Mastering React.js: A Comprehensive Crash Course

Introduction

Introduction to React.js

History of React.js

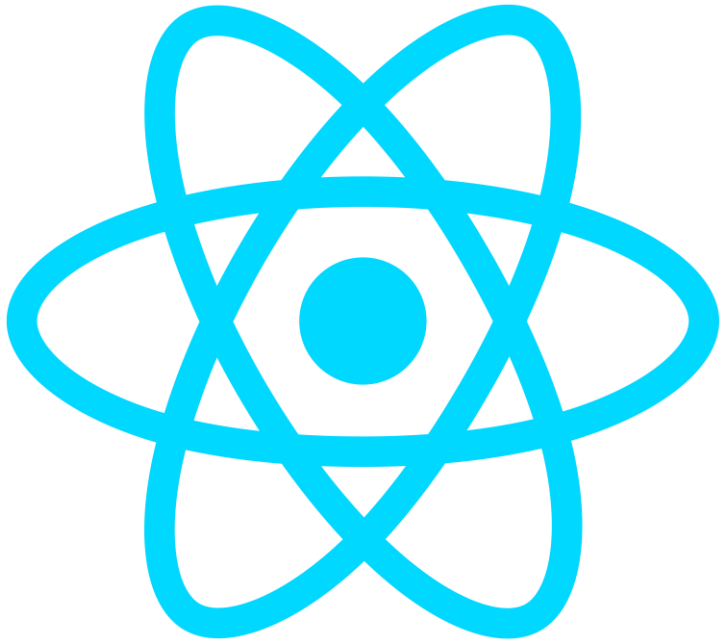
Why React.js over Vanilla JavaScript

Setting up development environment

Creating React.js project

Introduction to React.js

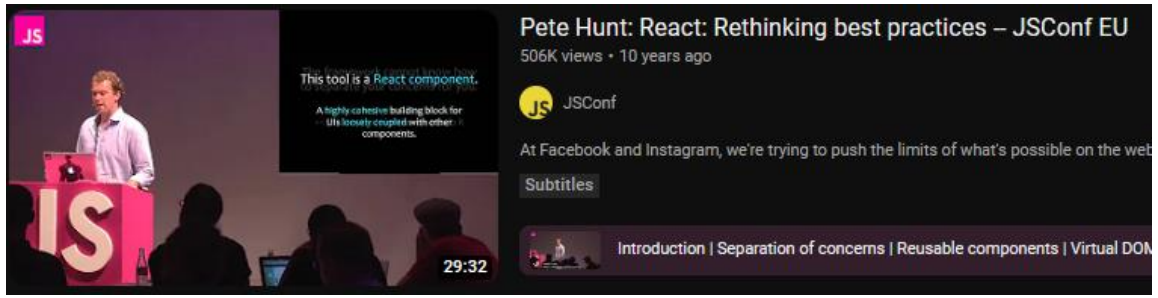
- React.js is a **JavaScript library** used for building **user interfaces (UIs)** and **single-page applications.**
- Created by **Jordan Walke** at **Facebook.**
- Most popular JavaScript library for frontend development.



History of React.js

- React.js began as an internal tool for dynamic Facebook components.
- Created in 2011 but remained private initially.
- React.js was open-sourced by Jordan Walke and Tom Occhino at JSConfUS 2013.
- Released with the belief that its success at Facebook could benefit others.
- Initially faced criticism for combining JavaScript and HTML in a single file.
- It was widely hated and criticized because of its combination of JavaScript and HTML in single file.
- It slowly gained traction and blew up in adoption.

First Keynote given by Tom Occhino and Jordan Walke at JSConfUS 2013.



Second Keynote given by Pete Hunt at JSConfEU.

Enable Inline Keywords

React.js History in 10minutes: The Revolutionary Journey from Facebook to...

