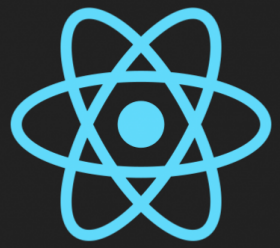


REACT live 4

Formateur :
TAUPIN Jérémy

Composant React

Un composant React est une fonction (actuellement privilégiée par React) ou une classe (méthode utilisée précédemment) qui renvoie un élément React.

Il prend généralement des props (propriétés - Vue plus loin dans ce fichier) en entrée et renvoie une vue en sortie.

Les composants peuvent être réutilisés à travers une application.

Les composants peuvent également être utilisés pour gérer des données de l'application. Les composants enfants peuvent être intégrés dans un composant parent pour créer une structure hiérarchique.

DECLARER UN COMPOSANT AVEC UNE CLASS

Bien que React se dirige aujourd'hui vers des composants fonctionnels, il reste important de savoir lire les composants créés avec une class.

Ces derniers étant très utilisés dans énormément de package node.
Leur déclaration passe par la création d'une class, au nom du composant, un constructeur, et une fonction de rendu.
L'appel aux props (vue plus loin dans ce slide) et aux states (vue dans le prochain slides) se fait via this.
Le composant étant une extension de React.

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

class ClassComponent extends Component {

    handleClick = () => {
        alert(`Hello ${this.props.name}!`)
    }

    render() {
        return (
            <div>
                <p>My name is {this.props.name}</p>
                <button onClick={this.handleClick}>Say
Hello</button></div>
        )
    }
}

export default ClassComponent
```

Explication

Dans un premier temps nous importont React et son éléments Component (grâce à la destructuration)

Ensuite nous déclarons une Class, la notre se nommera ClassComponent, cette dernière sera une extention de Component de React

A l'interieur de celle-ci nous crérons une première fonction en cas de click (handleClick) qui affichera une alert avec Hello, suivi du nom passé en propriété

Une seconde fonction, nommé render, nous permettra de constituer le rendu. celui-ci sera basé sur un return

Ici le rendu contiendra, un paragraph (balise p) avec le texte My name is, suivi du nom passé en propriété. et un bouton qui appellera notre première fonction en cas de click

Enfin, nous exporterons notre composant grâce à export default, afin de pouvoir l'importer ailleurs

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

class ClassComponent extends Component {

    handleClick = () => {
        alert(`Hello ${this.props.name}!`)
    }

    render() {
        return (
            <div>
                <p>My name is {this.props.name}</p>
                <button onClick={this.handleClick}>Say
Hello</button></div>
        )
    }
}

export default ClassComponent
```

DECLARER UN COMPOSANT AVEC UNE FUNCTION

A l'opposé des composants créé par Class JS cités précèdement, les composants fonctionnels n'étendent pas une partie de React.
Ils sont donc plus rapide à concevoir en terme d'import sur des composants simples, et leur création est simplifié.
Si nous reprenons l'exemple précèdent, notre composant sera codé ainsi

```
import React from 'react'

const FunctionComponent = (props) => {
    const handleClick = () => {
        alert(`Hello ${props.name}!`)
    }

    return (
        <div>
            <p>My name is {props.name}</p>
            <button onClick={handleClick}>Say
Hello</button>
        </div>
    )
}

export default FunctionComponent
```

