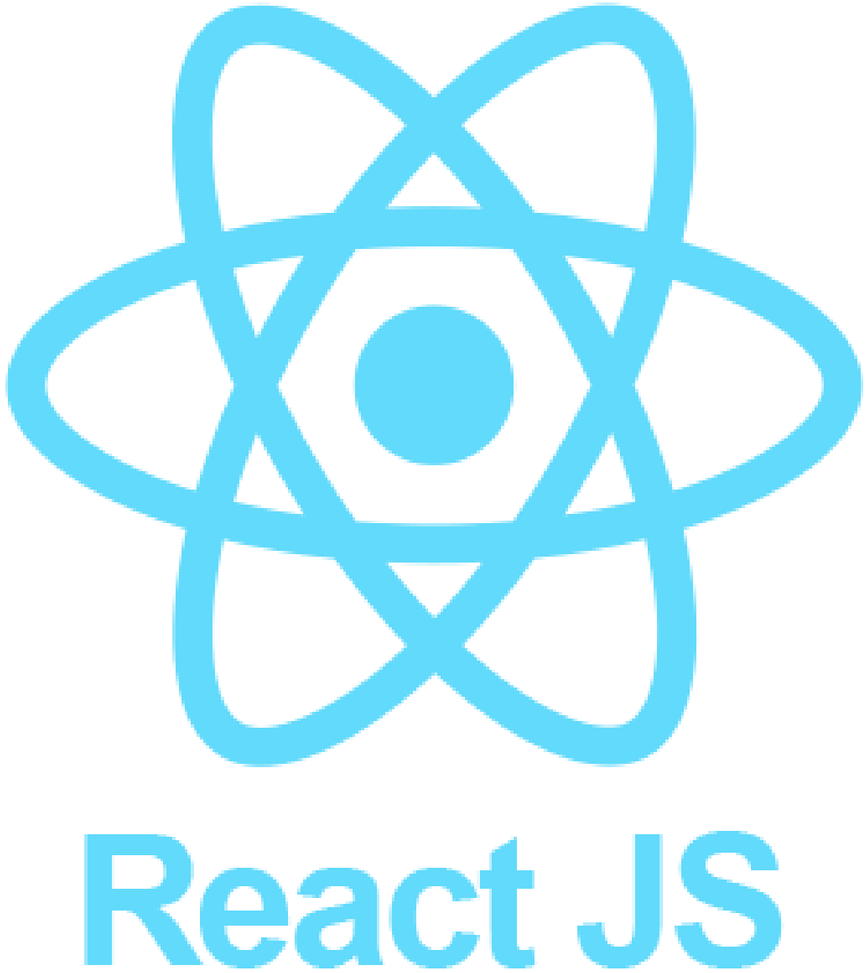
Estructura de Datos | |

Jonathan López Londoño
jlopezl@uao.edu.co
315 926 5443



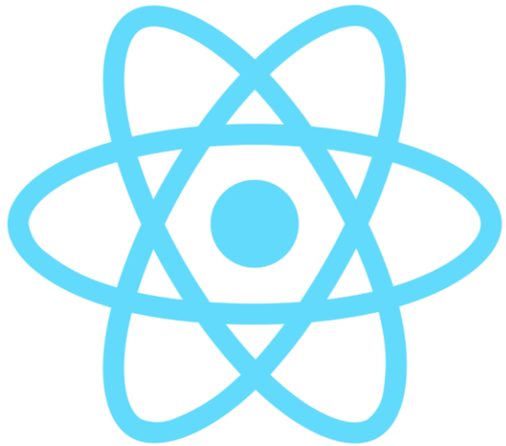
React JS

A JavaScript library for building user interfaces. Makes it painless to create interactive UIs.

Declarative: Handles states in your application. React will efficiently update and render just the right components when your data changes.

Component-Based: Build encapsulated components that manage their own state, then compose them to make complex UIs.

Learn Once, Write Anywhere: It doesn't matter about the rest of your technology stack, so you can develop new features in React without rewriting existing code.



