



REACT live

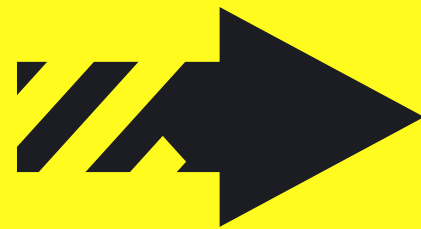
8

Les styles avec React

Il existe plusieurs approches pour ajouter des styles à des composants. Parmi les plus courantes :

- **Les feuilles de style en cascade (CSS)** : Cette méthode est la plus courante pour styliser les pages web. Il s'agit d'inclure des fichiers de styles et les appliquer comme pour une page HTML standard.
- **Les modules CSS** : Cette approche permet d'encapsuler les styles d'un composant dans un module CSS. Les noms de classe générés sont uniques pour chaque composant, ce qui évite les conflits de style entre les différents composants.
- **Les bibliothèques de composant** : Il s'agit de framework adapté pour React, telles que Bootstrap (via **Bootstrap-react** ou **Reactstrap**) ou Material-UI (via **MUI**). Elles offrent des composants React pré-conçus avec des styles prédéfinis. Elles permettent un gain de temps non-négligeable.
- **Les styled components** : Les Styled Components permettent d'écrire des styles directement dans le code JavaScript. Cette approche permet d'encapsuler les styles et de les rendre plus facilement réutilisables.

Quelle que soit la méthode choisie, il est important de garder à l'esprit que les styles ne doivent pas être considérés comme un aspect séparé du développement de l'application. Les styles mettent en avant les fonctionnalités de l'application et leur approche doit être un éléments définis dès le départ.



Les feuilles de style en cascade dans un composant fonctionnel

Les styles en cascade (CSS) sont une méthode courante pour personnaliser l'apparence des composants. La cascade signifie que les styles sont appliqués à un élément dans un ordre spécifique, et que les styles plus spécifiques ont la priorité sur les styles plus généraux.

A la création d'un composant, il est possible d'inclure un fichier CSS pour définir les styles de cet élément. Les sélecteurs CSS sont utilisés pour cibler des éléments spécifiques dans la hiérarchie du DOM, et les propriétés CSS sont utilisées pour définir les styles de ces éléments.

```
import PropTypes from 'prop-types'

const MyFunctionalComponent = () => {
  return (
    <div>
      <p>Name: {props.name}</p>
      <p>Age: {props.age}</p>
      <p>
        Is Married: {props.isMarried}
      </p>
    </div>
  )
}

MyFunctionalComponent.propTypes = {
  name: PropTypes.string,
  age: PropTypes.number,
  isMarried: PropTypes.bool
}
```

Explication

Dans cet exemple, nous avons créé un composant fonctionnel React appelé "MyApp". Nous avons importé le fichier "styleApp.css" contenant les styles CSS. Nous avons également créé une div contenant un titre h1 et un paragraphe p auquel les règles css correspondantes seront appliquées.

Le fichier styleApp.css contient des règles affectant uniquement les balises h1, ces dernières auront un texte rouge d'une taille de 15em avec un fond de couleur noir.

Résultat

Le titre h1 du composant est stylisé avec les propriétés CSS fournies dans le fichier styleApp.css, car ces styles s'appliquent à tous les éléments h1 dans la page. Le paragraphe p n'a pas de style spécifique, donc il n'est pas affecté par les styles définis dans le fichier CSS.

04

MyApp.js

```
import './styleApp.css'
// Import du fichier CSS

function MyApp() {
  return (
    <div>
      <h1>Mon titre</h1>
      <p>Mon texte</p>
    </div>
  )
}

export default MyApp
```

styleApp.css

```
h1 {
  color: red;
  background-color: black;
  font-size: 15em;
}
```

Avantage des styles en cascade

En utilisant la méthode d'importation de fichiers CSS dans le composant, les styles peuvent être facilement réutilisés et organisés dans des fichiers séparés pour faciliter la maintenance du code.

De plus cette méthode n'utilise aucune spécificité propre à React, sa prise en main est donc simplifiée.



Inconvénient des styles en cascade

En utilisant la méthode d'importation de fichiers CSS dans le composant, les styles peuvent facilement devenir plus générique et l'application perd les avantages de React, le style étant "figé" au fichier CSS



Les modules CSS dans un composant fonctionnel

Les modules CSS sont une méthode pour créer des styles de manière modulaire et locale, ce qui évite les conflits de style entre les différents composants d'une application.

