wawiwa
React
2022 © Wawiwa Tech | Confidential


wawiwa
18 - Dando estilo en React
visual design
2022 © Wawiwa Tech | Confidential

wawiwa

Introducción

El estilo es importante para nuestras aplicaciones para que se vean bien. Hay muchas maneras de estilar en React. En este capítulo vamos a ver más en detalle cada método.

Los métodos:

1. Inline Styling (estilo en línea)
2. Inline Styling using an object variable (estilo en línea usando una variables de objeto)
3. Inline Styling using a style object file (estilo en línea usando un archivo de objeto)
4. stylesheet de CSS
5. Styled-components

wawiwa

Inline Styling (Estilo en línea)

Podemos agregar estilos en línea a todo componente React que queramos renderizar. Estos estilos son escritos como atributos y son pasados al elemento.

Un ejemplo:


```
<h3 style={{color: "blue", background: "yellow"}}>This is Inline style!</h3>
```

Notas:

- El atributo se llama “style”
- Pasamos el estilo con **dos llaves** (la primera llave inyecta JavaScript en JSX. La llave interior crea un objeto)
- Las propiedades **se separan con una coma** (esto es porque pasamos un objeto)
- Las propiedades están escritas en **camelCases**

wawiwa

Inline Styling (Estilo en línea) - ¿El mejor método?

¿El estilo en línea es el mejor método para React?

Los estilos en línea es la **manera más directa de darle estilo** a cualquier aplicación de React. Darle estilo en línea a los elementos no requiere que crees un style sheet por separado. El estilo se aplica directamente a los elementos a diferencia de los estilos provenientes de un style sheet que tienen una **precedencia más alta**.

```
function App() {
  return (
    <>
      <h3 style={{color: "blue", background: "yellow"}}>Inline Style</h3>
    </>
  )
}
```

wawiwa

Ejercicio 19

Copia el siguiente código:

```
const products = [
  {
    name: 'Product 1',
    description: 'Description of product 1',
    price: '$10.99'
  },
  {
    name: 'Product 2',
    description: 'Description of product 2',
    price: '$24.99'
  },
  {
    name: 'Product 3',
    description: 'Description of product 3',
    price: '$15.49'
  },
];
```

```
return (
    <div>
      {products.map((product, index) => (
        <div key={index}>
          <div>{product.name}</div>
          <div>{product.description}</div>
          <div>{product.price}</div>
          </div>
      ))}
    </div>
  );
```

wawiwa

Ejercicio 19A

1. Da estilo al “product name” para que sea negrita con un tamaño de 20px.
2. Da estilo al “price” para que sea de color rojo.

wawiwa

Inline Styling (estilo en línea) - Mejora la legibilidad

```
<h3 style={{color: "blue", background: "yellow", border:"2px solid green", textAlign:"center", padding:"1px 2px 3px 4px"}}>This is</h3>
<h3 style={{color: "red", textAlign:"start", width:100, border:"20px solid red"}}>Inline Style</h3>
```

En el código arriba, solamente agregamos propiedades a los elementos a los que les dimos estilo. Se siente sobrecargado y es difícil de leer. Sin embargo, imagina que hemos agregado más estilos al elemento. Ahí es donde el método “inline” se rompe porque no se verá limpio.

La solución a este problema, es crear un variables de objeto y pasarlas al elemento.

wawiwa

Crear un variables de objeto usando estilo en línea

Creamos el variables de objeto de la misma manera en la que creamos un objeto de Javascript. Después, pasamos el objeto en donde escribimos en línea directamente. Por ejemplo:

The esto:

```
<p style={{fontSize: 20,color: "purple",textAlign: "left"}}>I love cats</p>
```

A esto:

```
const Label = {
  fontSize: 20,
  color: "purple",
  textAlign: "left"
}
<p style={Label}>I love cats</p>
```

wawiwa

Crear un Objeto de estilo usando estilo en línea

Podemos guardar las propiedades de dos maneras:

En variables separadas:

```
function App() {
  const Container = {
    color: "blue",
    background: "yellow",
    border: "2px solid green",
    textAlign: "center",
    padding: "1px 2px 3px 4px",
    width: 300
  };
  const Title = {
    color: "red",
    textAlign: "start"
  };

  const Label = {
    fontSize: 20,
    color: "purple",
    textAlign: "left"
  }

  return (
      <div style={Container}>
        <h3 style={Title}>Inline Style</h3>
        <p style={Label}>I love cats</p>
      </div>
  )
}
export default App;
```

En una variable:

```
function App() {
  const styles = {
    Container: {
      color: "blue",
      background: "yellow",
      border: "2px solid green",
      textAlign: "center",
      padding: "1px 2px 3px 4px",
      width: 300
    },
    Title: {
      color: "red",
      textAlign: "start"
    },
    Label: {
      fontSize: 20,
      color: "purple",
      textAlign: "left"
    }
  }

  return (
      <div style={styles.Container}>
        <h3 style={styles.Title}>Inline Style</h3>
        <p style={styles.Label}>I love cats</p>
      </div>
  )
}
export default App;
```

wawiwa

Crear un Objeto de estilo usando estilo en línea

El siguiente ejemplo que usa este método se ve claro, pero **sigue sobrecargado**.

