React JS

Francisco Javier Arocas Herrera | ALU0100906813
Daniel Barroso Rocío | ALU0101035826

Index

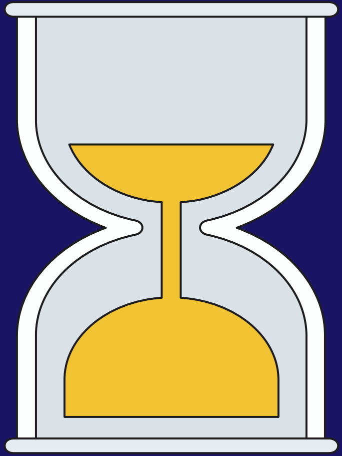
01

Introduction to Frameworks

What is a Framework?

It is a defined conceptual and technological support structure, normally with libraries and a programming structure.

Why should I use a framework? What problems do they solve for me?



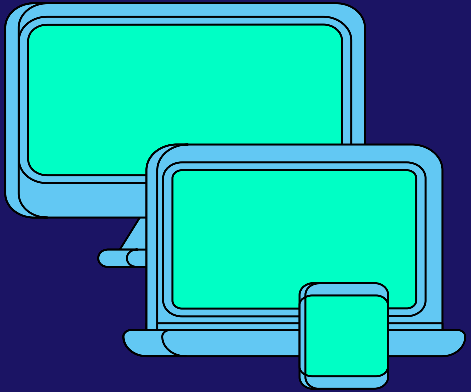
- Every time we change our application's state, we need to update the UI to match.

- **Predictability and maintainability are essential for the health and longevity of software**

Frameworks help developers

- They ease up a lot the communication between the developer and the DOM object.

- **Easier code -> Easier readability -> Easier maintainability -> Less time spent**



Vanilla JS

- Long code
- Needs good knowledge of DOM

```
function buildTodoItemEl(id, name) {
  const item =
document.createElement('li');
  const span =
document.createElement('span');
  const textContent =
document.createTextNode(name);
  span.appendChild(textContent);
  item.id = id;
  item.appendChild(span);
  return item;
}
```

ReactJS (JSX)

- Shorter code.
- Really comprehensible
- Looks just like HTML

```
render() {
  return (
    <li id={this.props.id}>
      <span>{this.props.name}</span>
    </li>
  );
}
```

Before using frameworks

- Easier to debug
- Easier to understand
- Predictable

Which framework should I use?

Angular

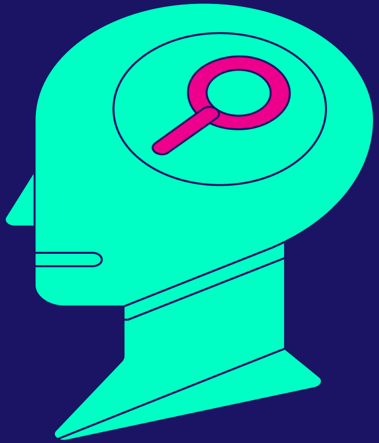
Ember

Vue

React

Ask yourself

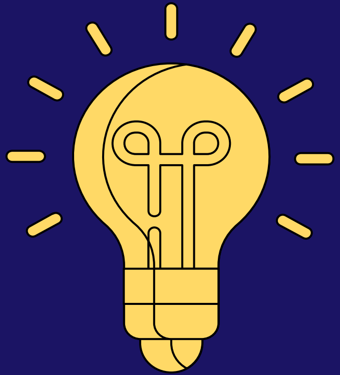
- What browsers does the framework support?
- What domain-specific languages (DSL) does the framework utilize?
- Does the framework have a strong community and good docs?



Matrix of the “Big Four”

	Browsers	Preferred DSL	Supported DSLs
Angular	IE9+	TypeScript	HTML-based; TypeScript
React	Modern (IE9+ with Polyfills)	JSX	JSX; TypeScript
Vue	IE9+	HTML-based	HTML-based, JSX, Pug
Ember	Modern (IE9+ in Ember version 2.18)	Handlebars	Handlebars, TypeScript

What is ReactJS?