6K views • 10 days ago

Thapa Technical

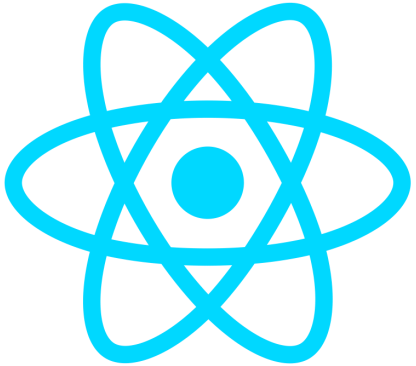
Discover the story of React.js in our exciting documentary, "Unveiling React.js: The Revolutionary Journey from Facebook to ...

CC

Watch this Documentary to know complete history of React.js

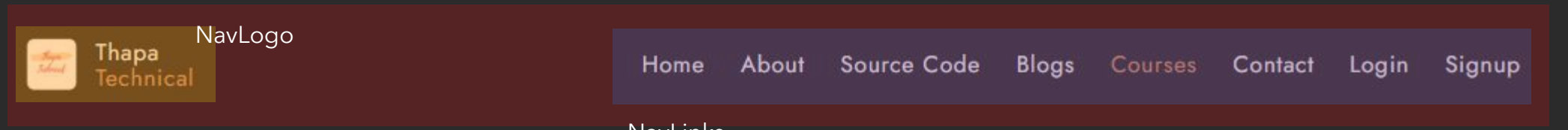
Is React JS Library or Framework?

- React is not a framework. React is a JavaScript library for building user interfaces.
- It is also known as ReactJS and React.js, so don't get confused if you read different notation in different places.
- React knows only one thing that is to create an awesome UI.