Cette méthode est souvent utilisée en combinaison avec le préprocesseur CSS Sass, qui offre des fonctionnalités supplémentaires pour la création de feuilles de style.

Lorsque vous utilisez des modules CSS dans un composant React, vous créez un fichier de style spécifique pour ce composant, et vous utilisez une syntaxe spéciale pour définir les classes de style.

Au lieu d'utiliser des noms de classe génériques, vous utilisez des noms de classe spécifiques au module, qui sont automatiquement générés par le compilateur CSS. Cela garantit que les classes de style sont uniques et ne se chevauchent pas avec d'autres classes de style dans l'application..

```
import PropTypes from 'prop-types'

const MyFunctionalComponent = () => {
  return (
    <div>
      <p>Name: {props.name}</p>
      <p>Age: {props.age}</p>
      <p>
        Is Married: {props.isMarried}
      </p>
    </div>
  )
}

MyFunctionalComponent.propTypes = {
  name: PropTypes.string,
  age: PropTypes.number,
  isMarried: PropTypes.bool
}
```

Explication

Dans cet exemple, nous avons importé le module CSS spécifique. Nous avons également utilisé les noms de classe définis dans le fichier CSS en tant que propriétés className des éléments HTML. Le compilateur CSS de React génère automatiquement des noms de classe uniques pour chaque module CSS, ce qui garantit que les classes de style ne se chevauchent pas avec d'autres classes de style dans l'application.

Le fichier myComponent.module.css contient des règles affectant des classe spécifiques

Résultat

Dans ce composant, nous avons créé une div avec la classe "myComponent", qui contient un titre h1 avec la classe "title" et un paragraphe p. Les styles spécifiés dans le module CSS s'appliquent uniquement à cet élément et ne se chevauchent pas avec d'autres styles définis dans l'application.

ainsi le style .myComponent n'affectera que la div ayant pour valeur className -> styles.myComponent

MyComponent.js

```
import styles from './myComponent.module.css'
// Import du module CSS

function MyComponent() {
  return (
    <div className={styles.myComponent}>
      <h1 className={styles.title}>Mon titre</h1>
      <p>Mon texte</p>
    </div>
  )
}

export default MyComponent
```

myComponent.module.css

```
.myComponent {
  background-color: #fff;
  border: 1px solid #000;
}

.myComponent .title {
  font-size: 2em;
  font-weight: bold;
}
```

Avantage des modules CSS

En utilisant les modules CSS, il est possible de créer des styles de manière modulaire et locale, ce qui évite les conflits de style entre les différents composants d'une application.

La syntaxe spéciale utilisée pour définir les classes de style garantit que les classes sont uniques et ne se chevauchent pas avec d'autres classes de style dans l'application.

Inconvénient des modules CSS

Les modules CSS possèdent 3 gros inconvénients :

-> Ils complexifient la mise en place d'une charte graphique cohérente (les modules étant locaux de base).

-> Les imports peuvent être multiples dans certains cas et cette méthodologie favorise l'oubli.

-> Bien que faisant partie de React, certains navigateurs ont un comportement spécifique avec les modules CSS

Les bibliothèques de composant dans un composant fonctionnel

Il existe de nombreuses bibliothèques de style avec React qui peuvent faciliter la gestion des styles et accélérer le développement.

Les bibliothèques de composant sont des collections de styles pré-définis, de composants de style et de modèles de design qui peuvent être utilisés pour styliser rapidement les applications React.

Certaines des bibliothèques de style les plus populaires incluent Bootstrap, Material UI, Semantic UI, Tailwind CSS , One UI ou même ANT Design.

```
import PropTypes from 'prop-types'

const MyFunctionalComponent = (props) => {
  return (
    <div>
      <p>Name: {props.name}</p>
      <p>Age: {props.age}</p>
      <p>
        Is Married: {props.isMarried}
      </p>
    </div>
  )
}

MyFunctionalComponent.propTypes = {
  name: PropTypes.string,
  age: PropTypes.number,
  isMarried: PropTypes.bool
}
```

Explication

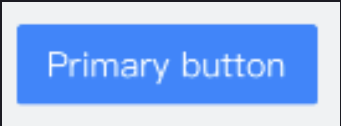
Dans cet exemple, nous importons le composant Button à partir d'Ant Design. Nous créons ensuite un composant fonctionnel MyButton qui prend deux propriétés : text et onClick. Le texte sera le texte affiché sur le bouton, et onClick sera la fonction qui sera appelée lorsque le bouton est cliqué.