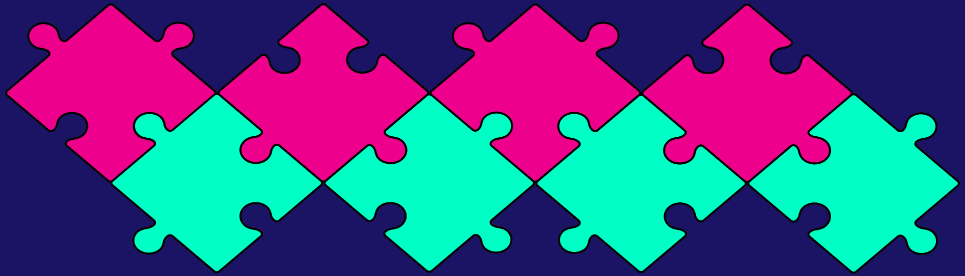
React

ReactJS is Declarative

- Easier to debug
- Easier to understand
- Predictable

ReactJS is Component Based

Build webs composing encapsulated components.



02

Main Concepts

Installation

Create-react-app

```
npx create-react-app my-app
cd my-app
npm start
```

Script tag

```
<script src="https://unpkg.com/react@17/umd/react.development.js" defer></script>
<script src="https://unpkg.com/react-dom@17/umd/react-dom.development.js" defer></script>
```

Introduction to JSX

1. Is a JavaScript syntax extension
2. It prevents injection attacks
3. It produces React Elements
4. It uses camelCase property naming instead HTML attribute names
   - ➔ className
   - ➔ tabIndex

Example of JSX Code

1. *Default JSX*

```
const title = <h1>Hello World</h1>;
```

2. *JSX with JavaScript Variables*

```
const city = "Santa Cruz";
const cityTitle = <h1>Welcome to {city}</h1>;
```

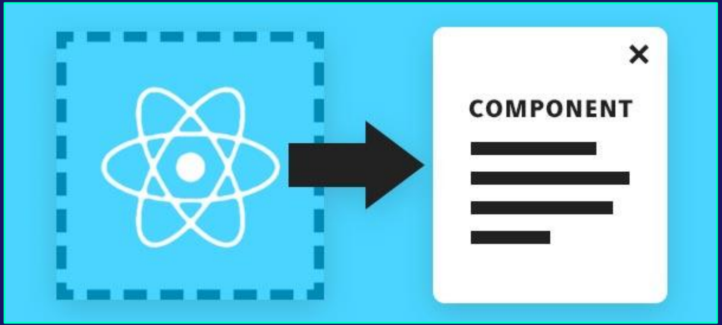
3. *Get values, calling javascript Variables*

```
const getFullName(firstName, surName) {
      return firstName + ' ' + surName;
}
const element = (
  <h1> Hello, {formatName(userName, userSurName)}
</h1>
);
```

Render Elements

Create an HTML web page

1. Create a DOM to Render
   a. Normally a div with id "root"
2. Pass it to ReactDom.render()
   a. The first component to render
   b. The DOM to render it



See "Hello World" Example

03

Components and Properties

What is a Component

"Components are like JavaScript functions. They accept arbitrary inputs (Properties) and return React elements describing what should appear on the screen". Its composed of:

➔ Properties

➔ States

See "WelcomeMessage" Example

About Properties

- Create properties, when render a method (Like HTML):
  "property"="value"
- Received by Class constructor (Don't forget super!)
- Access using this.property.name
- Properties are immutable

04

States of the Component

What is a State

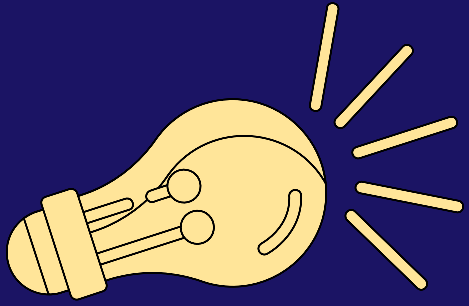
“State is an object which store the properties values that belongs to the component”.
“When the ***state change***, the component will ***render again***”

See “clock” and “CanvasClock” Example

About States (I)

1. Initialize it in the Constructor
2. Change state only using **setState()** function
   a. If use *this.state.property*, will **NOT** render the component again
3. Important Methods:
   a. ComponentDidMount()
   b. ComponentDidUnMount()
   c. ComponentWillUpdate()

