REACT JS

A destination des développeurs front.

Objectifs de la formation :

« Apprendre à développer avec React JS et son écosystème. »

SOMMAIRE

JOUR 1

SOMMAIRE

JOUR 2

SOMMAIRE

JOUR 3

FONCTIONNEMENT DE FORMATION

Beaucoup de pratique !

- Des petits exercices

- Un fil rouge : créer une application qui gère des pneus / marques

TOUR DE TABLE

MAREK ALGOUD

Lead creative dev / scrumaster

Chez Smile depuis 2013

Travaille sur des projets **React** / Vue / Angular depuis + de 2 ans

… Parle trop vite

Et vous ?

INTRODUCTION

SMILE

I.T IS OPEN

HISTORIQUE

Créer par facebook :

2011 - Problème de maintenance du code

➔ FaxJS, prototype de react

2013 – lancement de react

2015 – react est stable, et est utilisé par

- Airbnb
- Netflix

POURQUOI UTILISER REACT JS ?

- Composants !

- Haute performance (virtualDOM, reconcilation)

- Reactive Native pour les applications mobiles

- Grande communauté

- Maintenu par facebook

QUI UTILISE REACT

Facebook

Instagram

Airbnb

Netflix

Paypal

Reddit

Ubber

…

ALTERNATIVES

Rappel es 6+

RAPPEL ES 6+

Edition	Nom officiel	Date de publication
ES9	ES2018	Juin 2018
ES8	ES2017	Juin 2017
ES7	ES2016	Juin 2016
ES6	**ES2015**	**Juin 2015**
ES5.1	ES5.1	Juin 2011
ES5	ES5	Décembre 2009
ES4	ES4	Abandonné
ES3	ES3	Décembre 1999
ES2	ES2	Juin 1998
ES1	ES1	Juin 1997

RAPPEL ES 6+

1/ primitive / object reference

2/ let / const / var

3/ arrow functions

4/ Export / import

5 / Classe et héritage, propriété et méthode

6/ Spread / rest operator

7/ Destructing

8/ Fonction sur les tableaux : filter / map / find / reduce

C’EST PARTI !

Créons notre première application react JS

SMILE
I.T IS OPEN

JUSTE AVANT DE COMMENCER…

Différentes façon de commencer un projet

Create-react-app => recommandé par facebook pour démarrer un projet

Mais aussi …

Next-js => server side rendering

Gatsby => static site generator

JUSTE AVANT DE COMMENCER…

IDE : VS CODE

Visual studio code adapté aux projets front :

- Leger

-Market place riche

- Personnalisable

- Auto complétion

- Accès au terminal directement dans l’éditeur

- etc…

C’EST PARTI !

Vérifions notre version de node npm

```
node  -v
npm  -v
```

➔ node v8+

Créons notre premier projet hello-world

```
npx create-react-app my-app
cd my-app
npm start
```

➔ Ouvrir l’url http://localhost:3000 dans le navigateur

ARCHITECTURE DE BASE

CREATE-REACT-APP

Package.json

```
"scripts " : {
 "start " : "react-scripts start",
 "build "  : " react-scripts build",
 "test " : "react-scripts test ",
" eject " : "react-scripts eject "
}
```

Philosophie :

- Une seule dépendance (qui regroupe webpack, babel, ESLint, etc…)
- Pas de configuration requise
- Pas bloquant (possibilité de faire un eject pour faire du fine-tunning)

ARCHITECTURE DE BASE

Regardons de plus près le contenu du répertoire src

```
App.css
App.js
App.test.js
Index.css
registerServiceWorker.js
```

JSX

sugar syntax pour react.createElement

https://reactjs.org/docs/react-api.html#createelement

```
return React.createElement('div', {'className': 'App'},
      [
        React.createElement('img', { 'src': logo, 'alt': 'logo', 'className': 'App-logo' }),
        React.createElement('h1', null, `Hello world !`)
      ]
    )
```

Restriction

- Element englobant obligatoire
- Attribut class devient className (c'est du JS!)
- Attribut en camel case (ex: onClick et non onclick)
- La liste des propriétés des composants du DOM est disponible ici : https://reactjs.org/docs/dom-elements.html

CRÉER UN COMPOSANT FONCTIONNEL

Composant fonctionnel :

Fonction qui renvoi du code jsx.

Ex :

```
import React from ‘react’;

const MonComposant = () => (<h1>Hello World ! </h1>);

export default MonComposant;
```

Convention de nommage : un composant commence toujours par une majuscule.

CRÉER UN COMPOSANT FONCTIONNEL

Créer un composant « article »

- Pas de props
- Et l’insérer 2 fois en page d’accueil

Ajouter du style

- Inline
- Via import de fichier

CRÉER UN COMPOSANT FONCTIONNEL

Passer des paramètres au composant

- id
- Title
- content
- Category

```
import React from ‘react’;

const MonComposant = (props) => (<h1>Hello {props.name} ! </h1>);

export default MonComposant;
```

Propriété spéciale : children

CRÉER UN COMPOSANT FONCTIONNEL

Afficher la catégorie que si celle-ci existe

1 / operateur ternaire :

()? <Composant />: null

2/ exp && <Composant />

3/ If externe

COMPOSANT CLASSE

Transformer le composant article en class

```
class Article extends React.Component  {
  constructor(props) {
    super(props);
  }
  render()  {
    return /* Some JSX*/ ;
  }
}
```

➔ Les propriétés sont accessibles avec this.props

AJOUTER DU STYLE

- Ajouter un fichier externe

```
Import ‘./article.css’
```

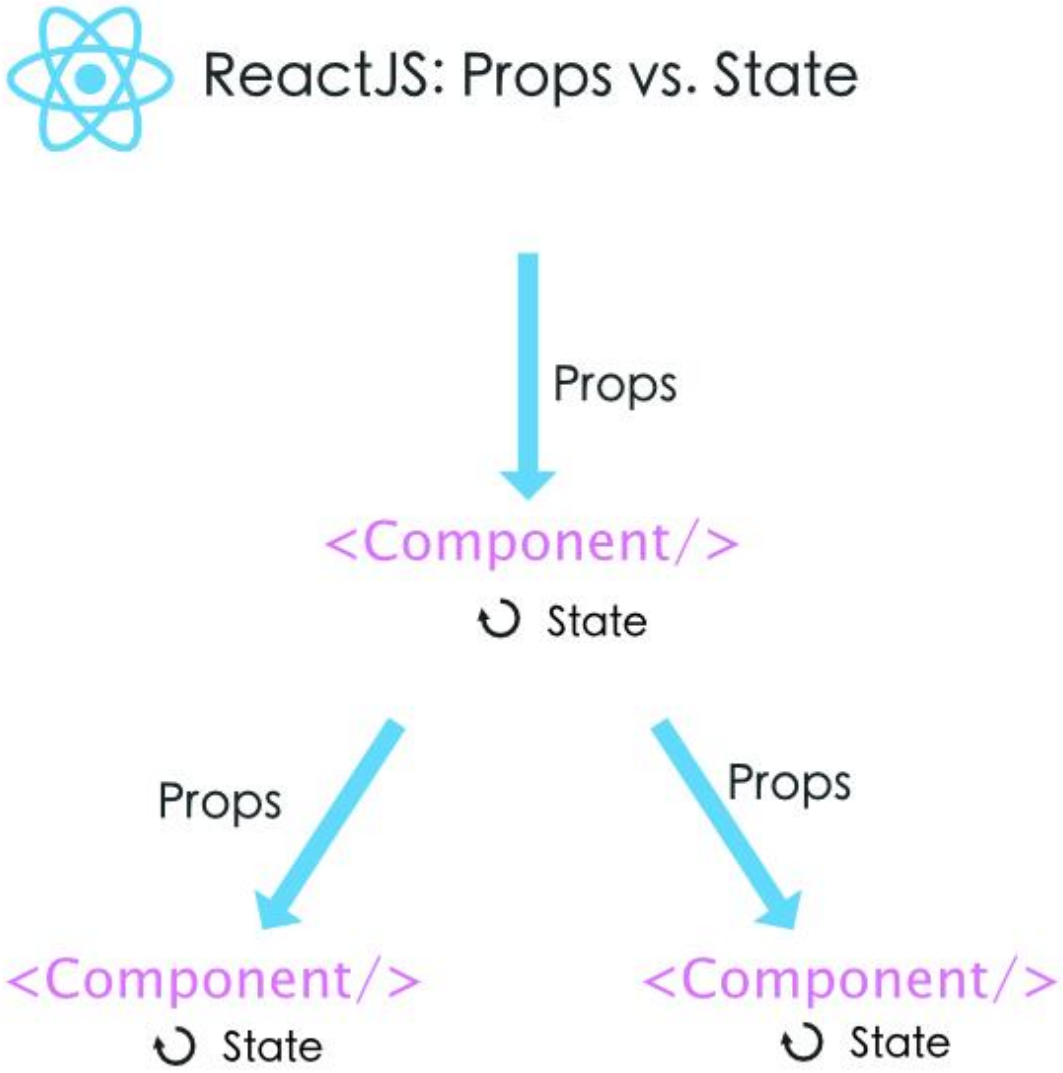
- Mettre l’article en background:red

- Faire la même chose avec du css inline

```
const style = {backgroundColor: red}
```