L'installation du package node Ant Design (appelé antd) est nécessaire. Ici 3 versions sont possibles install et i sont similaire avec npm et yarn add est via le gestionnaire de paquet yarn

Résultat

Le composant MyButton retourne un bouton Ant Design avec un type de "primary" et une fonction onClick qui est passée en tant que propriété. Le texte affiché sur le bouton est défini par la propriété text.

Cela renvoi un bouton visible dans la documentation de Ant Design avec un style primaire (appelé par le primary)



MyButton.js

```
import { Button } from 'antd'

const MyButton = ({ text, onClick }) => {
  return (
    <Button
      type="primary"
      onClick={onClick}
    >
      {text}
    </Button>
  )
}

export default MyButton
```

En console :

```
npm install antd
npm i antd
yarn add antd
```

Avantage des bibliothèques de composant

En utilisant des bibliothèques de composant, il est possible d'accélérer le développement des applications et améliorer leur apparence. Elles offrent une large gamme de styles pré-définis, de composants de style et de modèles de design pour aider à styliser rapidement les applications. Elles sont également faciles à utiliser et peuvent être personnalisées aux besoins.



Inconvénient des bibliothèques de composant

Les bibliothèques de composant ont cependant plusieurs points faibles

Ils sont souvent victimes d'un poids excessif, leur personnalisation peut être très complexe, les styles peuvent être victimes de surcharge et conflits.

Enfin cela rend l'application dépendante d'une bibliothèque externe



Les styled components dans un composant fonctionnel

Les "Styled components" permettent de créer des composants qui ont leurs styles CSS directement inclus dans le code JavaScript. Cette approche simplifie la création de composants réutilisables, en évitant les problèmes de portée globale associés aux feuilles de style CSS traditionnelles.

Ils sont conçus pour offrir une approche plus simple et plus pratique de la gestion des styles. Au lieu d'utiliser des fichiers CSS séparés, les styles sont définis directement dans le code JavaScript en utilisant la syntaxe CSS. Cela permet de créer des styles dynamiques qui peuvent être modifiés à la volée en fonction des interactions de l'utilisateur ou des changements de l'état de l'application.

```
import PropTypes from 'prop-types'

const MyFunctionalComponent = () => {
  return (
    <div>
      <p>Name: {props.name}</p>
      <p>Age: {props.age}</p>
      <p>
        Is Married: {props.isMarried}
      </p>
    </div>
  )
}

MyFunctionalComponent.propTypes = {
  name: PropTypes.string,
  age: PropTypes.number,
  isMarried: PropTypes.bool
}
```

Styled Components

Explication

Dans cet exemple, nous avons créé un composant fonctionnel Button qui utilise les "Styled components" pour gérer le design. Le composant Button prend trois props : primary, onClick, et children.

Nous avons créé un composant "Styled button" en utilisant la fonction styled de la bibliothèque "styled-components". Nous avons défini plusieurs propriétés CSS, y compris le style du fond, la couleur de police, le rembourrage, la bordure, la bordure de rayon et la taille de police.

Ce dernier servant de container dans le composant fonctionnel

Résultat

Le composant Button se verra recevoir toute les règles CSS notée dans le styled.button.

Certaines de ces valeurs contextuelles dépendra des props (comme color, qui sera blanc si la props primary est vrai et bleu dans le cas contraire)

13

styled-components est installé par défaut avec CRA*, cependant il peut être nécessaire de le réinstaller avec yarn ou npm, pour des raisons de version ou en cas d'installation d'application react, non débuté avec CRA

Button.js

```
import styled from 'styled-components'

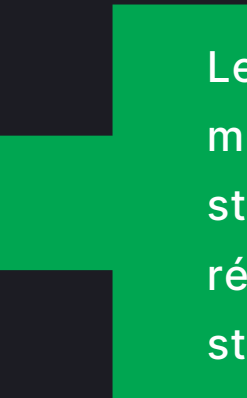
const StyledButton = styled.button`
  background-color: ${props => props.primary ? "blue" : "white"};
  color: ${props => props.primary ? "white" : "blue"};
  padding: 10px 20px;
  border: 2px solid blue;
  border-radius: 5px;
  font-size: 16px;
  cursor: pointer;
`

const Button = ({ primary, onClick, children }) => (
  <StyledButton primary={primary} onClick={onClick}>
    {children}
  </StyledButton>
)

export default Button
```

* CRA = Create React App


Avantage des styled components



Les "Styled components" amènent plusieurs avantages, une meilleure portabilité des composants, une meilleure isolation des styles pour éviter les conflits et les fuites de styles, une meilleure réutilisabilité des styles pour des composants personnalisés et des styles dynamiques qui peuvent être modifiés en temps réel.