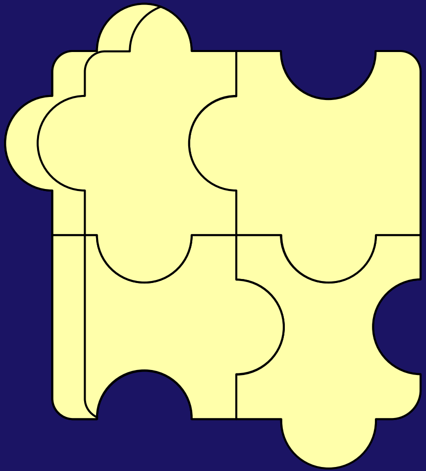
About States (II)



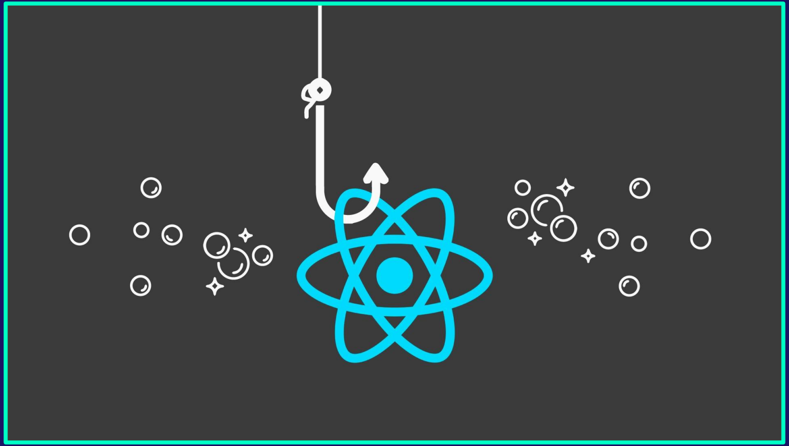
1. Remember **DO NOT UPDATE DIRECTLY**, use setState() function
2. Updates can be Asynchronous.
3. You can change one state at time, or various.

About States (III)

1. <u>**ComponentDidMount()**</u> : Called when the component is mounted (When created and after constructor)
2. <u>**ComponenDidUnMount()**</u> : Called when the componen is unMount (When is deleted)
3. <u>**ComponentWillUpdate()**</u> : Called when the state change



Hooks



Hooks are a new addition in React. They let you use state and other React features without writing a class.