React is all about Components

Navbar



NavLogo

NavLinks

PageHeader

CourseSearchInput

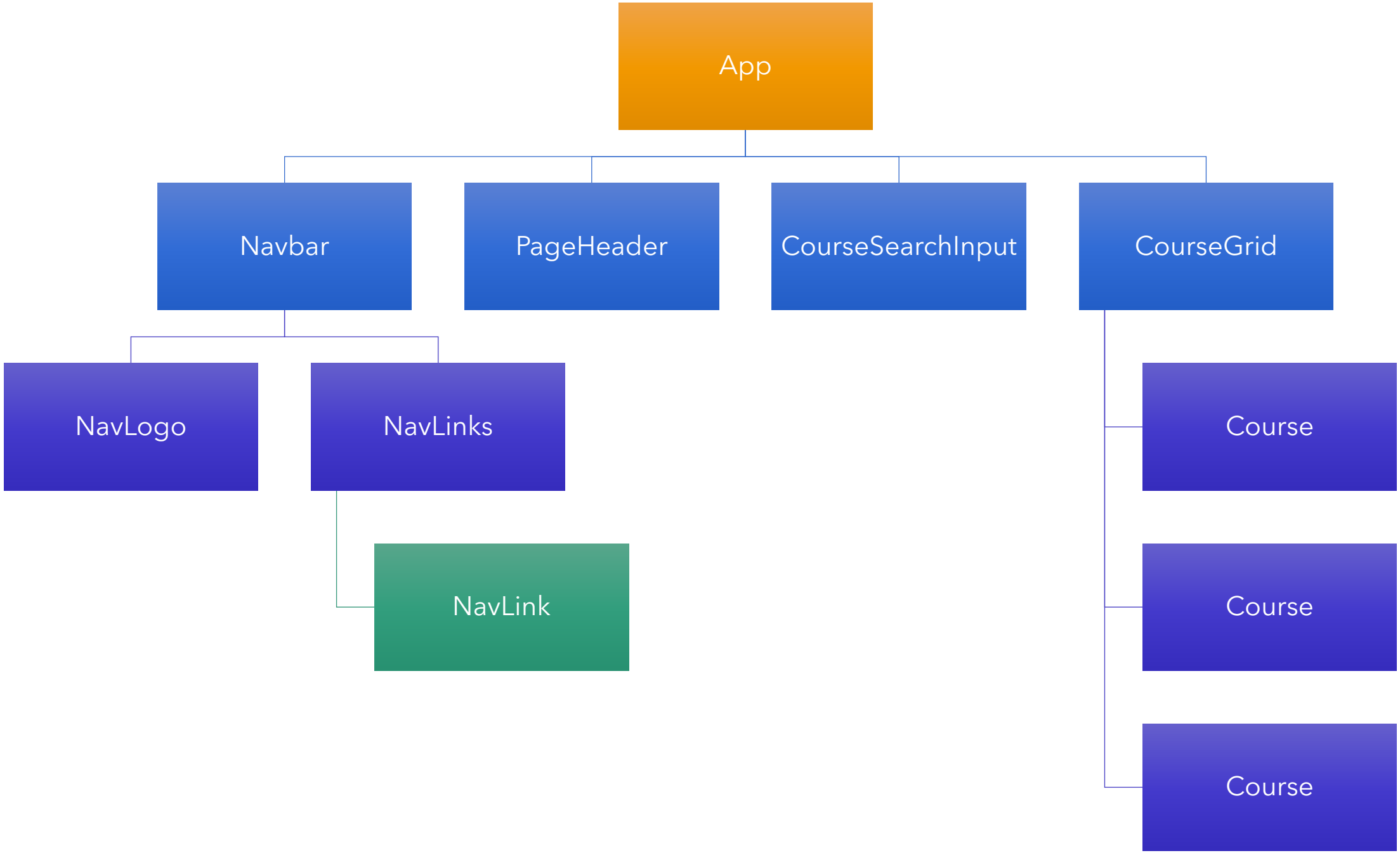
CourseCard

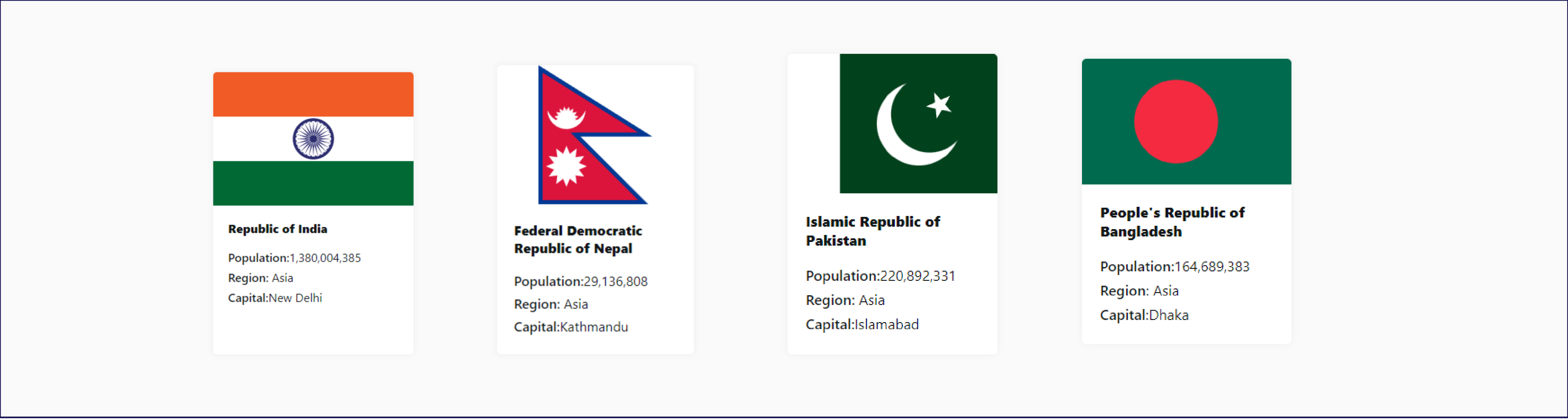
CourseCard

CourseCard

Here, **CourseCard** is shown 3 times. Traditionally, you would have to copy paste each of these but with React.js you can avoid duplicate code **(Do not Repeat Yourself, DRY).**

React.js application is just tree of such Components.