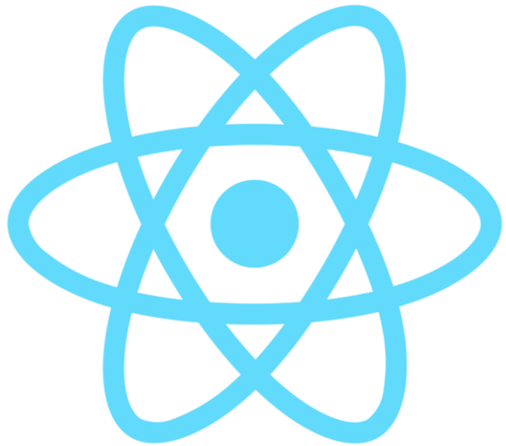
Create React Project

Injecting into any Project:

```
<script
    crossorigin
    src="https://unpkg.com/react@16/umd/react.production.min.js">
</script>

<script
    crossorigin
    src="https://unpkg.com/react-dom@16/umd/react-dom.production.min.js">
</script>

<script
    src="https://unpkg.com/babel-standalone@6/babel.min.js">
</script>
```



Requirements to Start

https://nodejs.org/en/download

https://www.w3schools.com/html/

https://www.w3schools.com/css/

https://www.w3schools.com/js/

https://code.visualstudio.com/download

Packages Managers

NPM

npm is the standard package manager for Node.js.

VITE

Vite is a build tool (build tool) for frontend projects that improves performance and development.

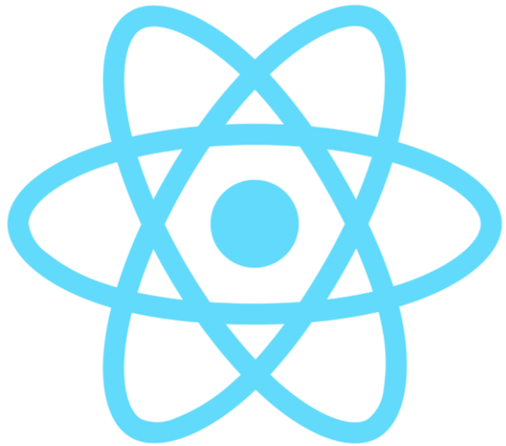
Create React Project

Injecting into any Project:

```
<script
    crossorigin
    src="https://unpkg.com/react@16/umd/react.production.min.js">
</script>

<script
    crossorigin
    src="https://unpkg.com/react-dom@16/umd/react-dom.production.min.js">
</script>

<script
    src="https://unpkg.com/babel-standalone@6/babel.min.js">
</script>
```



Create React Project

Create a new folder called **Estructuras2** and open the path in a console

NPM

In Estructuras2 folder write this command in console:

npx create-react-app my-npm-app

cd my-npm-app

npm start

VITE

In **Estructuras2** folder write this command in console:

npm create vite@latest

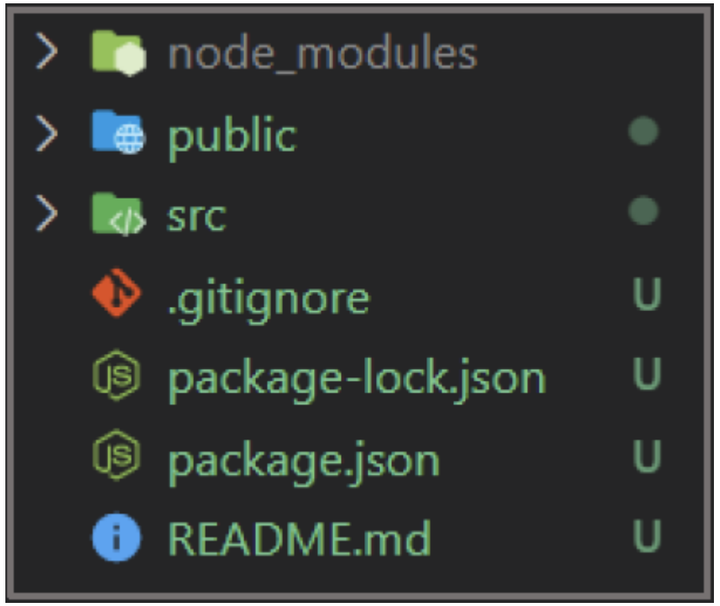
… Follow the menu (use my-vite-app as a name)