Examples of Hooks

useState

useEffect

useContext

useReducer

useMemo

useRef

05

Handling Events of Components

Two differences from DOM

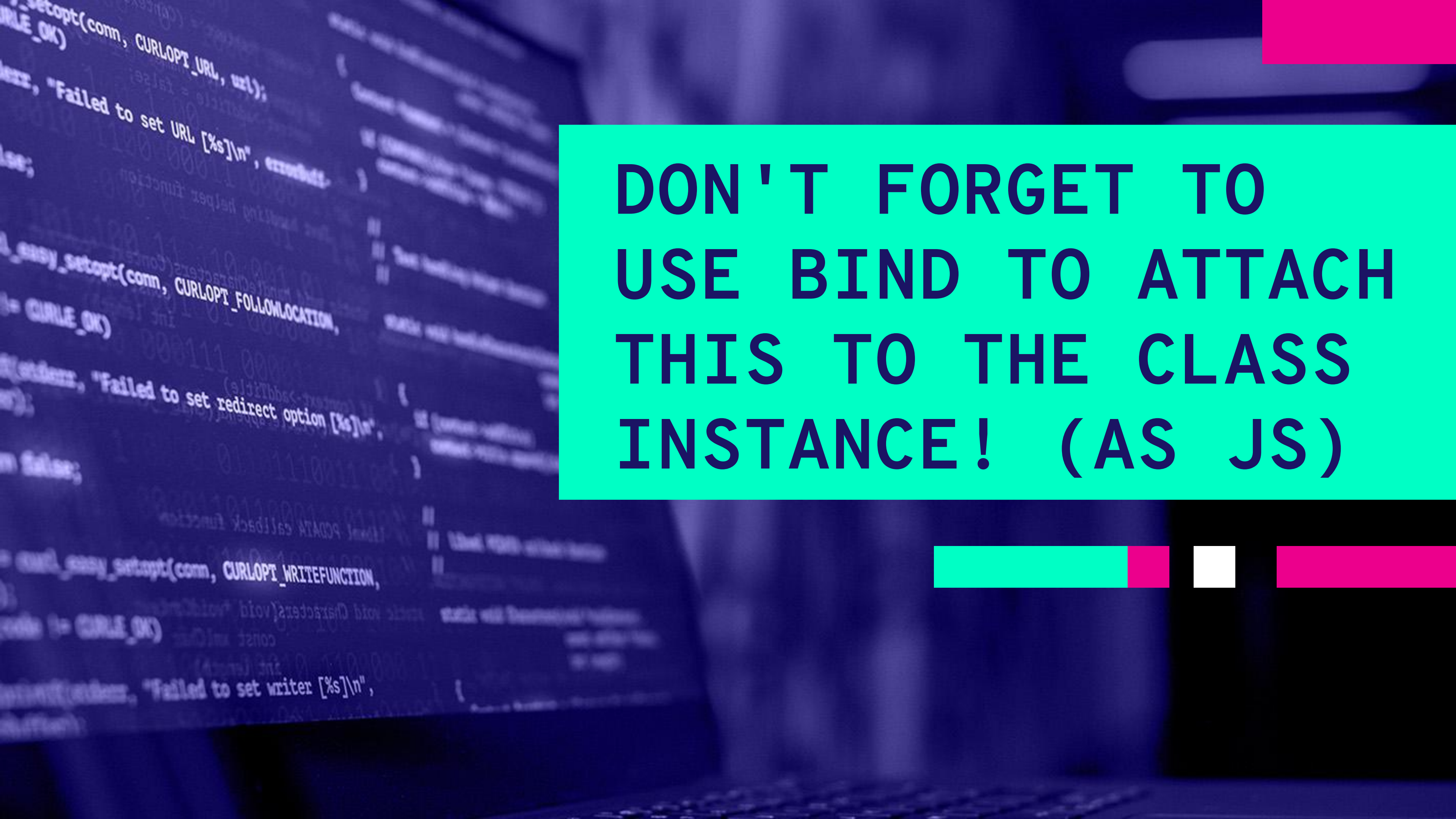
Use camelCase

React events are named using camelCase, rather than lowercase

Only Functions

With JSX you pass a function as the event, rather than a String

See "Colors" Example



DON'T FORGET TO USE BIND TO ATTACH THIS TO THE CLASS INSTANCE! (AS JS)

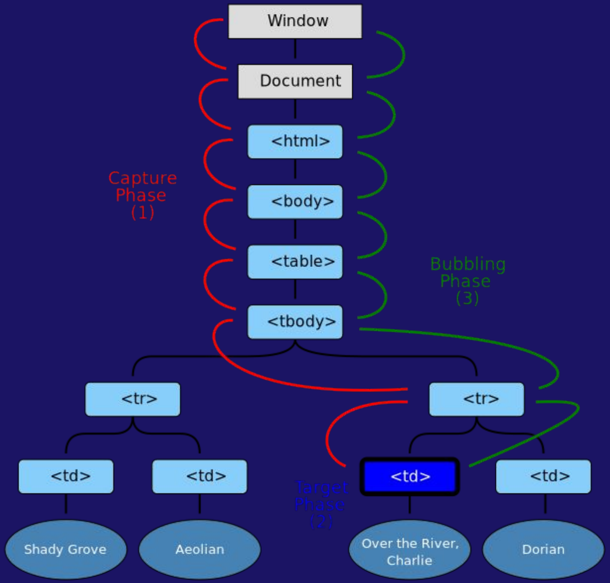
Pass arguments function

Arrow function

```
<button onClick={(e) => this.deleteRow(id, e)}>Delete Row</button>
```

This and Bind

```
<button onClick={this.deleteRow.bind(this, id)}>Delete Row</button>
```



Other events

Pointer

Image

Mouse

Forms

Touch

ClipBoard

Click to see all events

06

CONDITION RENDERING

Can use if/else

- Can use if/else to render one or other element.
- If we won't render an element, just return null in render function

See “Colors” Example

Condition Rending Example

```
function Greeting(props) {
  const isLogged = props.isLogged;

  return (
    if(isLogged) {
      return <UserGreeting/>;
    } else {
      return <GuestGreeting/>;
    }
  );
}
```

"Function will return differents components depending if user is logged or not!"

