



REACT.js

CLASE 3



The image features abstract geometric shapes in the corners. In the top-left, there are overlapping shapes in light blue, green, and orange. In the top-right, there are shapes in light blue, green, and purple. In the bottom-left, there are shapes in red, orange, and purple. In the bottom-right, there are shapes in blue, purple, and orange. The central text is in a bold, dark blue font.

Componentes 2.0

Atributos

class -> className

```
<div className="username"></div>
```

for -> htmlFor

```
<label htmlFor="username">Usuario</label>
```

Atributos

¡Podemos usar
objetos como
atributos!

```
class MiComponente extends Componentes {  
  const divStyle = {  
    color: 'white',  
    backgroundColor: 'black',  
  }  
  
  render(){  
    return <div style={divStyle}> Hello World! </div>  
  }  
}
```

Atributos

```
class MiComponente extends Componentes {  
  const divStyle = {  
    color: 'white',  
    backgroundColor:'black',  
  }  
  
  render(){  
    return <div style={divStyle}> Hello World! </div>  
  }  
}
```

Algunas propiedades de CSS como background-color, necesitan escribirse con la técnica camelCase para que puedan renderizarse de la manera correcta

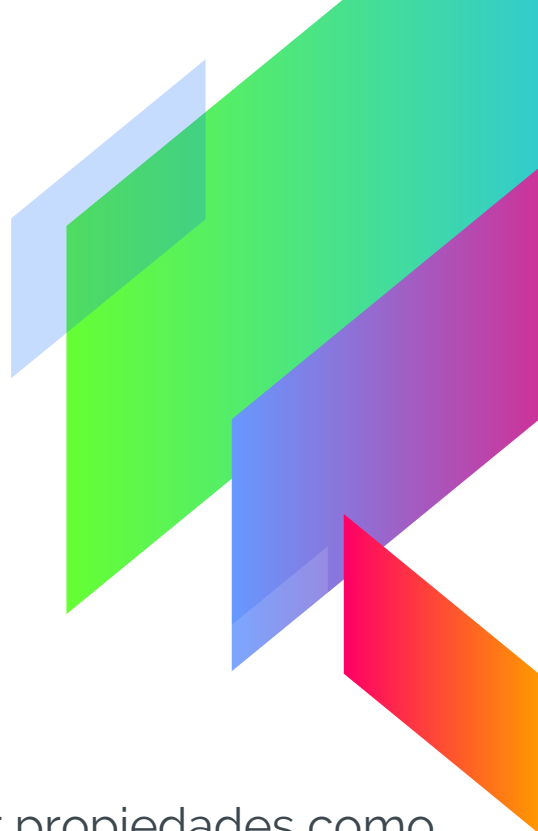
Atributos

<https://reactjs.org/docs/dom-elements.html>

accept acceptCharset accessKey **action** allowFullScreen allowTransparency alt
async autoComplete autoFocus autoPlay capture cellPadding cellSpacing challenge
charSet checked cite classID **className** colSpan cols content contentEditable
contextMenu controls controlsList coords crossOrigin data dateTime default defer
dir disabled download draggable encType form formAction formEncType formMethod
formNoValidate formTarget frameBorder headers **height** hidden high **href** hrefLang
htmlFor httpEquiv icon **id** inputMode integrity is keyParams keyType kind label
lang list loop low manifest marginHeight marginWidth max maxLength media
mediaGroup method min minLength multiple muted name noValidate nonce open
optimum pattern placeholder poster preload profile radioGroup readOnly rel
required reversed role rowSpan rows sandbox scope scoped scrolling seamless
selected shape size sizes span spellCheck **src** srcDoc srcLang srcSet start step
style summary tabIndex target title **type** useMap value **width** wmode wrap

A cluster of overlapping, semi-transparent geometric shapes in the top-left corner, including a green triangle, a light blue parallelogram, a brown trapezoid, an orange parallelogram, a red parallelogram, and a purple parallelogram.

Props

A cluster of overlapping, semi-transparent geometric shapes in the top-right corner, including a light blue parallelogram, a green parallelogram, a teal parallelogram, a purple parallelogram, and a red parallelogram.

Un componente en React puede recibir propiedades como parámetros para poder insertar valores y eventos en su HTML.