Inconvénient des styled components



Les "Styled components" bien que puissant ont tendance à surcharger le code, React à une gestion de composant pour séparer les fonctionnalités, l'intégration du CSS dans le JS vient casser cette logique, cela rend aussi compliqué leur maintenabilité et leur courbe d'apprentissage.

Cela créer aussi une lenteur sur la compilation et rend le travail plus compliqué pour quelqu'un ne maîtrisant pas le JS (designer, etc...)



Découverte d'une bibliothèque de composant : MUI

MUI (Material-UI) est une bibliothèque de composants d'interface utilisateur (UI) basée sur les principes du Material Design de Google. Elle propose un ensemble de composants réutilisables et personnalisables (boutons, champs de saisie, menus déroulants, listes et bien plus encore).

MUI a pour avantage d'être facile à utiliser grâce à une documentation détaillée, des exemples de code et des guides pour l'installation et la configuration.

MUI offre également un certain nombre de fonctionnalités pratiques, telles que la prise en charge du responsive, l'accessibilité et une grande variété d'icônes.

En résumé, MUI est une bibliothèque solide pour la construction d'interfaces utilisateur en React, qui permet de gagner du temps et de simplifier la tâche de développement.

```
import { makeStyles } from '@material-ui/core/styles'
import { Grid, Paper, Typography } from '@material-ui/core'

const useStyles = makeStyles((theme) => ({
  root: {
    flexGrow: 1,
    padding: theme.spacing(2),
  },
  paper: {
    padding: theme.spacing(2),
    textAlign: 'center',
    color: theme.palette.text.secondary,
  },
}))

function Dashboard() {
  const classes = useStyles()

  return (
    <div className={classes.root}>
      <Grid container spacing={3}>
        <Grid item xs={12}>
          <Typography variant="h4" align="center" gutterBottom>
            Dashboard
          </Typography>
        </Grid>
        <Grid item xs={12} sm={6} md={4}>
          <Paper className={classes.paper}>
            <Typography variant="h5" gutterBottom>
              Total Sales
            </Typography>
            <Typography variant="h6">
              $50,000
            </Typography>
          </Paper>
        </Grid>
        <Grid item xs={12} sm={6} md={4}>
          <Paper className={classes.paper}>
            <Typography variant="h5" gutterBottom>
              New Customers
            </Typography>
            <Typography variant="h6">
              50
            </Typography>
          </Paper>
        </Grid>
        <Grid item xs={12} sm={6} md={4}>
          <Paper className={classes.paper}>
            <Typography variant="h5" gutterBottom>
              Orders Processed
            </Typography>
            <Typography variant="h6">
              500
            </Typography>
          </Paper>
        </Grid>
      </Grid>
    </div>
  );
}

export default Dashboard;
```


Découverte d'une bibliothèque de composant : Reactstrap

Reactstrap est une bibliothèque de composants qui permet de construire des interfaces utilisateur basées sur Bootstrap 4. Elle fournit des composants prêts à l'emploi (boutons, formulaires, cartes et alertes).

Reactstrap utilise les styles de Bootstrap pour rendre les composants visuellement cohérents et permet aux développeurs de personnaliser les styles en utilisant des classes CSS de Bootstrap ou personnel.

En plus de fournir des composants prêts à l'emploi, Reactstrap utilise également les fonctionnalités de Bootstrap, telles que la grille responsive, pour faciliter la création d'interfaces utilisateur flexibles et adaptatives.

En résumé, utiliser Reactstrap, permet de créer des interfaces utilisateur professionnelles et réactives avec des fonctionnalités de base en peu de temps et avec un minimum de code personnalisé.

```
import {
  Container, Row, Col,
  Card, CardBody, CardTitle, CardText
} from 'reactstrap'

function Dashboard() {
  return (
    <Container>
      <Row>
        <Col>
          <h1 className="text-center my-3">Dashboard</h1>
        </Col>
      </Row>
      <Row>
        <Col xs={12} sm={6} md={4}>
          <Card>
            <CardBody>
              <CardTitle>Total Sales</CardTitle>
              <CardText>$50,000</CardText>
            </CardBody>
          </Card>
        </Col>
        <Col xs={12} sm={6} md={4}>
          <Card>
            <CardBody>
              <CardTitle>New Customers</CardTitle>
              <CardText>50</CardText>
            </CardBody>
          </Card>
        </Col>
        <Col xs={12} sm={6} md={4}>
          <Card>
            <CardBody>
              <CardTitle>Orders Processed</CardTitle>
              <CardText>500</CardText>
            </CardBody>
          </Card>
        </Col>
      </Row>
    </Container>
  )
}

export default Dashboard
```