07

LISTS AND FORMS

List and keys

- Lists use map function to create each of its items

- Each item must have a key.

Example:

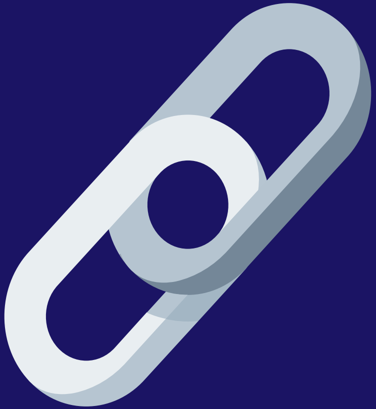
```
const listItems = todos.map((todo, index) =>
  <li key={index}>
     {todo.text}
   </li>
 );
```

See “Market” Example

Keys

A key is a property, which we will send to each element of the li DOM list.

The list key must be unique just in the context where it is created, not in the global context.



Forms and input “text”

- Inputs of the forms contain a state which stores the **value** of the input

- It updates the state when the user changes the input

Example:

```
handleChange(event) {
  this.setState({value: event.target.value});
}
```

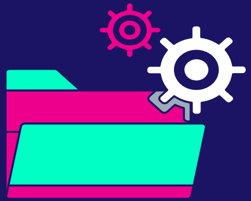
Forms and input “submit”

- When the user presses the submit button, it will call the handleSubmit function, because we put a onSubmit event into the form

- We use event.preventDefault(), because, if not, the page will send the form, and we get an error message.

Example:

```
handleSubmit(event) {
  alert('A name was submitted: ' + this.state.value);
  event.preventDefault();
}
```



Forms and select

- The options of the select tag, we don't use “select” to select the default options. Instead of this, we add a value attribute into the select

Example:

```
<select value={this.state.value}
onChange={this.handleChange}>
  <option value="grapefruit">Grapefruit</option>
  <option value="lime">Lime</option>
  <option value="coconut">Coconut</option>
  <option value="mango">Mango</option>
</select>
```

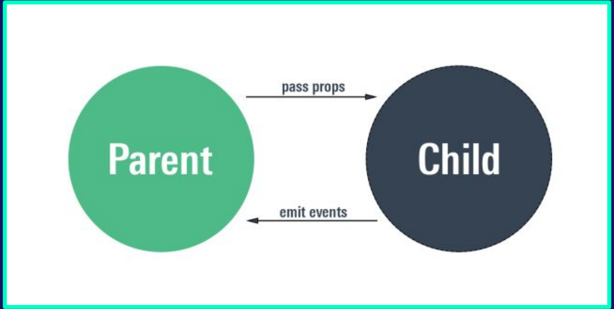
08

Lifting State Up

What is Lifting State Up?

"When various components with different states, if one of state change, need to change all states"

Click to see example on CodePen



To do this, parent must send a handling event to a children.

09

Composition VS Inheritance

Use of props.children

➔ Used to pass components to other components, where the component does not know which are their children.