Props (atributos)

```
<MiComponente titulo="Clase 3"/>
```


Props (atributos)

`<MiComponente titulo="Clase 3"/>`

```
class MiComponente extends Component{
  render(){
    return(
      <div>
        <h1> {this.props.titulo} </h1>
      </div>
    );
  }
}

export default MiComponente;
```

Props (atributos)

```
<MiComponente  
  titulo="Clase 3"  
  texto="Elementos de un componente"/>
```

```
class MiComponente extends Component{  
  render(){  
    return(  
      <div>  
        <h1> {this.props.titulo} </h1>  
        <p> {this.props.texto} </p>  
      </div>  
    );  
  }  
}  
  
export default MiComponente;
```

Key Props

```
const usuarios = ["Dario","Javier","Alejandro"];  
<MyList items={usuarios} />
```

Key Props

```
const usuarios = ["Dario","Javier","Alejandro"];
```

```
<MyList items={usuarios} />
```

```
class MyList extends Component {  
  render(){  
    const items = this.props.items;  
    const listItems = items.map((item) =>  
      <li>  
        {item}  
      </li>  
    );  
    return (  
      <ul>{listItems}</ul>  
    );  
  }  
}
```

```
export default MyList;
```

Key Props

⊗ ► Warning: Each child in an array or iterator should have a unique "key" prop.

```
class MyList extends Component {
  render(){
    const items = this.props.items;
    const listItems = items.map((item) =>
      <li>
        {item}
      </li>
    );
    return (
      <ul>{listItems}</ul>
    );
  }
}

export default MyList;
```

Key Props

```
const usuarios = ["Dario","Javier","Alejandro"];
```

```
<MyList items={usuarios} />
```

```
class MyList extends Component {  
  render(){  
    const items = props.items;  
    const listItems = items.map((item) =>  
      <li key={item}>  
        {item}  
      </li>  
    );  
    return (  
      <ul>{listItems}</ul>  
    );  
  }  
}  
  
export default MyList;
```

Key Props

Las key ayudan a React a identificar qué elementos han cambiado, agregado o eliminado.

Es decir, React por medio de las keys determina si es el mismo elemento o no.

- **Solo es necesario** agregar keys cuando devolvemos un array de elementos iguales.
- La key solo debe ser única entre elementos hermanos.
- Las keys no se muestran en el HTML final (si quisiéramos esto también deberíamos utilizar id)

The slide features decorative geometric shapes in the top corners. On the left, there are overlapping triangles in shades of green, blue, orange, and purple. On the right, there are overlapping triangles in shades of green, blue, purple, and red. The main content is centered on a white background.

PropTypes

Con React creamos componentes reusables. Por lo tanto, es de buena práctica definir las props que acepta cada componente.

Con PropTypes validamos las propiedades.

PropTypes

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

PropTypes

```
class MiComponente extends Component{  
  render(){  
    return(  
      <h1> {this.props.titulo} </h1>  
    );  
  }  
}  
MiComponente.propTypes = {  
  titulo: PropTypes.string.isRequired  
};
```

PropTypes

- ◆ `PropTypes.string`
- ◆ `PropTypes.number`
- ◆ `PropTypes.array`
- ◆ `PropTypes.func`
- ◆ Etc.

<https://www.npmjs.com/package/prop-types>

Default Props

```
class MiComponente extends Component{  
  render(){  
    return <h1> {this.props.titulo} </h1>  
  }  
}
```

```
MiComponente.propTypes = {  
  titulo: PropTypes.string.isRequired,  
};
```

```
MiComponente.defaultProps = {  
  titulo: "Sin titulo"  
}
```

Template Strings

//ES5

```
console.log('Hola ' + nombre + ' ' + apellido + '!');
```

ES6

```
console.log(`Hola ${nombre} ${apellido}!`);
```



ESLint



ESLint

Un Linter es una pieza de Software que nos ayuda a detectar y corregir errores en el código y a su vez unificar estilos de código pudiendo ver los errores tanto en el navegador como en nuestro IDE.

Uno de los Linters de JS se llama ESLint dándonos la habilidad de personalizar las configuraciones.



ESLint

create-react-app trae su propia configuración de ESLint que es la que vemos en la consola del navegador. Para generar nuestra propia configuración debemos verla desde el IDE. En Atom la extensión se llama **linter-eslint** En Sublime se llama **SublimeLinter-eslint**



ESLint

<https://eslint.org/docs/rules/>

Para crear nuestra propia configuración debemos crear un archivo `.eslintrc` con el siguiente formato:

```
{
  "extends": "react-app",
  "rules":{
    //Fill with rules and have fun!
  }
}
```



Ejercicios

Tiempo de práctica



Gracias!

¿Preguntas?