- Passer la couleur dans les propriétés

STATE


ReactJS: Props vs. State
Props
<Component/>
State
Props
<Component/>
State
Props
<Component/>
State

STATE

- Rajouter dans le « state » la variable published avec pour valeur initiale false

- Si l’article n’est pas publié, afficher « brouillon », sinon afficher « publié »

- Ajouter un bouton qui change l’état

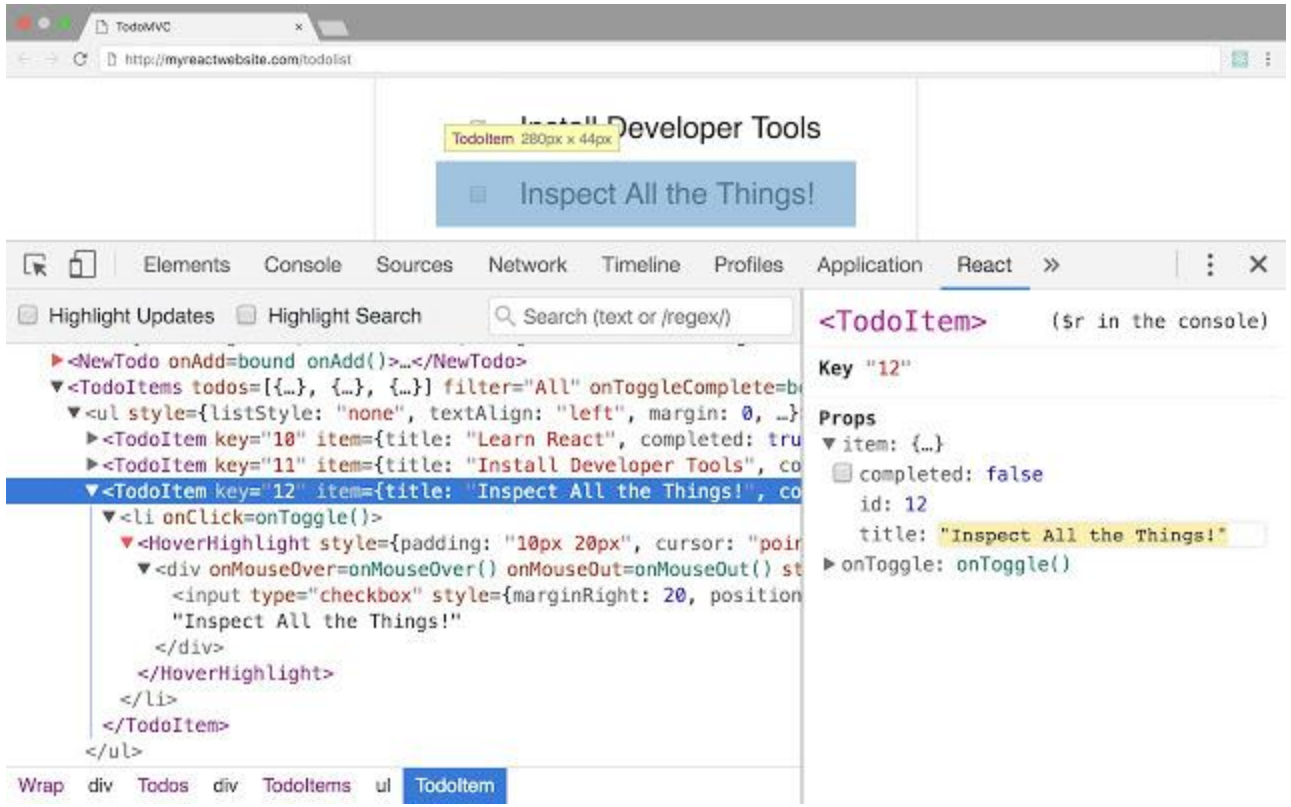
STATE

- Ajouter un second bouton « compteur » et afficher le nombre de click sur le bouton

- Appeler deux fois le this.setState dans la même fonction

DEBUGGING

https://chrome.google.com/webstore/detail/react-developer-tools



UN PEU DE REFACTO…

- Bonne pratique :
    - Utiliser autant que possible des « fonction components »
    - Mettre la logique métiers dans des « containers » qui s'occupe de l'état de l'application (où d'une partie de l'application)

➔ Transformer le composant article en « fonction component »

➔ Mettre dans le composant App la liste des articles

FORM, INPUT, TWO WAY BINDING

- rajouter un input type text dans l’article qui permet de changer le nom de l’article

LISTE

- Créer une liste d’article

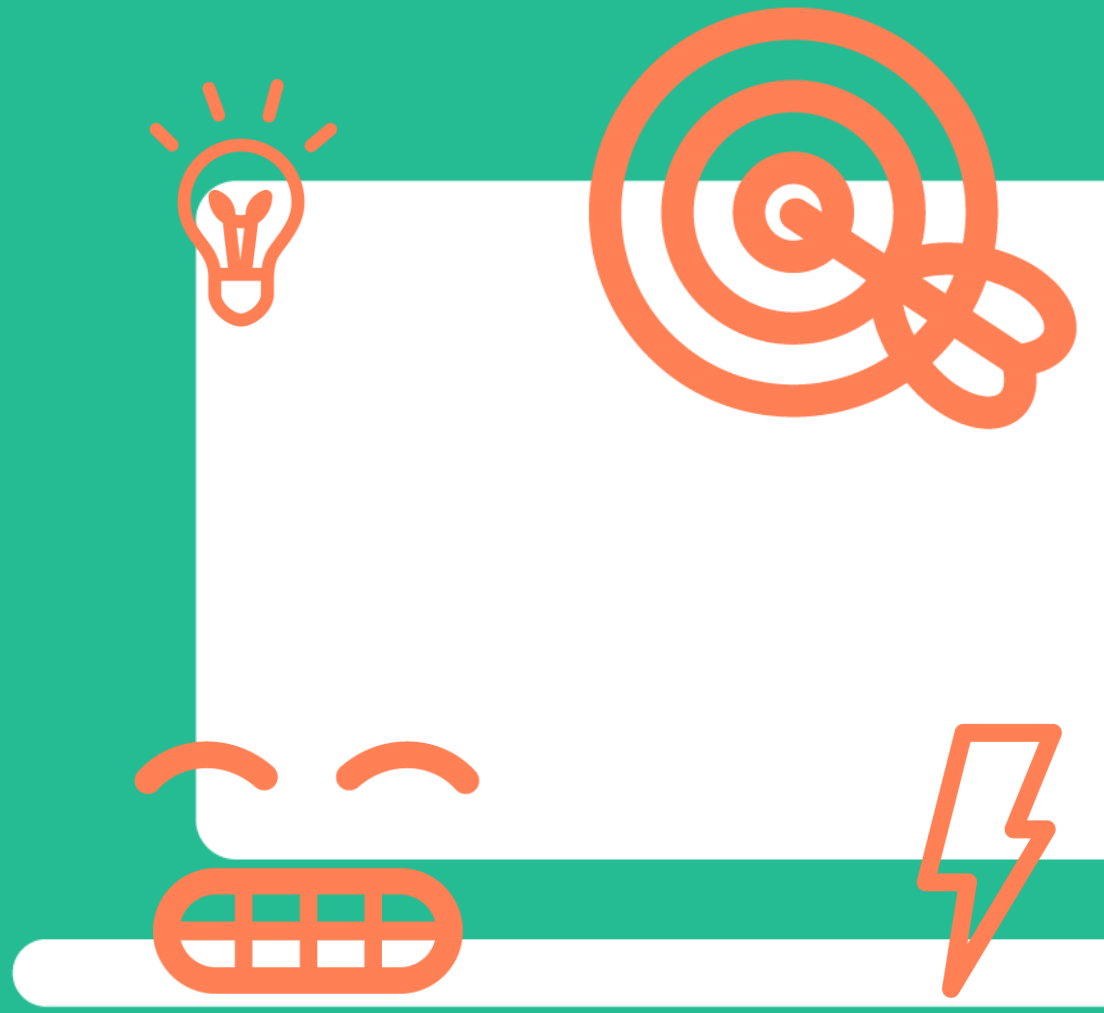
- Mettre à jour la fonction qui permet de changer le titre d’un article