➔ Also can send components as props

Click to see example on CodePen

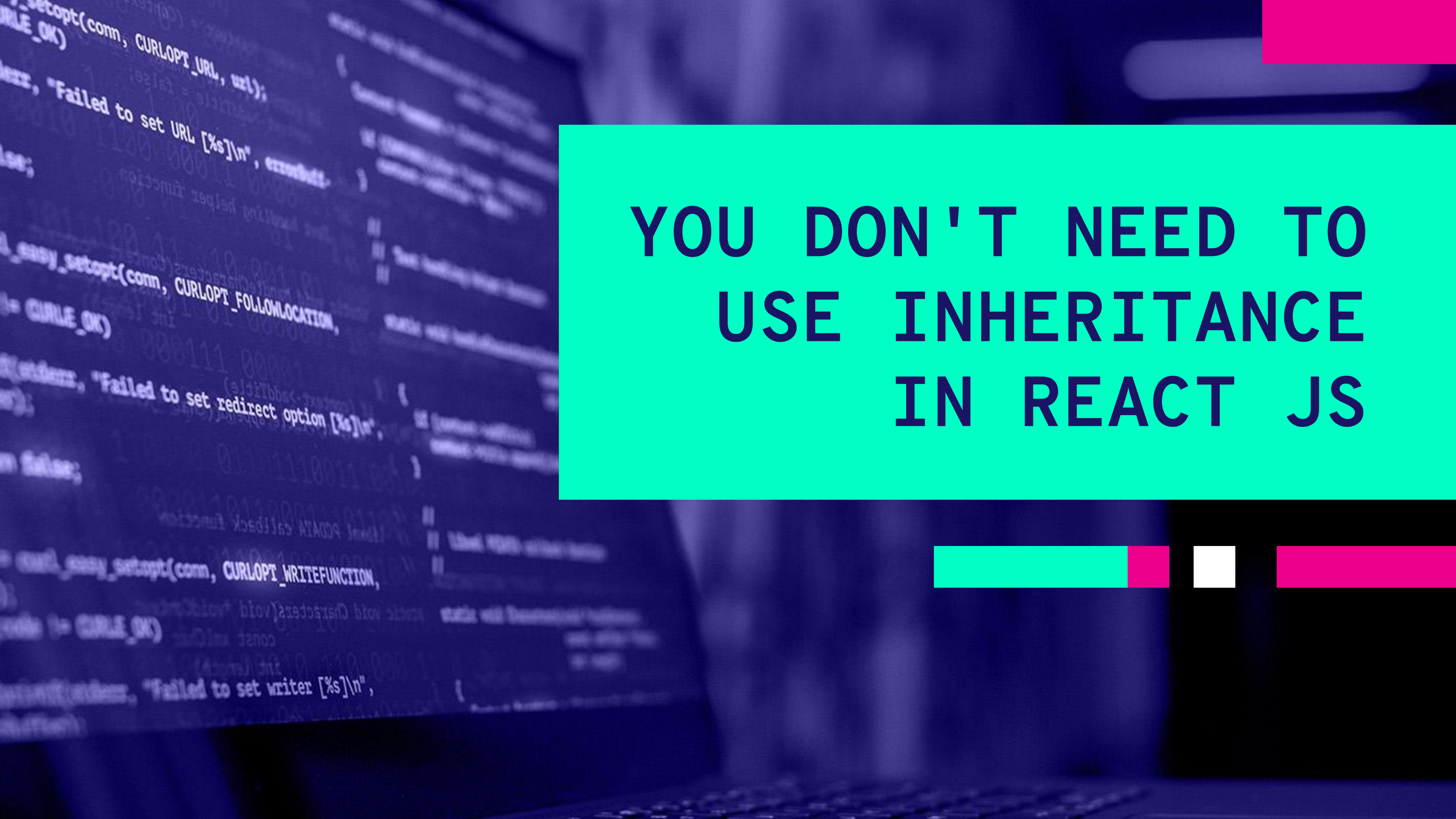
Inheritance

Derive properties and characteristics from one class to other

VS

Composition

Use objects inside other objects

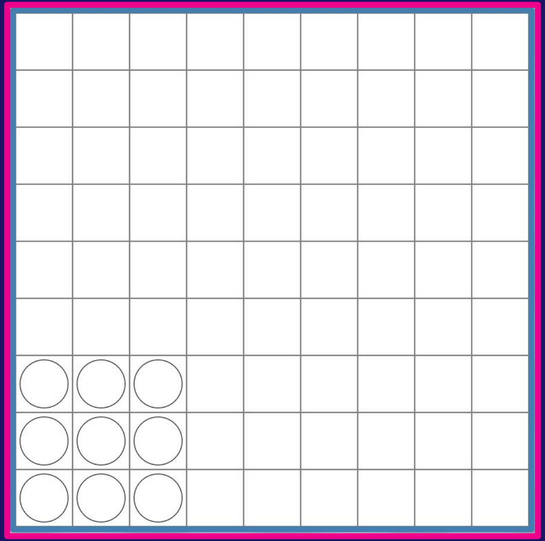


YOU DON'T NEED TO USE INHERITANCE IN REACT JS

10

Project Example

Halma Game



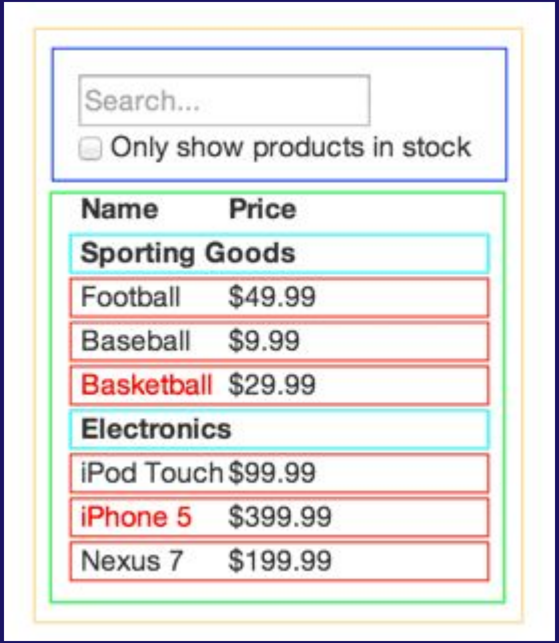
See “Halma” Example

11

Think in React

5 ReactJS tips from the creators of ReactJS

1. Break UI into a Hierarchy



Name	Price
Sporting Goods	
Football	$49.99
Baseball	$9.99
Basketball	$29.99
Electronics	
iPod Touch	$99.99
iPhone 5	$399.99
Nexus 7	$199.99

- Draw a rectangles of colors on each component.
- Use the **principle of single responsibility**

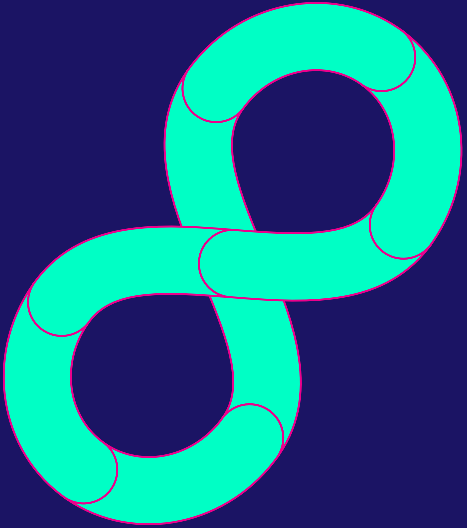
2. Create a static version

- → Start using props first, avoid to use states.
- → Remember differences between props and states
  - ◆ Props: It is passed to the component