Si le damos estilo a todo un componente con más de 10 objetos de variables,se puede volver todavía más confuso.

El siguiente método puede ayudarnos con esto. La solución involucra guardar las variables en un archivo .js separado.

```
function App() {
  const Container = {
    color: "blue",
    background: "yellow",
    border: "2px solid green",
    textAlign: "center",
    padding: "1px 2px 3px 4px",
    width: 300
  };
  const Title = {
    color: "red",
    textAlign: "start"
  };

  const Label = {
    fontSize: 20,
    color: "purple",
    textAlign: "left"
  }

  return (
      <div style={Container}>
        <h3 style={Title}>Inline Style</h3>
        <p style={Label}>I love cats</p>
      </div>
  )
}
export default App;
```

wawiwa

Ejercicio 19B

Crea una variable de objeto de estilo “ProductContainer” que le de estilo a cada “product” con:

```
display: 'flex',
justifyContent: 'space-between',
alignItems: 'center',
padding: '10px'
```

wawiwa

Crear un Archivo de Objeto de Estilo (Usando estilo en línea)

El componente y su objeto de estilo no tienen que estar juntos en el mismo archivo. Por esto, podemos crear un **archivo .js separado** para nuestros estilos.

Después vamos a **exportar** esos estilos e **importarlos** al componente.
De esta manera, los estilos pueden ser **reutilizados en componentes múltiples** y no solamente en el actual.

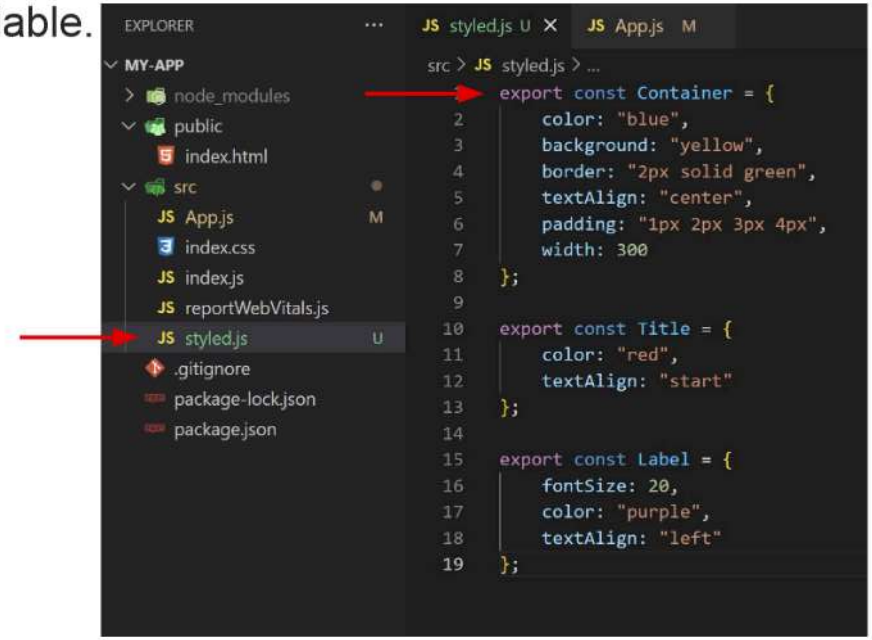
wawiwa

Crear un Archivo de Objeto de Estilo (Usando estilo en línea)

paso 1 - Crea un archivo separado .js

paso 2 - Agrega los estilos

paso 3 - Agrega un “export” antes de cada variable.



wawiwa

Crear un Archivo de Objeto de Estilo (Usando estilo en línea)

paso 4 - importa los estilos en el componente

paso 5 - úsalos en el componente

```
import { Container, Title, Label } from "./styled";
function App() {

  return (
      <div style={Container}>
        <h3 style={Title}>Inline Style</h3>
        <p style={Label}>I love cats</p>
      </div>
  )
}
export default App;
```

wawiwa

Pros y Contras de usar estilo en línea

Pros:

Usar estilos en línea puede ayudarte rápidamente a hacer un prototipo de tu interfaz.