Découverte d'une bibliothèque de composant : ANT Design

ANT Design est une bibliothèque de composants d'interface utilisateur qui fournit une grande variété de composants prêts à l'emploi pour créer des interfaces utilisateur modernes et esthétiques. Elle est basée sur le framework de conception Ant Design, qui est très populaire dans les pays de l'est avec de nombreux composants pour la mise en page (formulaires, boutons, icônes, menus, tables, cartes, ect).

Les composants sont facilement personnalisables, avec de nombreux paramètres de configuration pour ajuster leur apparence et leur comportement.

ANT Design dispose également d'un système de grille pour la mise en page et d' options de personnalisation pour les couleurs, les typographies et les icônes. Il est également possible d'utiliser le thème par défaut ou un personnalisé.

En résumé, ANT Design est une bibliothèque de composants d'interface utilisateur riche en fonctionnalités, facile à utiliser et à personnaliser.

```
import { Row, Col, Card } from 'antd'

function Dashboard() {
  return (
    <div>
      <h1 className="text-center my-3">Dashboard</h1>
      <Row gutter=[[16, 16]]>
        <Col xs={24} sm={12} md={8}>
          <Card>
            <Card.Meta
              title="Total Sales"
              description="$50,000" />
          </Card>
        </Col>
        <Col xs={24} sm={12} md={8}>
          <Card>
            <Card.Meta
              title="New Customers"
              description="50" />
          </Card>
        </Col>
        <Col xs={24} sm={12} md={8}>
          <Card>
            <Card.Meta
              title="Orders Processed"
              description="500" />
          </Card>
        </Col>
      </Row>
    </div>
  )
}

export default Dashboard
```



Défi
Les styles

Le styled components

Dans un composant ListArticles.js qui contiendra plusieurs articles, utilisez les "styled components" afin de créer une grille de présentation des produits.

Ces derniers seront styliser dans le prochain défi.

```
import ListArticles from
'./Components/ListArticles'
const fakeDate = [
  {name : 'margarita', 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
```