  - ◆ State: Created and managed inside the component
- → Then, start to use states

3. Identify the minimal state

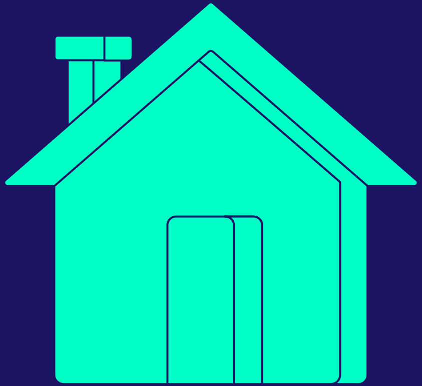
- → Think in the minimal quantity of state.
- → Use **DRY (Don't repeat yourself)**
- → 3 Questions
    1. Is it passed in from a parent via props?
    2. Does it remain unchanged over time?
    3. Can you compute it based on any other state or props in your component

If answer is yes... **IT ISN'T A STATE**



4. Where state should live

- Identify which components change or owns this state.
- This is often the most challenging part for the new people in React



5. Add inverse data flow

- Now start with under components to upper components.
- To do this, use the: Lifting State Up
  - Remember to pass states to parents using handle events

THANKS!

Do you have any questions?

Francisco Javier Arocas Herrera (francisco.arocas.37@ull.edu.es)
Daniel Barroso Rocío (daniel.barroso.rocio.25@ull.edu.es)