Explication

Dans un premier temps nous importont uniquement React

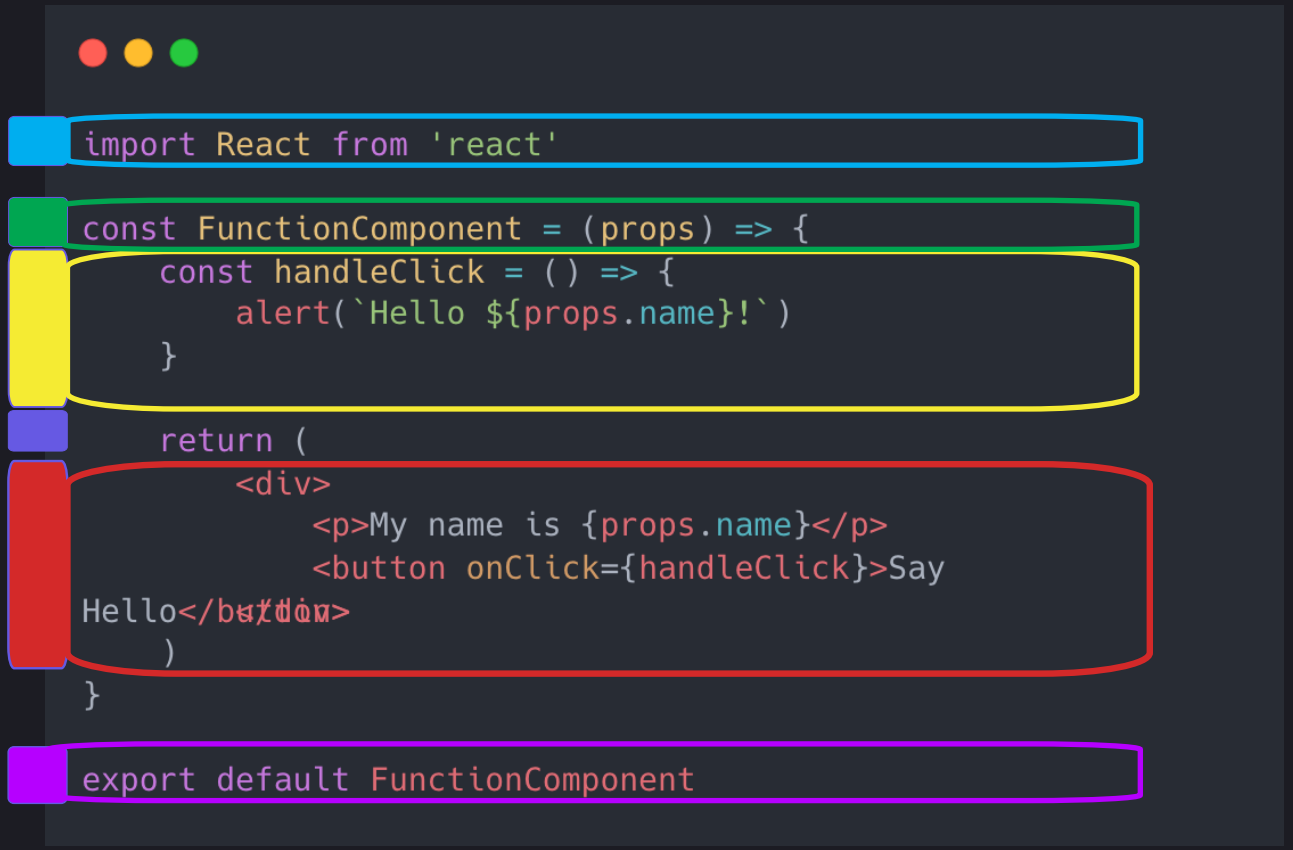
Ensuite nous déclarons une Function qui aura comme propriété props. Ici elle se nomme FunctionComponent

A l'interieur de celle-ci nous crérons une première fonction en cas de click (handleClick) qui affichera une alert avec Hello, suivi du nom passé en propriété

Un return renverra ensuite le contenu de notre composant

Ici le rendu contiendra, un paragraph (balise p) avec le texte My name is, suivi du nom passé en propriété. et un bouton qui appellera notre première fonction en cas de click

Enfin, nous exporterons notre composant grâce à export default, afin de pouvoir l'importer ailleurs



Différences

Evidement les 2 déclarations ont donc des différences. Sur un composant simple (sans gestion de donnée dynamique géré par l'état).

- L'import nécessite React et Component avec un composant par class. Un composant fonctionnel aura uniquement besoin de React et cela vient à disparaitre

- La class étend Component. La fonction est une fonction simple

- Il est nécessaire de déclarer props comme propriété dans un composant fonctionnel

- Il sera nécessaire d'utiliser this dans le composant de class, non nécessaire dans la fonction car c'est un appel à la propriété

- La class aura une fonction render qui contiendra le return, la où la fonction réalise directement le return

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

class ClassComponent extends Component {

    handleClick = () => {
        alert(`Hello ${this.props.name}!`)
    }

    render() {
        return (
            <div>
                <p>My name is {this.props.name}</p>
                <button onClick={this.handleClick}>Say
Hello</button></div>
        )
```

```
import React from 'react'

const FunctionComponent = (props) => {
    const handleClick = () => {
        alert(`Hello ${props.name}!`)
    }

    return (
        <div>
            <p>My name is {props.name}</p>
            <button onClick={handleClick}>Say
Hello</button></div>
    )
}

export default FunctionComponent
```

3 Règles

1 Un composant aura toujours une majuscule comme première lettre de son nom, qu'il soit fonctionnel ou par Class

2 Il sera toujours retourné 1 élément JSX uniquement, ce dernier sera entourée de parenthère ( ) si son contenu est sur plusieurs lignes (Sur l'exemple div sera retournée)

3 Les éléments dynamiques (JS) seront toujours inséré à l'intérieur d'accolade { }

```
class ClassComponent extends Component

const FunctionComponent

    return (
        <div>
            <p>My name is {props.name}</p>
            <button onClick={handleClick}>Say
Hello</button>
        </div>
    )