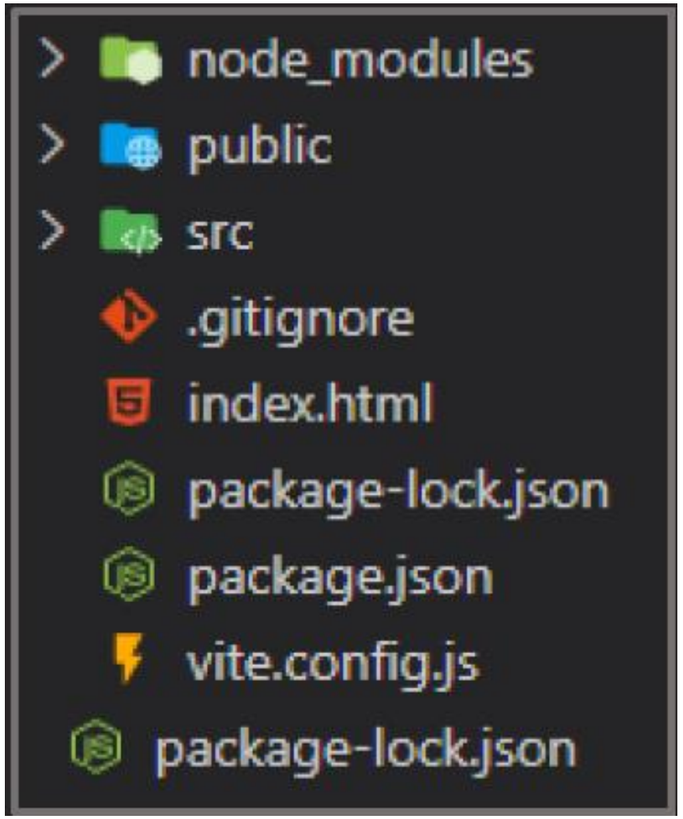
cd my-vite-app

npm install

npm run dev

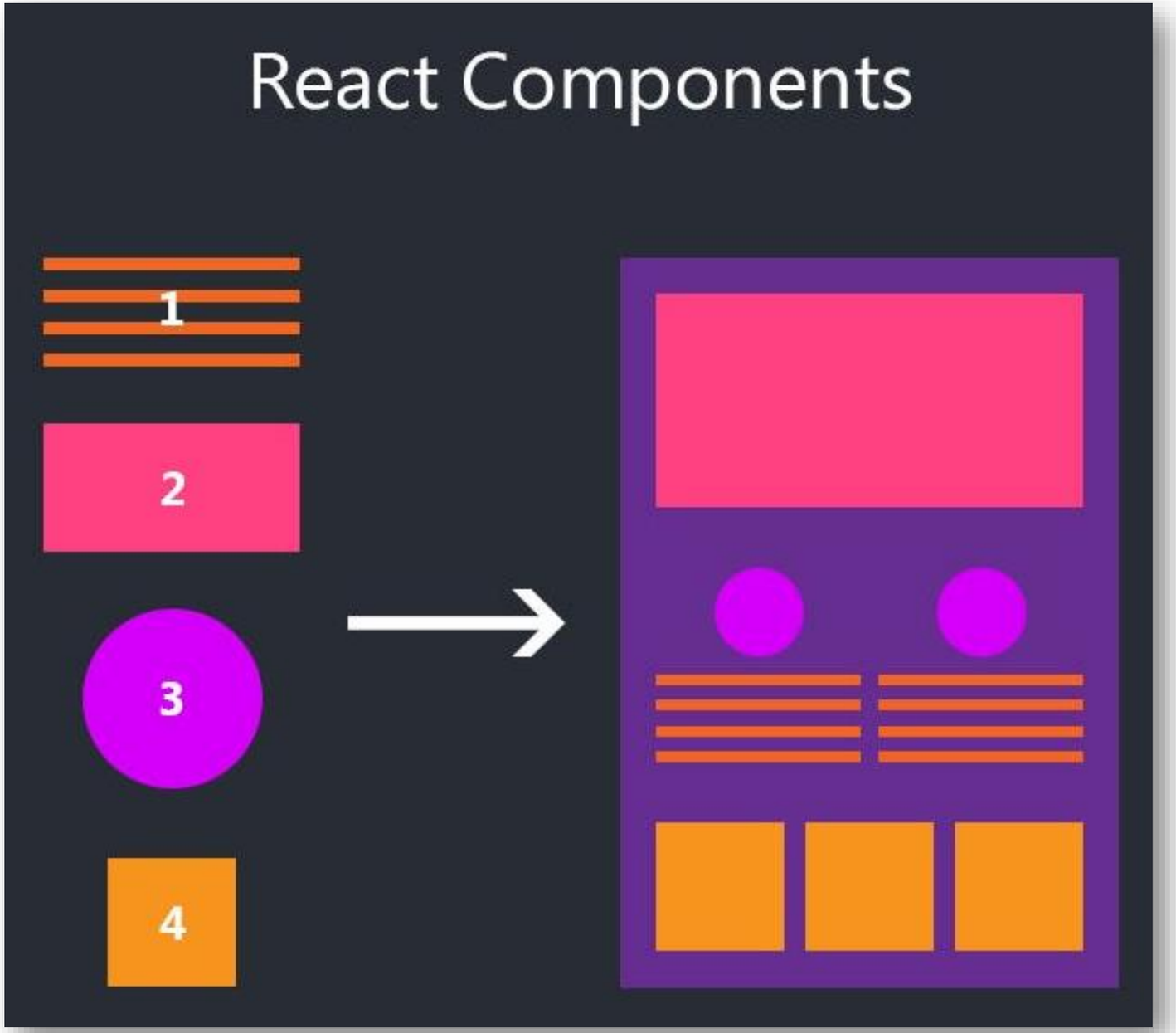
Folder Structure





Components

Little piece of reusable encapsuled code which could either contain state or not.



Hello World!

Inside SRC FOLDER, search main.jsx and write the following code

Now, function app is a new component

```
import React from 'react';
import ReactDOM from 'react-dom/client';

function App() {
    return <h1> Hola Mundo </h1>
}

ReactDOM.createRoot( document.getElementById('root')).render(
    <React.StrictMode>
        <App/>
    </React.StrictMode>
)
```

App Component

Inside SRC FOLDER, search main.jsx and write the following code

Inside SRC FOLDER, create a new file called App.jsx

```jsx
import './App.css'

const App = () => {
  return (
    <>
      <h1> Hola Mundo! </h1>
      <h2> Bienvenido </h2>
    </>
  )
}

export default App
```

