <div>
                Hello World !
            </div>
        );
    }
}
```

```
class Hello extends React.Component {
    render() {
        return React.createElement(
            "div",
            null,
            "Hello World !"
        );
    }
}
```

Mise en place

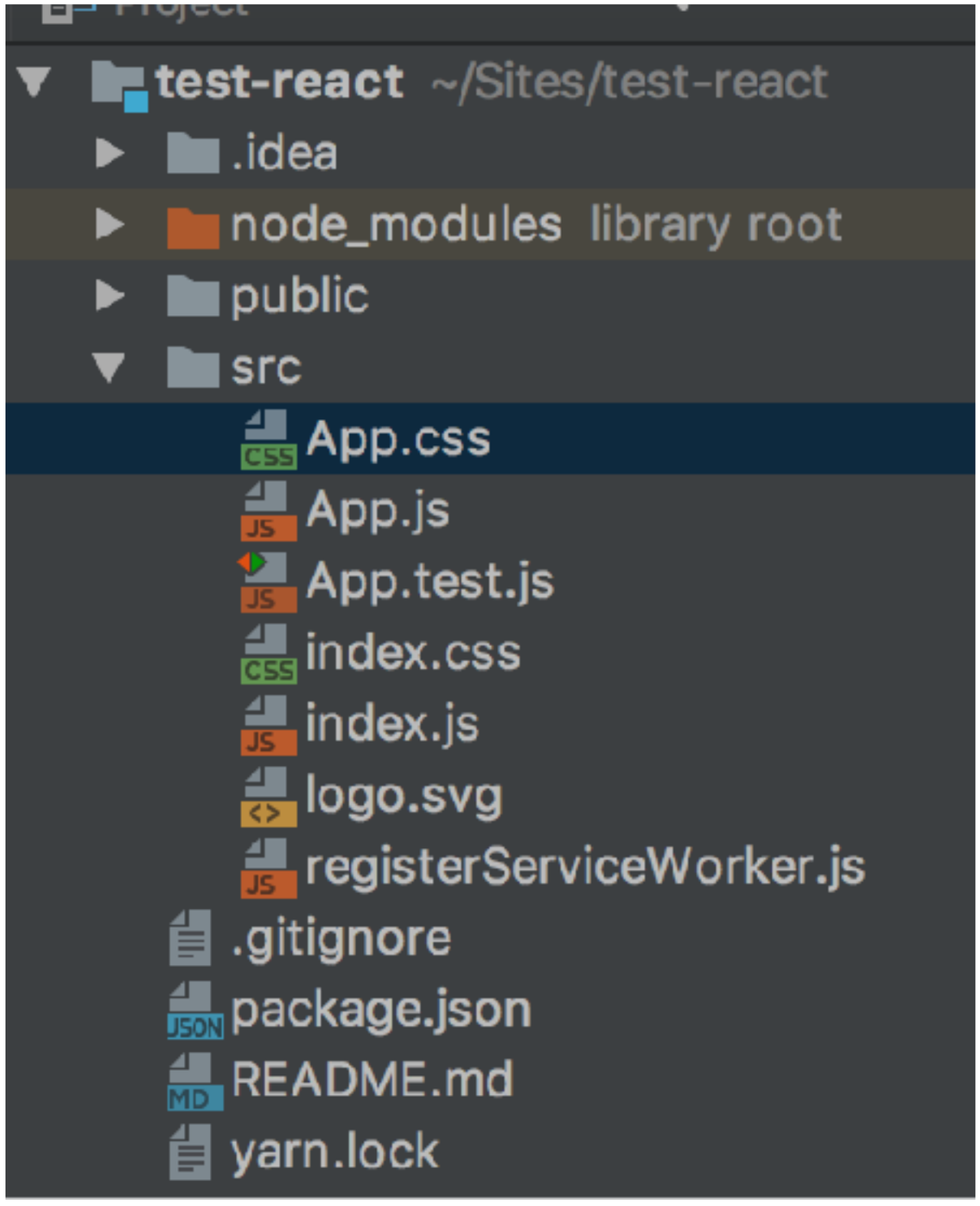
- Installation de `create-react-app`

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

- Création d'une application React

```
$ create-react-app my-app
```

Projet React



Mise en place

$ npm start

```
Compiled successfully!

You can now view test-react in the browser.

  Local:            http://localhost:3000/
  On Your Network:  http://192.168.1.27:3000/

Note that the development build is not optimized.
To create a production build, use yarn build.
```

React : Props

- Props = Arguments du composants

- ⚠ Immuable ⚠

React : Props

```
class Hello extends Component {
    render() {
        return (
            <div>
                <h1>Hello, {this.props.name}</h1>
            </div>
        );
    }
}

<Hello name="Yohan" />
```

PropTypes

- Validation des props

- Evite les erreurs

PropTypes

```
import PropTypes from 'prop-types';

class Hello extends Component {
    static propTypes = {
        name: PropTypes.string.isRequired,
    };

    render() {
        // …
    }
}
```

Javascript / JSX

```
<Hello name={'Yohan' + ' ' + 'Giarelli'} />

<div>{Math.random()}</div>

{[1, 2, 3, 4, 5].map(number => (
    <h2>{number * 2}</h2>
))}
```

Événements

- Événements React gérés directement dans les composants (en props)
- Pas de « addEventListener » ou équivalent
- Très simple, mais demande de l’organisation

Événements

```
<button onClick={() => alert('Coucou !')}>
    Cliquez-moi
</button>
```

Événements

```
class Hello extends Component {
    onButtonClick() {
        alert('coucou');
    }

    render() {
        return (
            <div>
                <button onClick={this.onButtonClick}>
                    Cliquez-moi
                </button>
            </div>
        );
    }
}
```

State

- Contient l’état du composant
- Peut-être modifié (contrairement aux props)
- À utiliser avec parcimonie

State

```
class Hello extends Component {
    state = {
        active: false,
    };

    render() {
        return (
            <div>
                <h2>{this.state.active ? 'Actif' : ‘Inactif'}</h2>

                <button onClick={() => this.setState({active: true})}>
                    Activer
                </button>
                <button onClick={() => this.setState({active: false})}>
                    Désactiver
                </button>
            </div>
        );
    }
}
```

Application

- Conception d’un composant TicTacToe (jeu de morpion)