                        {props.name}
                          {handleClick}
```

Les PROPS

Une props est une variable passé en argument d'un composant.

Les props fonctionnes comme un objet JS avec une clée et une valeur.

Cette valeur peut être de n'importe quel type : array , object, string, function

Le composant parent transmet la donnée et le composant enfant peu l'exploiter.

Ici ⇒ App envoi la donnée de category "Pizza" et un array à la donnée items

List articles créer ensuite sa vue grâce à ses 2 propriétés reçues.

09

```
import ListArticles from
'./Components/ListArticles'
const fakeDate = [
  {name : 'marguarita', price : 25},
  {name : '4 saisons', price: 32},
]
function App() {
  return (
    <div className="App">
          <ListArticles
                items={fakeDate}
                category="Pizza"/>
    </div>
    )
}

export default App
```

```
function ListArticles (props) {
        const itemsListing = props.items.map((item)=>
(         <h2> {item.name} : {item.price}</h2>
        ))
        return (
            <div>
            <h1>{category}</h1>
                {itemsListing}
            </div>
        )
}

export default ListArticles
```

Fonctionnement des props

Les props ont un fonctionnement assez simple.

Imaginons une enveloppe avec un message à l'intérieur et une adresse dessus.

Vous souhaitez transmettre ce message à votre enfant afin qu'il le lise de vive voix.

Le fonctionnement des props est similaires.

En comparaison vous seriez le composant parent, l'enveloppe la props, l'adresse serait la clée, le message à l'intérieur la valeur et votre enfant, le composant enfant

Quid des fonctions et portée

Comme dit précèdement, une props peut contenir une fonction.

Se pose alors la question de la portée.
Nous verrons plus tard que les données ne sont pas contenu dans des variables principalement avec React mais dans un éléments d'états (le state).
Ce dernier peut être modifié uniquement via une fonction dans le cas de composant fonctionnel.

Les props permettent alors, la transmission de donnée entre le parent et l'enfant

Défi
Les composants

Les Variantes

Créez un composant qui stockera les divers variation du produit.

Celui ci recevra en props :

- Un nom

- Un eventuel ajout de prix

- Une image

Le tout stocké dans un array d'objet.

Par exemple, pour un vetement cela sera des couleurs et des tailles.

Pour une pizza, cela pourra être les ingrédients et les tailles.

```
import ListArticles from
'./Components/ListArticles'
const fakeDate = [
  {name : 'marguarita', price : 25},
  {name : '4 saisons', price: 32},
]
function App() {
  return (
    <div className="App">
          <ListArticles
                items={fakeDate}
                category="Pizza"/>
    </div>
    )
}

export default App
```

```
function ListArticles (props) {
        const itemsListing = props.items.map((item)=>
(         <h2> {item.name} : {item.price}</h2>
        ))
        return (
            <div>
            <h1>{category}</h1>
                {itemsListing}
            </div>
        )
}

export default ListArticles
```

Les Categories

Créez un composant qui affichera les diverses catégorie de produit.

Celui ci recevra en props :

- Un nom

- Une image

Le tout stocké dans un array d'objet.

Par exemple, pour un magasin de vetement cela sera des types de vétements (écharpes , chemises, etc ...).

Pour une pizzerias, cela pourra être les pizzas, les desserts, les boissons.

```
import ListArticles from
'./Components/ListArticles'
const fakeDate = [
  {name : 'marguarita', price : 25},
  {name : '4 saisons', price: 32},
]
function App() {
  return (
    <div className="App">
          <ListArticles
                items={fakeDate}
                category="Pizza"/>
    </div>
    )
}

export default App
```

```
function ListArticles (props) {
        const itemsListing = props.items.map((item)=>
(         <h2> {item.name} : {item.price}</h2>
        ))
        return (
            <div>
            <h1>{category}</h1>
                {itemsListing}
            </div>
        )
}

export default ListArticles
```

Les Articles

Créez un composant qui affichera les divers produit.

Celui ci recevra en props :

- Un nom - String
- Une catégorie - String
- Une liste de variant - Array
- Un prix - Number
- Disponibilité - Boolean
- Une image - Composant

La liste de variant fera appel au composant variants conçu précèdemment.

```
import ListArticles from
'./Components/ListArticles'
const fakeDate = [
  {name : 'marguarita', price : 25},
  {name : '4 saisons', price: 32},
]
function App() {
  return (
    <div className="App">
          <ListArticles
                items={fakeDate}
                category="Pizza"/>
    </div>
    )
}

export default App
```

```
function ListArticles (props) {
        const itemsListing = props.items.map((item)=>
(         <h2> {item.name} : {item.price}</h2>
        ))
        return (
            <div>
            <h1>{category}</h1>
                {itemsListing}
            </div>
        )
}

export default ListArticles
```