Contras:

La declaración de estilos puede rápidamente convertirse en desordenada y desorganizada con estilos en línea. Adicionalmente, los estilos en línea no soportan el principio DRY (Do not Repeat Yourself - “no te repitas a tí mismo”). Usar estilos en línea no es adecuado para proyectos grandes con mucho código.

wawiwa

CSS Stylesheet

Otro método común para dar estilo en React es usar un archivo de stylesheet de CSS (archivo con extensión .css).

1. Crea en la misma ubicación un archivo CSS y agrega estilos.
2. Importa el archivo.
3. Usa la propiedad

styled.css U

src > styled.css > body

```css
.container {
    color: blue;
    background-color: yellow;
    border: 2px solid gray;
    width: 300px;
}

body {
    background-color: red;
}
```

```jsx
import "./styled.css";

function App() {

  return (
      <div className="container">
```

wawiwa

CSS Stylesheet

Problema:

Un pequeño problema puede ser tu convención de **denominación.** Una vez bien desarrollada la aplicación, es más difícil proponer **nombres de clase únicos** para tus elementos, especialmente si tienes 5 divs envueltas dentro de ellas mismas.

Si no tienes una convención de denominación que te es familiar, fácilmente puedes cometer errores creando varias clases con el mismo nombre, **causando conflictos**.

Solución:

Una solución simple, dale a tu archivo CSS un nombre único. Después comienza cada nombre de la clase con el nombre del archivo.

wawiwa

Styled-components

Introducción:

Los styled-components utilizan los literales de plantilla etiquetados de ES6 para diseñar componentes. Esto elimina la asociación entre componentes y estilos.

Esto quiere decir que cuando tú defines tus estilos, creas un componente de React normal que tiene tus estilos adjuntados.

Con styled-components podemos usar componentes con estilos reusables.

wawiwa

Styled-components

Los beneficios de usar styled-components:

- **Optimiza la experiencia** - Para developers y para usuarios.
- **CSS crítico automático -** Styled-components siguen la pista de cuáles componentes son renderizados una una página e inyecta sus estilos y nada más, completamente en automático. Los usuarios **descargan la menor** cantidad de código necesario.
- **No hay bugs por nombres de clases** - Styled-components generan nombres de clases únicos para tus estilos. Nunca te tienes que preocupar por que se dupliquen, superpongan o errores al escribirlos.
- **Eliminación de los no usados -** Si el componente no es usado, se elimina y todos sus estilos se eliminan con el.

wawiwa

Styled-components

Installation:

Installing styled components takes a single command:

- With npm:

  npm install styled-components

- With yarn:

  yarn add styled-components

wawiwa

Styled-components

La sintaxis:

```
const ComponentName = styled.element`
css_property: css_value;
`;
```

Explicación:

1. ComponentName - puede ser cualquier nombre
2. element - puede ser cualquier elemento soportado con JSX
3. css_property - representa en nombre de una propiedad en CSS
4. css_value - representa el valor de una propiedad

wawiwa

Styled-components

Primero, debemos importar styled-components y React :

```
import React from "react";
import styled from "styled-components";
```

Después, crea un styled component:

```
const Title = styled.div`
  color: purple;
  font-size: 30px
`;
```

Alternativamente, puedes usar objetos de estilo. Esto nos permite la fácil portabilidad de CSS de los estilos en línea, mientras soporta las capacidades más avanzadas de styled-components, como selectores y “media queries”.

```
const Title = styled.div({
  color: "purple",
  fontSize: "30px"
});
```

wawiwa

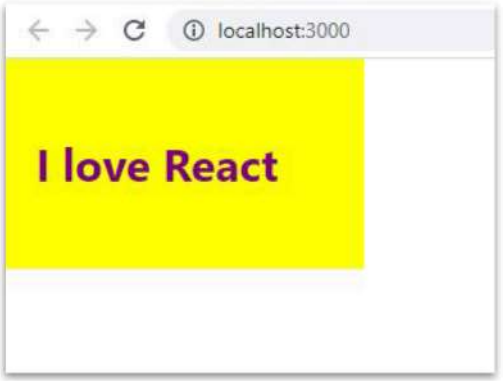
Styled-components - Ejemplo

```
import React from "react";
import styled from "styled-components";
function App() {

  const Container = styled.div`
    background: yellow;
    width: 200px;
    padding: 20px;
  `;
  const Title = styled.p({
    color: "purple",
    fontSize: "30px",
    fontWeight: "bold"
  });

  return (
      <Container>
        <Title>I love React</Title>
      </Container>
  )
}
export default App;
```

importa React y styled-components

Nombre del tag

Nombre del componente



wawiwa

Styled-components - Dinámicamente

Styled-components con **funcionales**, por lo que podemos **cambiar el estilo del componente dinámicamente, dependiendo de sus props**.