/App.jsx

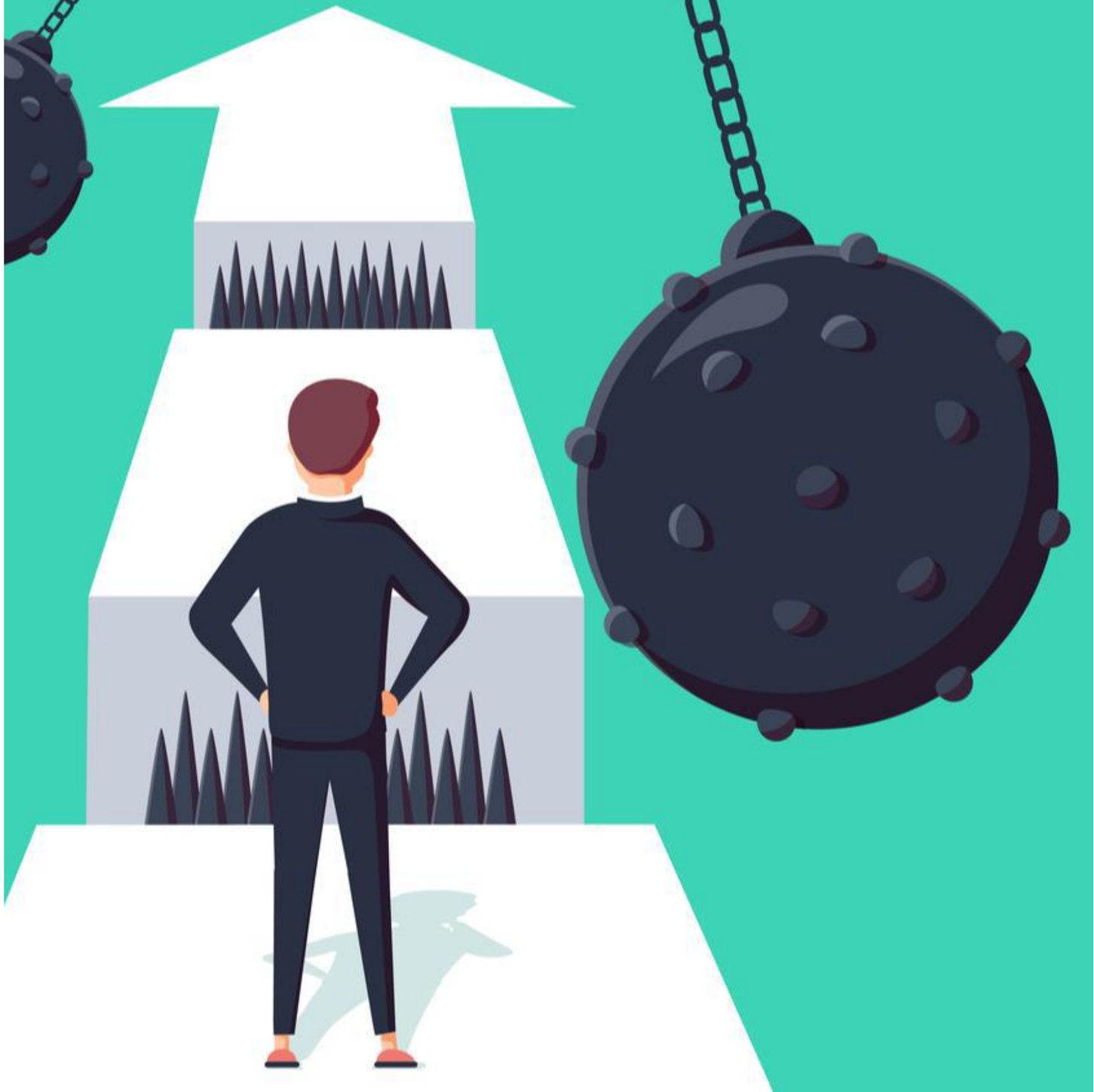
```jsx
import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App'
import './index.css'

ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
)
```

/main.jsx

CHALLENGE 02

1. Create a new FirstApp Component