- Ajouter une fonction qui au clic sur un bouton supprimer d’un article le supprime de la liste

- …. Key props

Exercice 1

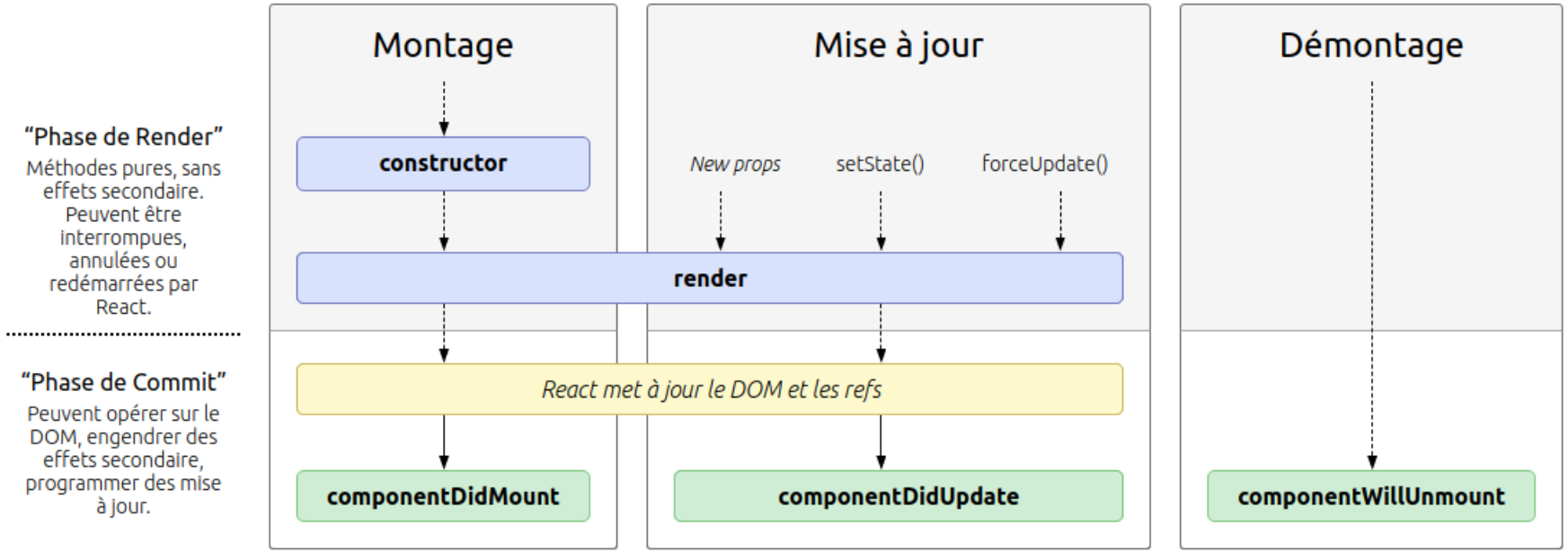
FONCTIONNALITÉS AVANCÉES



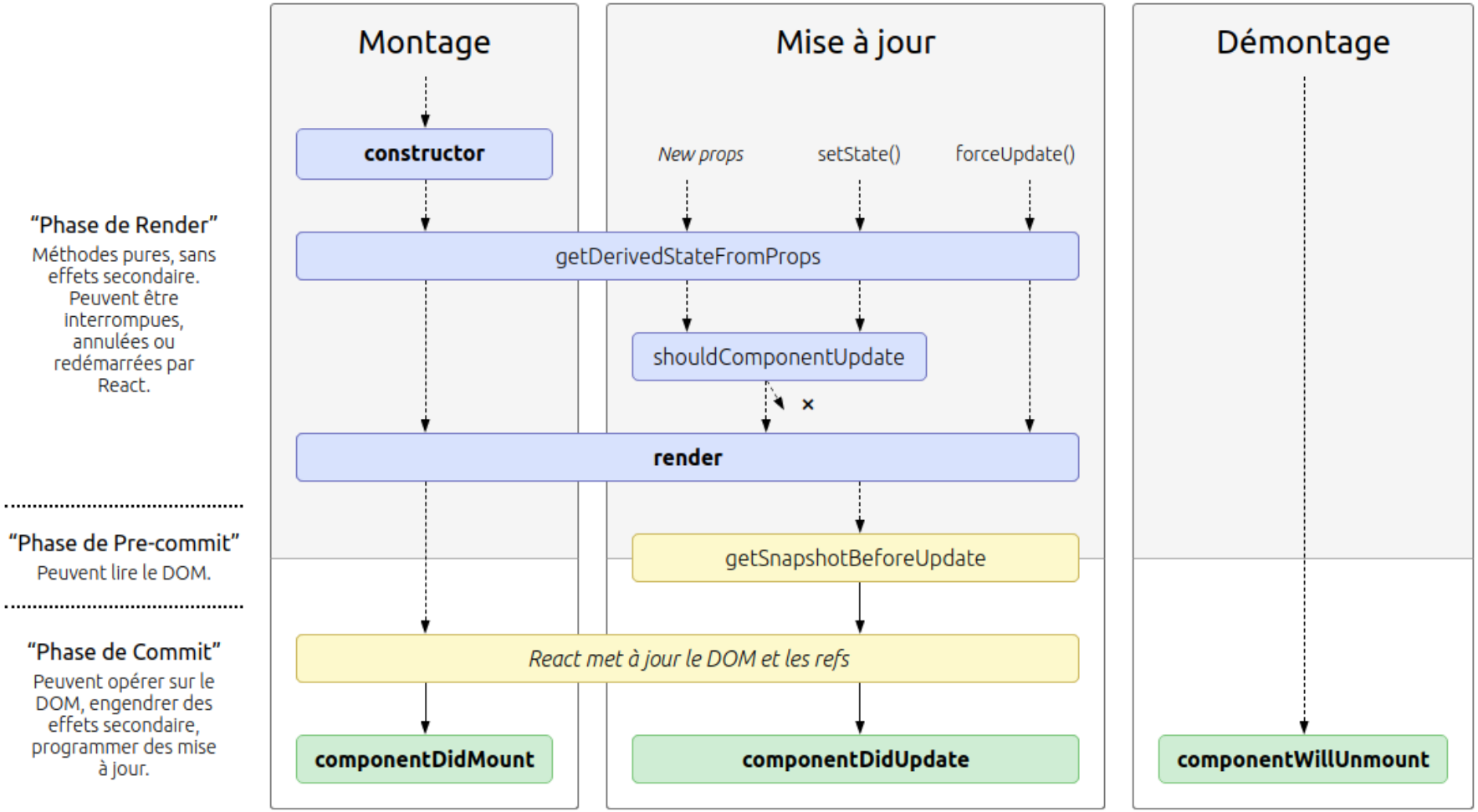
SMILE
I.T IS OPEN

LIFE CYCLE

http://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/



LIFE CYCLE


Montage
Mise à jour
Démontage
constructor
New props
setState()
forceUpdate()
getDerivedStateFromProps
shouldComponentUpdate
render
getSnapshotBeforeUpdate
React met à jour le DOM et les refs
componentDidMount
componentDidUpdate
componentWillUnmount
"Phase de Render"
Méthodes pures, sans effets secondaire. Peuvent être interrompues, annulées ou redémarrées par React.
"Phase de Pre-commit"
Peuvent lire le DOM.
"Phase de Commit"
Peuvent opérer sur le DOM, engendrer des effets secondaire, programmer des mise à jour.