App
Navbar
PageHeader
CourseSearchInput
CourseGrid
NavLogo
NavLinks
NavLink
Course
Course
Course




```
<a href="/country/Republic%20of%20India">
  <img
    src="path_to_india_flag.svg"
    alt="Indian national flag"
    class="IndiaFlag"
  />
  <div class="countryInfo">
    <p class="countryTitle">Republic of India</p>
    <div>
      <p>
        <span class="infoTopic">Population:</span> 1,380,004,385
      </p>
      <p>
        <span class="infoTopic">Region:</span> Asia
      </p>
      <p>
        <span class="infoTopic">Capital:</span> New Delhi
      </p>
    </div>
  </div>
</a>
```


```
<a href="/country/Republic%20of%20India">
  <img
    src="path_to_india_flag.svg"
    alt="Indian national flag"
    class="IndiaFlag"
  />
  <div class="countryInfo">
    <p class="countryTitle">Republic of India</p>
    <div>
      <p>
        <span class="infoTopic">Population:</span> 1,380,004,385
      </p>
      <p>
        <span class="infoTopic">Region:</span> Asia
      </p>
      <p>
        <span class="infoTopic">Capital:</span> New Delhi
      </p>
    </div>
  </div>
</a>
```

```
<a href="/country/Republic%20of%20India">
  <img
    src="path_to_india_flag.svg"
    alt="Indian national flag"
    class="IndiaFlag"
  />
  <div class="countryInfo">
    <p class="countryTitle">Republic of India</p>
    <div>
      <p>
        <span class="infoTopic">Population:</span> 1,380,004,385
      </p>
      <p>
        <span class="infoTopic">Region:</span> Asia
      </p>
      <p>
        <span class="infoTopic">Capital:</span> New Delhi
      </p>
    </div>
  </div>
</a>
```

```
<a href="/country/Republic%20of%20India">
  <img
    src="path_to_india_flag.svg"
    alt="Indian national flag"
    class="IndiaFlag"
  />
  <div class="countryInfo">
    <p class="countryTitle">Republic of India</p>
    <div>
      <p>
        <span class="infoTopic">Population:</span> 1,380,004,385
      </p>
      <p>
        <span class="infoTopic">Region:</span> Asia
      </p>
      <p>
        <span class="infoTopic">Capital:</span> New Delhi
      </p>
    </div>
  </div>
</a>
```

Republic of India

Population:1,380,004,385
Region: Asia
Capital:New Delhi

Federal Democratic Republic of Nepal

Population:29,136,808
Region: Asia
Capital:Kathmandu

Islamic Republic of Pakistan

Population:220,892,331
Region: Asia
Capital:Islamabad

People's Republic of Bangladesh

Population:164,689,383
Region: Asia
Capital:Dhaka

```jsx
return (
  <div className={`container ${styles.countriesCards}`}>
    {filteredCountries.map((country) => (
      <CountryCard country={country} key={country.name.common} />
    ))}
  </div>
);
```

```jsx
<Link to={`/country/${country.name.official}`}>
  <img
    src={country.flags.svg}
    alt={country.flags.alt}
    className={styles.countryFlag}
  />
  <div className={styles.countryInfo}>
    <p className={styles.countryTitle}>{country.name.official}</p>
    <div>
      <p>
        <span className={styles.infoTopic}>Population:</span>
        {country.population.toLocaleString()}
      </p>
      <p>
        <span className={styles.infoTopic}>Region:</span> {country.region}
      </p>
      <p>
        <span className={styles.infoTopic}>Capital:</span>
        {country.capital[0]}
      </p>
    </div>
  </div>
</Link>
```

React is Declarative

Why React.js ?