2. Return a new title into h1 tag and a text “10” into one span tag
3. Import and Deploy FirstApp



Print variables into HTML

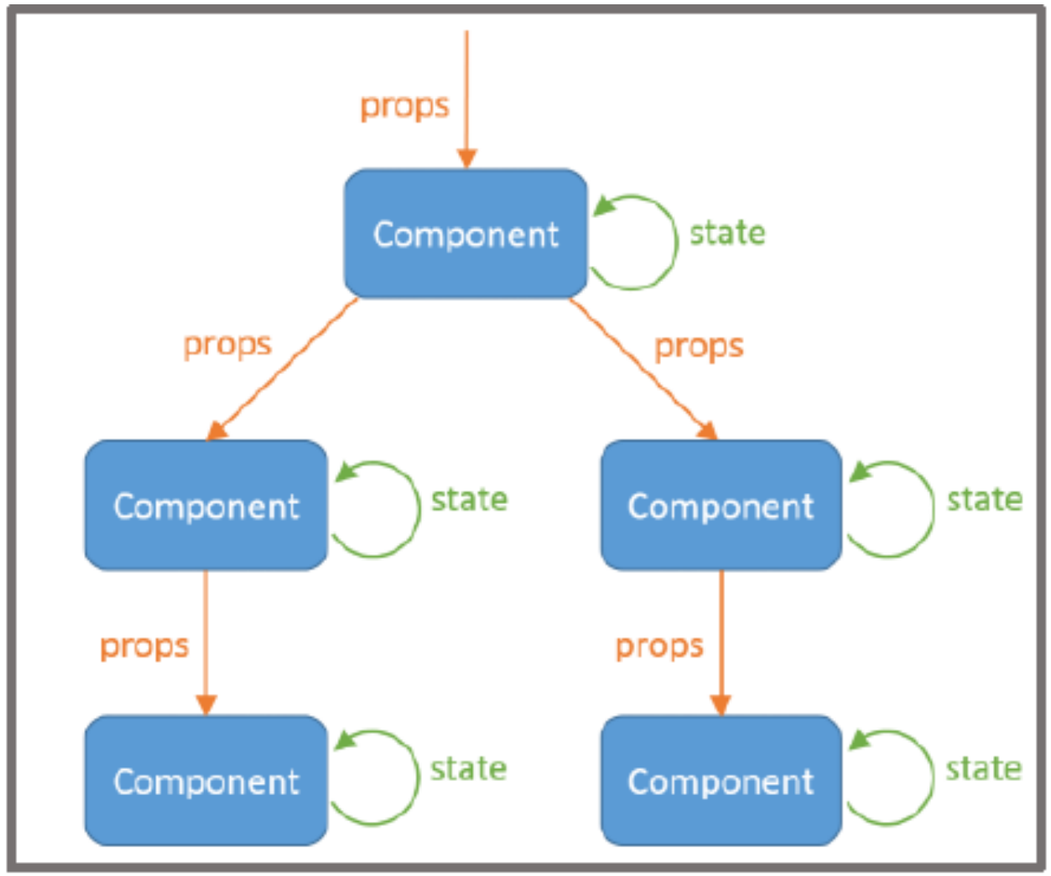
To print variables inside components, simply put them in curly braces:

const title = 'First App';

…

{ title }

States and Properties



Properties - PROPS

Use props to pass information/data from a parent component to child components.

Use props by sending attributes into the component tag and then add either a props parameter or a destructured object into the exported component

/main.jsx

```jsx
<React.StrictMode>
  <FirstApp title="My First App" />
</React.StrictMode>
```

First way
/FirstApp.jsx

```jsx
const FirstApp = ({title}) => {
    return (
        <>
            <h1> {title} </h1>
            <span> 10 </span>
        </>
    )
}

export default FirstApp;
```

Second way
/FirstApp.jsx

```jsx
const FirstApp = (props) => {
    return (
        <>
            <h1> {props.title} </h1>
            <span> 10 </span>
        </>
    )
}

export default FirstApp;
```

Props Validations

npm install prop types

Implements them by using **Component.propTypes** and assign a **PropTypes.TYPE** as value for each prop :

You can use default values into propTypes by using **defaultProps** as an object

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

const FirstApp = ({ title, sum } ) => {
  return (
    <>
      <h1> { title } </h1>
      <span> { sum } </span>
    </>
  )
}

FirstApp.propTypes = {
  title: PropTypes.string.isRequired,
  sum: PropTypes.number.isRequired
}

FirstApp.defaultProps = {
  title: 'No hay titulo',
  sum: 300
}

export default FirstApp
```

Functions

Functions are invoked by prop **events** in HTML

First Way:

onClick={ () => function(params) }

onClick={ (evt) => function(evt, params) }

Second Way:

onClick={ (**evt**) => function(**evt**) }

onClick={function}

evt is related to the event itself, will have some attributes like target and this one will have the value from the target

```
const FirstApp = ({ value} ) => {

  const handleAdd = () => {
    console.log( 'calling handleAdd' )
  }

  return (
    <>
      <h1> Counter </h1>
      <span> { value } </span>
      <button onClick={ () => handleAdd() }> +1 </button>
    </>
  )
}


export default FirstApp
```

https://www.w3schools.com/TAGs/ref_eventattributes.asp

Hooks - UseState

A special function that allows to connect to features from React

UseState : Hook that allow to add the React state to a functional component.

Use it by importing useState from react and then assign the useState function with an initial value into a destructured array

import { useState } from ‘react’;

…

const [count,setCount] = useState (initialValue);

```
import { useState } from "react"

const FirstApp = ({ value }) => {

  const [counter, setCounter] = useState( value )

  const handleAdd = () => {
    setCounter( counter + 1 )
  }

  return (
    <>
      <h1> Counter </h1>
      <span> { counter } </span>
      <button onClick={ () => handleAdd() }> +1  </button>
    </>
  )
}

export default FirstApp
```

CHALLENGE 03

1. Create two news buttons: handleSubstract and handleReset
2. Implement handleSubsstract by substracting from counter
3. Implement handleReset by setting counter as a default value. Remember the default value is a prop
4. Use useState hook to show working buttons into the web.