Por ejemplo, si tenemos un título <p> donde solamente el tamaño de la letra cambia, podemos configurar esta propiedad por medio de props. Lo mismo con el color del borde del “Container”.


```jsx
import React from "react";
import styled from "styled-components";
function App() {

  const Container = styled.div`
    background: yellow;
    border: 5px solid ${props=>props.borderColor};
    padding: 20px;
    width: 300px
  `;
  const Title = styled.p((props)=>({
    color: "purple",
    fontSize: props.font,
    fontWeight: "bold"
  }));

  return (
      <Container borderColor="purple">
        <Title font="40px">I love React</Title>
        <Title font="20px">Semi title</Title>
      </Container>
  )
}
export default App;
```

wawiwa

Styled-components - Dinámicamente

El código:

```
import React from "react";
import styled from "styled-components";
function App() {

  const Container = styled.div`
    background: yellow;
    border: 5px solid ${props=>props.borderColor};
    padding: 20px;
    width: 300px
  `;
  const Title = styled.p((props)=>({
    color: "purple",
    fontSize: props.font,
    fontWeight: "bold"
  }));

  return (
      <Container borderColor="purple">
        <Title font="40px">I love React</Title>
        <Title font="20px">Semi title</Title>
      </Container>
  )
}
export default App;
```

wawiwa

Styled-components - Funcional

Otro ejemplo: Queremos configurar las propiedades del botón dependiendo del “mode” (dark - “oscuro”/light - “claro”).

dark=”true”: Click

dark:”false”: Click

```
import React from "react";
import styled from "styled-components";
function App() {

  const Button = styled.button((props)=>({
    background: props.dark =="true"? "black":"white",
    color:props.dark =="true"? "white":"black"
  }));

  return (
    <Button dark="true" >Click</Button>
  )
}
export default App;
```

wawiwa

Styled-components - Reutilizar y heredar Styles 1

Podemos reutilizar componentes y heredar su estilo. Por ejemplo:

```
const Title1 = styled.p({
  color: "purple",
  fontWeight: "bold",
  background:"yellow",
});
const Title2 = styled.p({
  color: "purple",
  fontWeight: "bold",
  background: "yellow",
  fontSize:"10px"
});
```

```
const Title1 = styled.p({
  color: "purple",
  fontWeight: "bold",
  background:"yellow",
});
const Title2 = styled(Title1)`
  font-size: 10px
`
```

En Title2, vamos a obtener todos los estilos de Title1 y font-size va a ser agregado.

wawiwa

Styled-components - Reutilizar y heredar Styles 2

Adicionalmente, poder anular propiedades. Por ejemplo:

```
const Title1 = styled.p({
  color: "purple",
  fontWeight: "bold",
  background:"yellow",
});

const Title2 = styled(Title1)`
  color: red
`

return (
  <>
  <Title1>1</Title1>
  <Title2>2</Title2>
  </>
)
```

wawiwa

Styled-components - Aplicar pseudo-clases y elementos

También, podemos coincidir interacciones del usuario con nuestro estilo, aplicando pseudo-clases y pseudo-elementos a nuestro styled-component :

```
const Card = styled.div`
border: 1px solid black;
&:hover {
  border: 1px solid red;
}
`
return (
  <>
  <Card>I love styled-components</Card>
  </>
)
```

wawiwa

Styled-components - Dándole estilo al componente hijo

Mientras aplicamos styled-components a un componente, podemos aplicar estilos a elementos hijo, sin crear un styled-component por cada elemento. Por ejemplo:

```
const Card = styled.div`
border: 1px solid black;
width: 300px;
h4 {
  color: red;
  text-align: center;
}
`
return (
  <>
    <Card>
      <h4>I love styled-components</h4>
  </Card>
  </>
)
```

wawiwa

Ejercicio 19C

1. Convierte “ProductContainer” en un styled-component.
2. Crea un styled-component para el div que contiene {product.name}, llámalo “ProductName”.
3.
   a. Crea un styled-component para el div que contiene {product.price}, llámalo “ProductPrice”.
   b. Cuando el usuario pase el mouse sobre el componente ProductPrice, el color de la letra debe ser verde.
4. Mueve los styled-components a un archivo separado. Exporta los componentes ahí e importalos.