Exercice 2

PURE COMPONENT

Si `shouldComponentUpdate(nextProps, nextState)`

Ne fait qu'une comparaison complète de nextProps avec Props et de nextState avec state, alors, il est possible d'étendre la classe « PureComponent »

VIRTUAL DOM

shouldComponentUpdate()
ok !

Render()

Virtual Dom

Current

…

Next

…

Comparison

Différences ?

Oui ! On ne met
à pas jour le DOM

Oui ! On met à jour
Le vrai DOM

Plus rapide que
le vrai DOM,
On ne met pas
à jour le dom
directement

Reprendre l’exercice 2 et utiliser les PureComponent

HOC

High order component : Pas une fonctionnalité react, mais un pattern bonne pratique

Deux écritures possibles

```
<EnhancedComponent>
                <WrappedComponent />
<EnhancedComponent>
```

```
const EnhancedComponent =
higherOrderComponent(WrappedComponent);
```

Mise en pratique : créer un HOC qui permet d’enlever le div englobant

FRAGMENT

Fragment est un HOC : il permet de supprimer le div englobant

```
<React.Fragment>
    <td>valeur 1</td>
</React.Fragment >
```

ou

```
<>
    <td>valeur 1</td>
</>
```

React.Fragment permet de rajouter l’attribut key dans le cas de liste

Mise en pratique : créer un HOC qui permet d’ajouter un div avec une classe reçue en param

AXIOS

Axios : Promise based HTTP client for the browser and node.js

https://github.com/axios/axios

Pour l'installer

```
npm install axios --save
```

Deux écritures possibles

```
axios.get('/user?ID=12345')
  .then(function (response) {
    // handle success
    console.log(response);
  })
  .catch(function (error) {
    // handle error
    console.log(error);
  })
```

```
async function getUser() {
  try {
    const response = await
axios.get('/user?ID=12345');
    console.log(response);
  } catch (error) {
    console.error(error);
  }
}
```

Doit être fait dans le componentDidMount() / componentDidUpdate()

Mise en pratique : exercice 3

PROPS-TYPE

Props-type : permet de valider que le composant reçoit le bon type te de propriétés en entrée, sinon soulève des warnings dans la console

➔ Augmente la qualité, utile pour le travail en équipe

Pour l'installer

```
npm install --save prop-types
```

Utilisation : uniquement dans des classes

```
MyComponent.propTypes = {
  title: PropTypes.string.isRequired
  published: PropTypes.bool.isRequired,
  number: PropTypes.number
}
```

Mise en pratique : Rajouter props-type sur Brand

Accéder au DOM créé dans la fonction render()

```
class MyComponent extends React.Component {
  constructor(props) {
    super(props);
    this.myRef = React.createRef();
  }
  render() {
    return <div ref={this.myRef} />;
  }
}
```

Note : uncontrolled component :

https://reactjs.org/docs/uncontrolled-components.html

UNCONTROLLED COMPONENT

https://reactjs.org/docs/uncontrolled-components.html

Utilise les références au lieu des handler pour controller la valeur des champs de formulaire.

Recommandé pour les formulaires simples

```
class NameForm extends React.Component {
  constructor(props)  {
    super(props);
    this.input = React.createRef();
  }

  handleSubmit = (event)  => {
    alert('A name was submitted: ' + this.input.current.value);
}

  render() {
    return (
      <form onSubmit={this.handleSubmit}>
        <label>    Name:  <input type="text" ref={this.input} />   </label>
        <input type="submit" value="Submit" />
      </form>
    );
  }
}
```

Mise en pratique : Mettre le focus sur le champ de recherche au chargement de la page

PORTAL

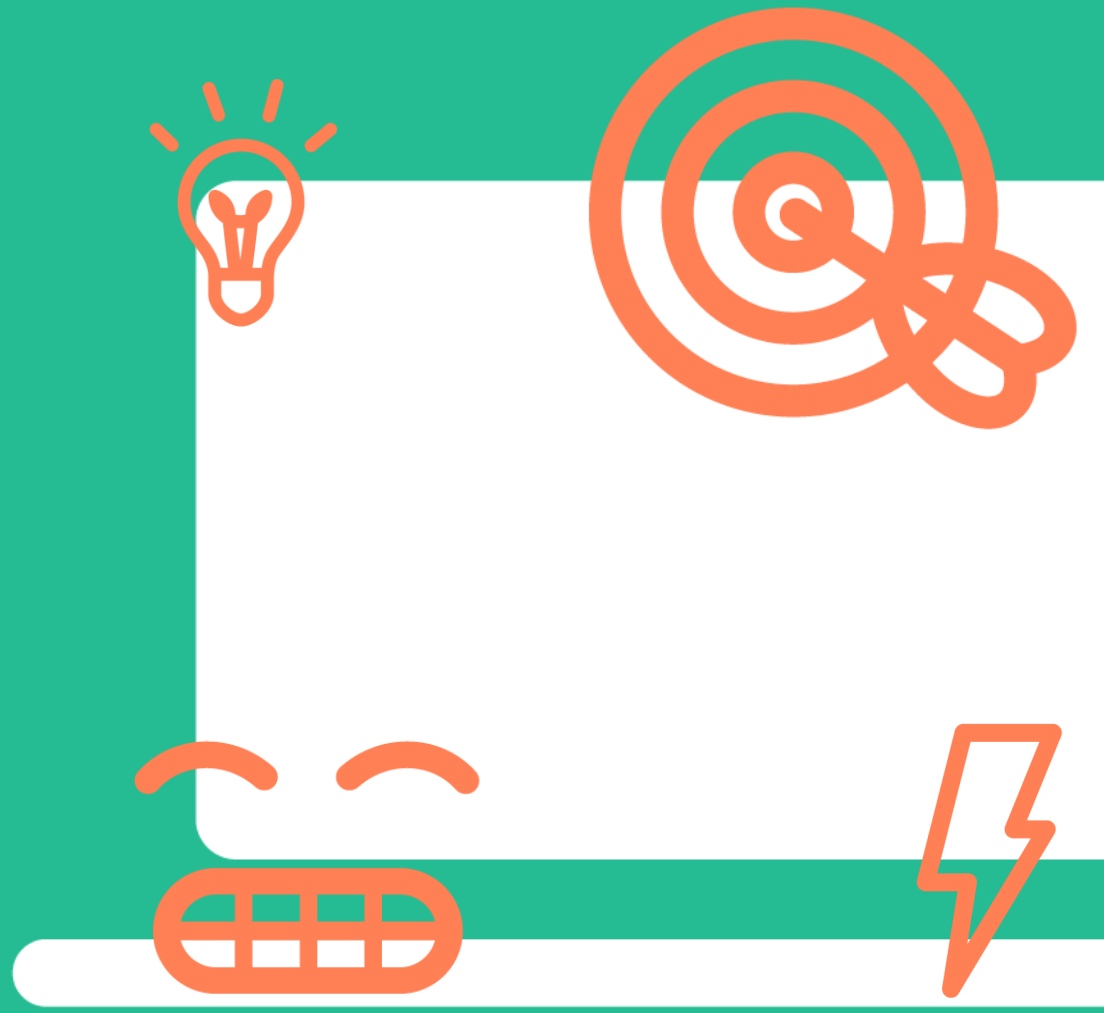
Permet d’afficher du DOM hors de l’arborescence parente

https://reactjs.org/docs/portals.html

Mise en pratique remplacer l’alerte de suppression d’une marque par une modale

Il est temps de faire un peu de refacto…
Exercice 4 !