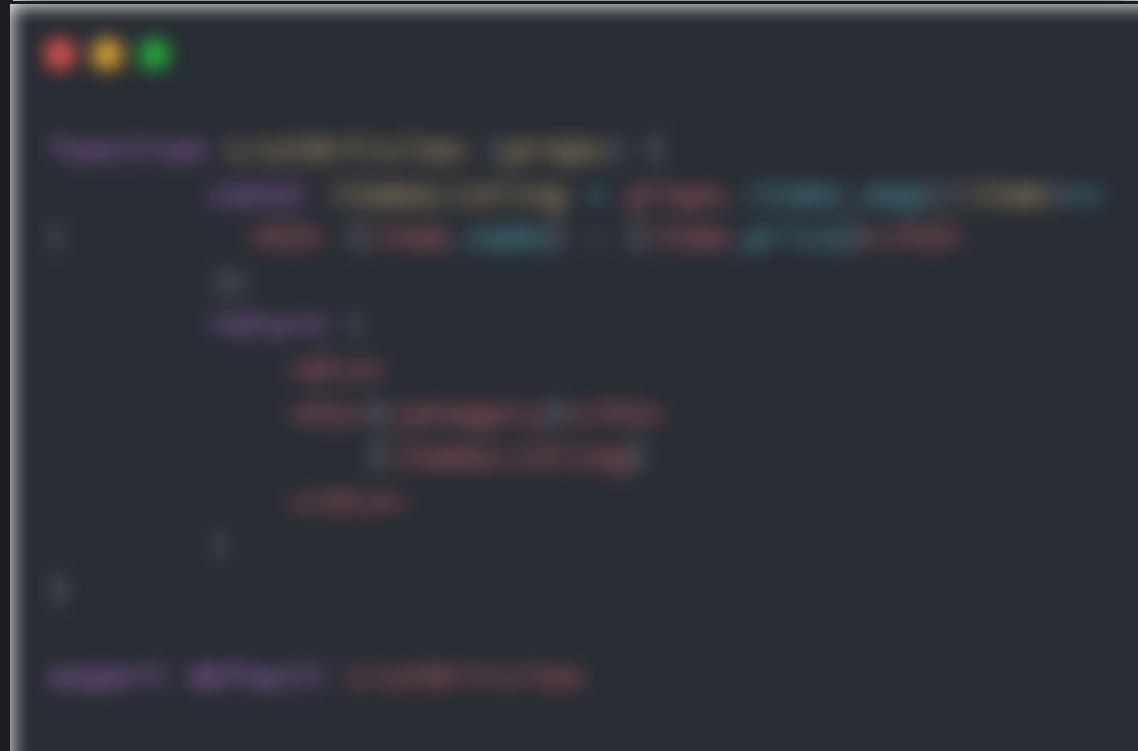
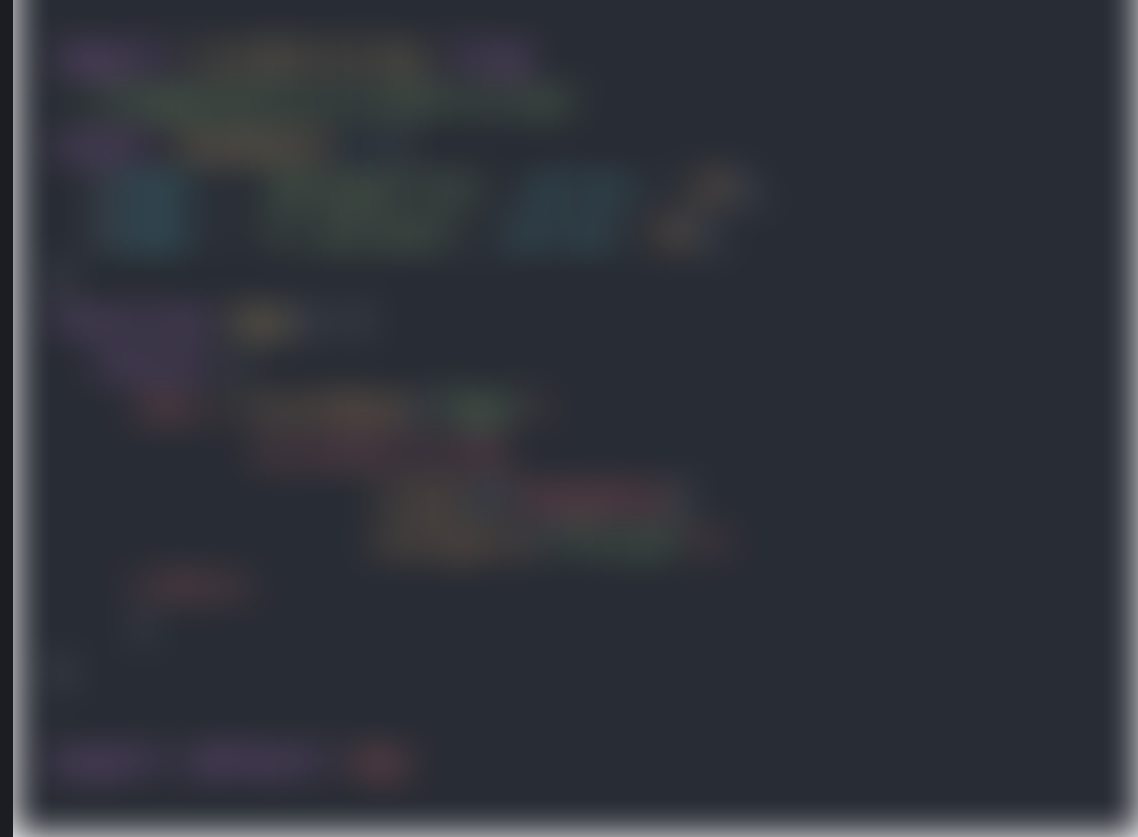
Material UI

Créer une card article avec le nom,
une photo et le prix de chaque articles
(que vous recevrez en props)

Cela grâce à la bibliothèque de
composant Material UI aussi nommé
MUI.

La documentation est disponible à
l'adresse suivante :

<https://mui.com/material-ui/getting-started/installation/>



ANT Design

Comme le défi précédent, créer une card article avec le nom, une photo et le prix de chaque articles (que vous recevrez en props)

Cette fois grâce à la bibliothèque de composant ANT Design (antd).

La documentation est disponible à l'adresse suivante :

<https://ant.design/docs/react/getting-started>