TESTS



SMILE

I.T IS OPEN

JEST

Jest : solution recommandée par facebook pour faire les tests

Embarqué nativement dans create-react-app

Basé sur Jasmine

Inclus également Istanbul pour faire le code coverage

Compatible avec Enzyme (airbnb) ou **react-testing-library**

Fichier tests :

- Tous les fichiers présents dans des répertoires __tests__
- Fichier xxx.test.js
- Fichier xxx.spec.js

JEST

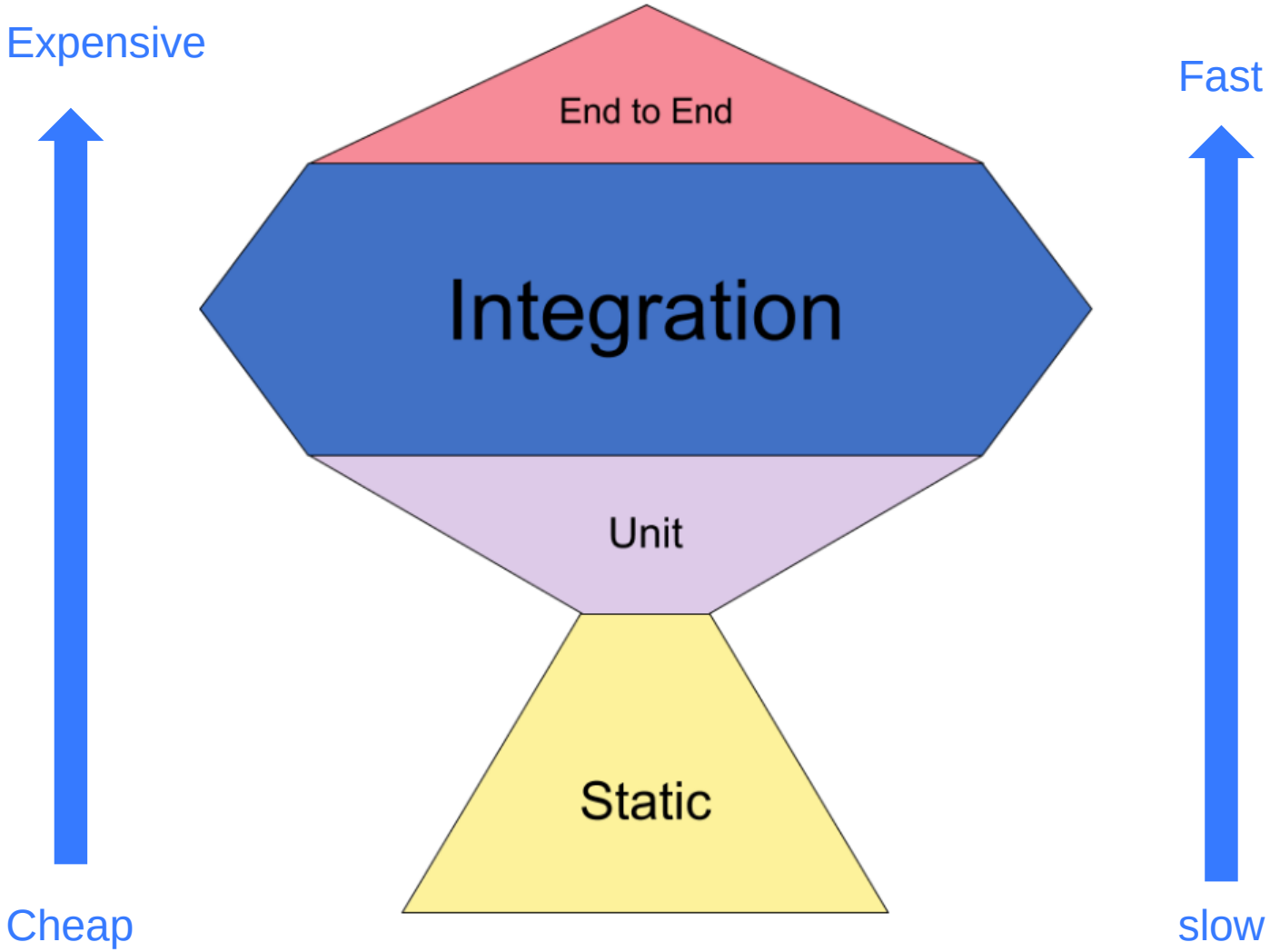
Différents type de tests :

- Tests statiques → ES Lint
- Tests unitaires
- Tests d'intégration

} Jest

- Test end-2-end → Cypress

The Testing Trophy



JEST

SNAPSHOT

Créer un rendu du composant, et le compare au précédent. S'il a changé, il soulève une erreur

➔ A besoin de react-test-renderer

```
import React from 'react';
import Link from '../Link.react';
import renderer from 'react-test-renderer';

test('Link changes the class when hovered', () => {
  const component = renderer.create(
    <Link page="http://www.facebook.com">Facebook</Link>,
  );
  let tree = component.toJSON();
  expect(tree).toMatchSnapshot();
});
```

Faire un snapshot pour le composant Brand

JEST

STRUCTURE D’UN TEST

```
Describe(‘<Component />’, () => {

        it(‘ should render title’, () => {

        // arrange

        // act

        // assert

        });

});
```

Ecrire nos premiers tests

JEST

1er test :

Tester le composant Brands (liste des marques)

1 – si pas de marque passée en param

2 – si un tableau de marque est passée en param

2eme test :

Tester le composant Brand (affichage d’une marque)

- Vérifier qu’un clic appelle bien la fonction passée en param

Mocking module :

Jest.mock(…)

Ex : il est possible de mocker le module axios

Jest.mock(axios)

Il est possible d’affecter une valeur de retour directement dans les tests

```
const resp = {data: [{name: 'Bob'}]};
  axios.get.mockResolvedValue(resp);
```

Ou de créer un répertoire __mock__ qui contient les versions mockées des modules

JEST

React-testing-library :

Philosophie : écrire les tests comme si c'était un utilisateur qui testait l'application / le composant.

Render : wrapper qui permet de « rendre » le composant dans un container qui sera ajouter au document.body.

fireEvent : déclenche un événement sur un node

Cleanup : permet de démonter et de nettoyer le dom, à faire généralement après chaque test (dans un afterEach)

waitForElement : permet de tester les fonctions asynchrone. Attend 5 sec (timeout personnalisable) avant de soulever une erreur si l'élément n'est pas trouvé dans de DOM.

JEST
ZOOM SUR RENDER

React-testing-library : la fonction render :

Renvoie :

- container : le container du composant à tester (DOM)

- getByText : utilitaire qui permet de trouver un node contenant un text

- getByLabelText : renvoi l'input qui à un label associé qui contient le texte recherché

- getByTestId : permet de retrouver un node qui a un attribut data-testid correspondant au pattern de recherche

3eme test :

Tester le composant App

- Vérifier que le composant appelle bien l’API

- Vérifier que le l’API delete est bien appelée sur le clic sur une marque

Exercice : faire les tests pour la partie Pneu

Test end-2-end

```
npm install cypress –save-dev

Npx cypress open
```

Exercice 5 : tester le filtre sur les pneus

ROUTES

SMILE

I.T IS OPEN

ROUTER

Plusieurs solution pour faire le routing avec React :

- La plus connue : react-router
(https://github.com/ReactTraining/react-router)

- L’alternative qui monte : reach-router
(https://github.com/reach/router)

REACT-ROUTER

Plusieurs type de router :

<BrowserRouter> => Si le serveur est configuré pour gérer les redirections

<HashRouter> => Rajoute un # dans l'url, utile lorsque le serveur ne gère par les redirection

<StaticRouter> => Pour le rendu coté serveur (voir partie sur l'isomorphism)

REACT-ROUTER

Les routes :

Path : si l'url match avec path, alors la route est affichée

Exact : Si exact, il faut que l'url match exactement path pour charger la route

Component: composant à afficher

```
<Route exact path="/article/:id" component={ArticleDetail} />
```

Render : si l'on souhaite passer des params, utiliser render plutôt que component

```
<Route exact path="/article/:id" render={() => <ArticleDetail param1=true />} />
```

Plusieurs Façon de faire un lien:

```
<Link to=“/” />
```

Ou

```
this.props.history.push(‘/’)
```

➔ Si le composant n’est pas rendu par ReactRouter directement importer le HOC withRouter et l’appliquer sur le composant

REACT-ROUTER

HELMET

Permet de modifier les headers sur chaque page

```
npm install --save react-helmet
```

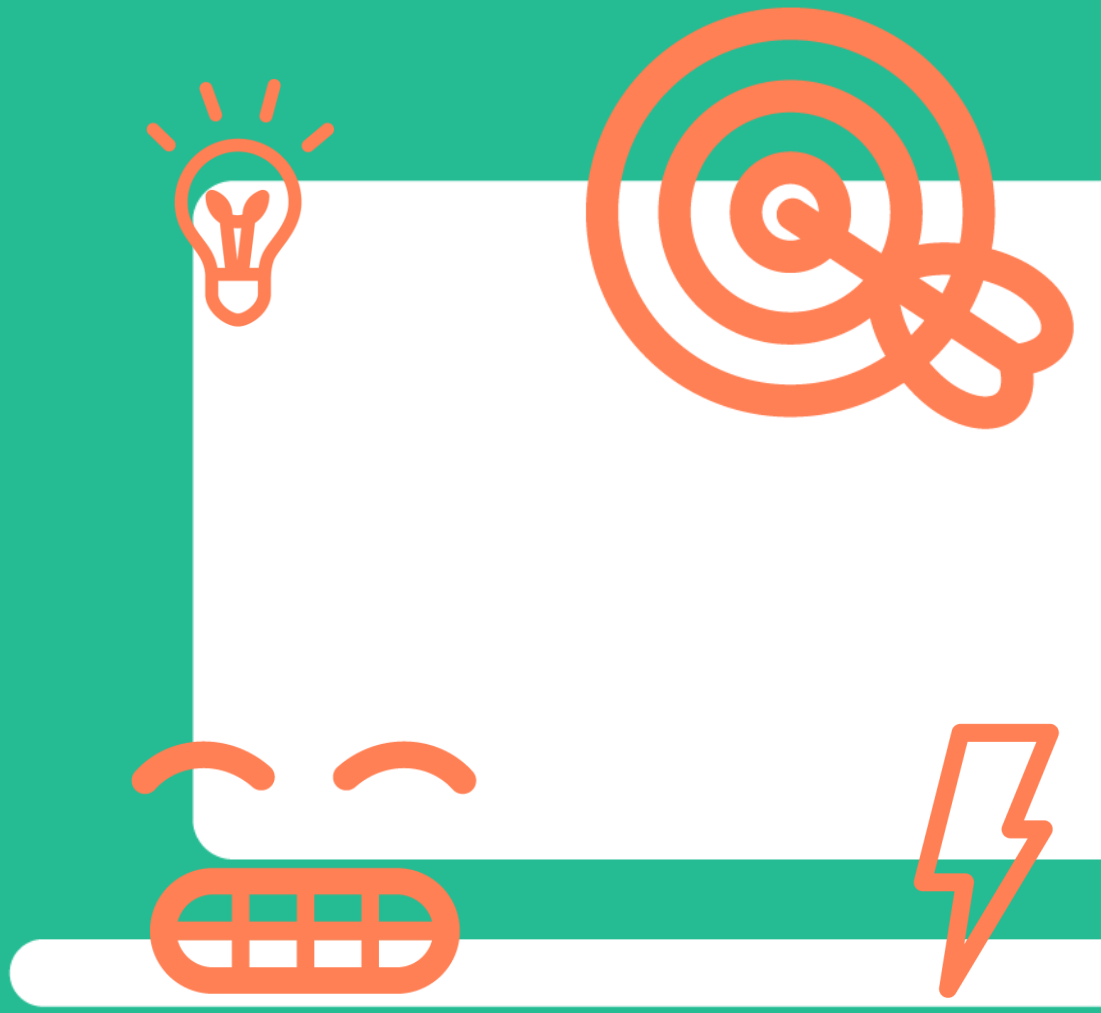
Puis, dans les composants

```
<Helmet>
        <title>My Title</title>
        <meta name="description" content="Helmet application" />
 </Helmet>
```

Mise en place de react-router

Exercice 6

REDUX



SMILE

I.T IS OPEN

REDUX

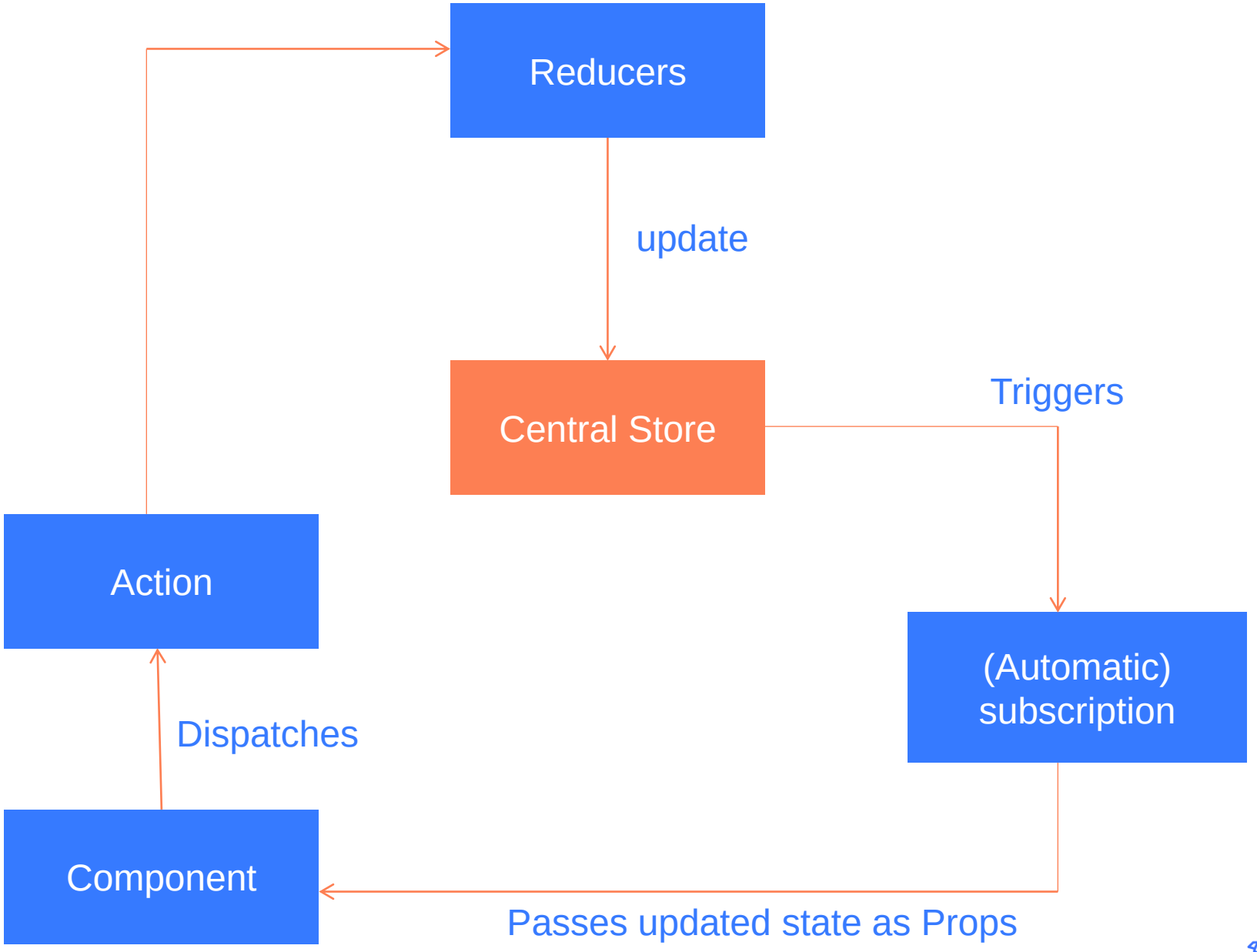
Avantages :

- Prédictif
- Testable
- Outils de débogage avancé (time travelling)

Inconvénient :

- Ajoute de la complexité
- nécessite de la discipline

REDUX
QU'EST CE QUE C'EST ?



REDUX

QUAND L'UTILISER ?

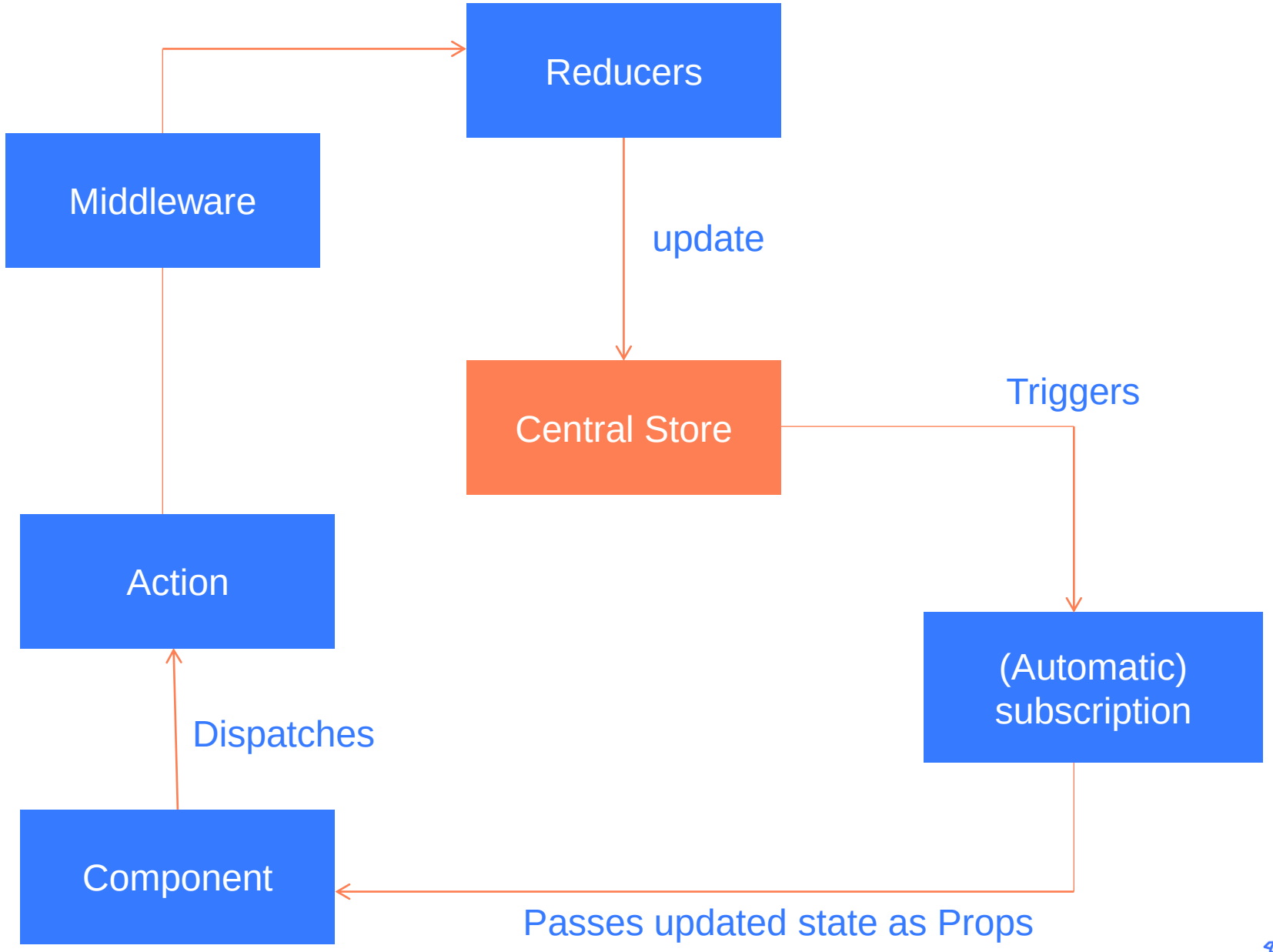
Type	Exemple	Redux ?
Local UI state	Show / hide dropdown	Managed by components (mostly)
Persistent state	All users, all posts, all categories, …	Stored on server, relevant slice managed by redux
Client state	Is authenticated ? Filters set by user, …	managed by redux

Mise en place de redux

Mise en place de multiple reducer

REDUX

MIDDLEWARE

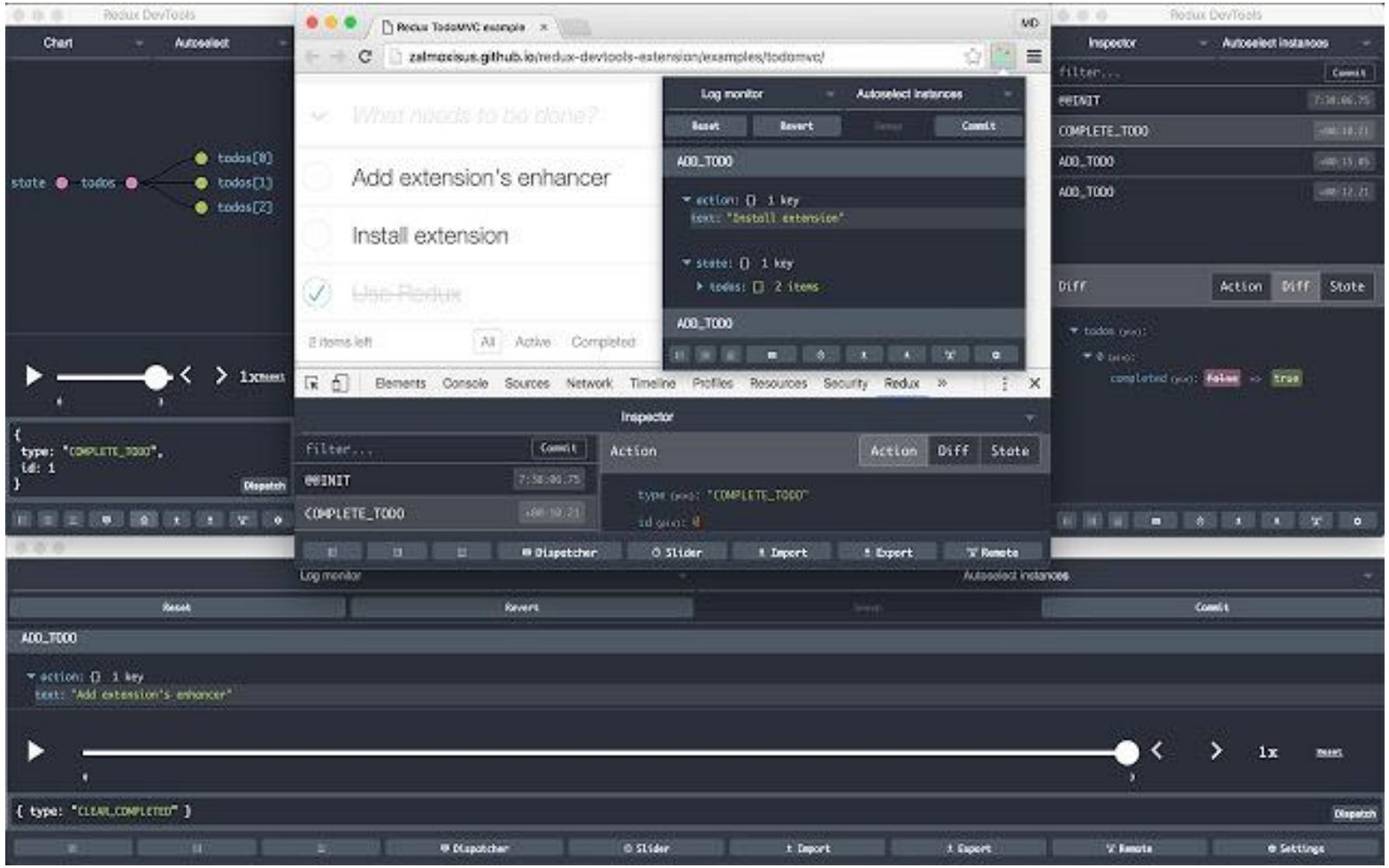


Mise en place d’un middleware « logger »

REDUX

REDUX DEV TOOLS

https://github.com/zalmoxisus/redux-devtools-extension



Mettre en place redux devTools

REDUX

REDUX THUNK

Ajoute un middleware qui permet de faire des actions asynchrone

https://github.com/reduxjs/redux-thunk

What's a thunk?!

A thunk is a function that wraps an expression to delay its evaluation.

```
// calculation of 1 + 2 is immediate
// x === 3
let x = 1 + 2;

// calculation of 1 + 2 is delayed
// foo can be called later to perform the calculation
// foo is a thunk!
let foo = () => 1 + 2;
```

The term originated as a humorous past-tense version of "think".

Mettre en place une action asynchrone

Exercice 7

REDUX

ALTERNATIVE

Context : fonctionnalité de react qui n’est plus en version expérimentale depuis la version 16+

Permet de partager des données à l’ensemble des composants d’un arbre

https://reactjs.org/docs/context.html#when-to-use-context

CODE SPLITTING

SMILE
I.T IS OPEN

CODE SPLITTING

React-loadable

https://github.com/jamiebuilds/react-loadable

A **higher order component** for loading components with dynamic imports.

```
import Loadable from 'react-loadable';

const LoadableComponent = Loadable({
  loader: () => import('./my-component'),
  loading:  <h1> Chargement …. </h1>,
});

export default class App extends React.Component {
  render() {
    return <LoadableComponent/>;
  }
}
```

Exercice 8 : Mettre en place le code splitting sur les composants le nécessitant

ISOMORPHISM

SPA + SSR == ISOMORPHISM

SMILE
I.T IS OPEN

EXERCICE 9 : MISE EN PLACE DU SSR

L’isomorphisme est « built-in » dans react grace au Virtual DOM.

Il faut mettre en place un serveur node (puisque react, c’est du js) qui va parser le code React, extraire le Virtual Dom (grace à la fonction **renderToString**) et le renvoyer au client :

```
import ReactDOMServer from 'react-dom/server';
const html = ReactDOMServer.renderToString((
<div>
<Component />
</div>
));
```

Exercice 9 : mise en place du SSR

EXERCICE 9 : MISE EN PLACE DU SSR

Render vs hydrate

Une fois que l'application est rendue coté serveur, le faut remplacer la fonction React.render par react.hydrate

https://reactjs.org/docs/react-dom.html#hydrate

EXERCICE 9 : MISE EN PLACE DU SSR

Suppression des appels aux variables du browser

Windows et document sont des variables spécifiques au navigateur, elle ne sont pas disponible coté serveur. Il faut donc déplacer leur utilisation pour ne pas avoir d'erreur :

- Mettre dans index.js l'appel du router (qui utilise history)

- Mettre dans les componentDidMount les appels à window ou document

CSS MODULE

SMILE

I.T IS OPEN

CSS MODULE

Permet d’avoir classes limité au composant automatiquement

Convention de nommage : [name].module.css

Ex :

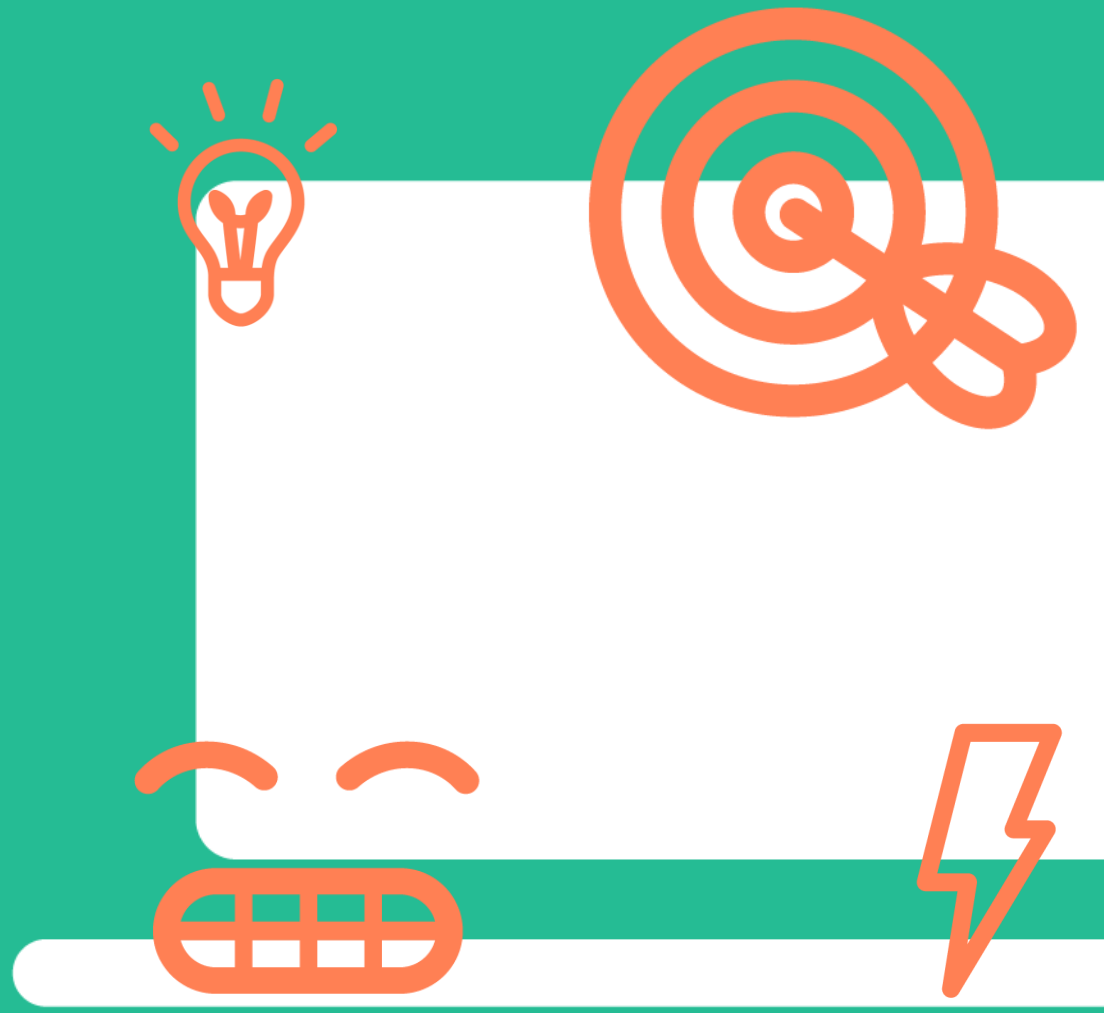
```
// button.module.css

.error {
  background-color: red;
}
```

```
// Button.js

import styles from './Button.module.css'; // Import css modules stylesheet as styles
import './another-stylesheet.css'; // Import regular stylesheet

class Button extends Component {
  render() {
    // reference as a js object
    return <button className={styles.error}>Error Button</button>;
  }
}
```

Note : il est possible d’utiliser sass : https://facebook.github.io/create-react-app/docs/adding-a-sass-stylesheet

Exercice 11 : mise en place de css.module

STORYBOOK



SMILE

I.T IS OPEN

STORYBOOK

Storybook est un environnement de développement pour les composants graphique :

https://github.com/storybooks/storybook

Il impose les bonnes pratiques de séparation des composant « stateful » / « stateless »

Il permet de tester le rendu des composants avec différentes propriétés

STORYBOOK

Storybook est un environnement de développement pour les composants graphique :

https://github.com/storybooks/storybook

Il impose les bonnes pratiques de séparation des composant « stateful » / « stateless »

Il permet de tester le rendu des composants avec différentes propriétés

Exercice : mise en place de storybook

Exercice : mise en place de storybook : composant avec Redux et Router

NEXT.JS

BONUS : NEXT.JS

BONUS : NEXT.JS

NEXT.js

Framework créé par la société Zeit.

Permet de créer facilement une application avec le routing, le server side rendering et le code splitting inclus nativement.

Setup :

https://nextjs.org/

I.T IS OPEN