- React is declarative because it describes what the UI should look like rather than how to achieve it. This makes the code easier to read and maintain, as it is more focused on the end result rather than the steps involved in getting there.

```jsx
function MyComponent({ name }) {
    return <div>Hello, {name}!</div>;
}
```

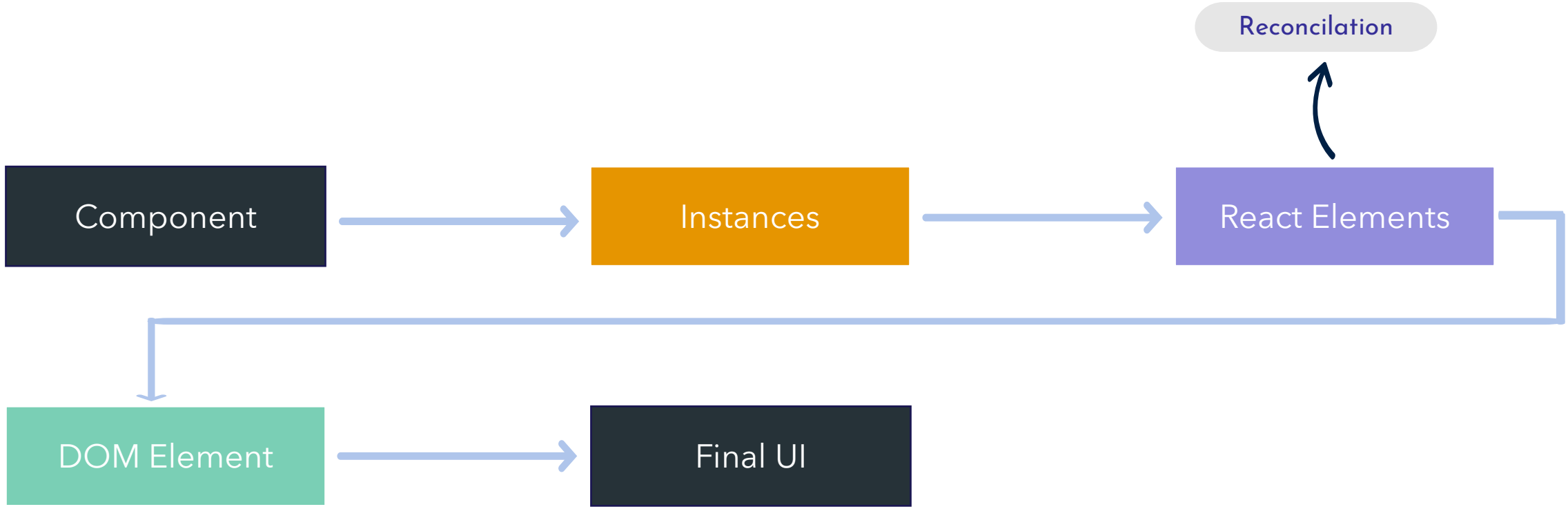
React - Declarative

JS - Imperative

```jsx
function MyComponent(name) {
    const element = document.createElement('div');  // Create a new div element
    element.textContent = `Hello, ${name}!`;        // Set text content manually
    return element;                                 // Return the created element
  }
```

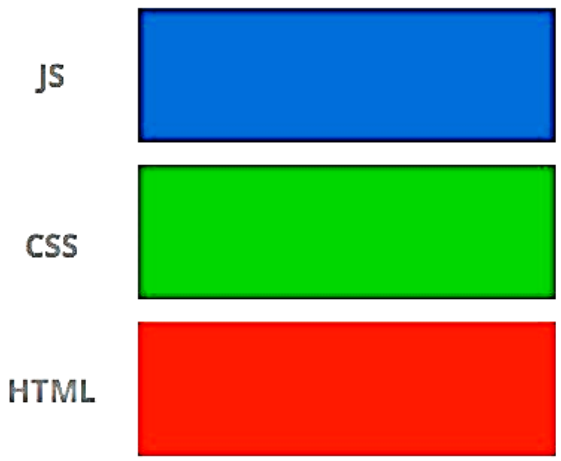
- Here, we are manually creating elements, setting their properties, and appending them to the DOM.

How React works?

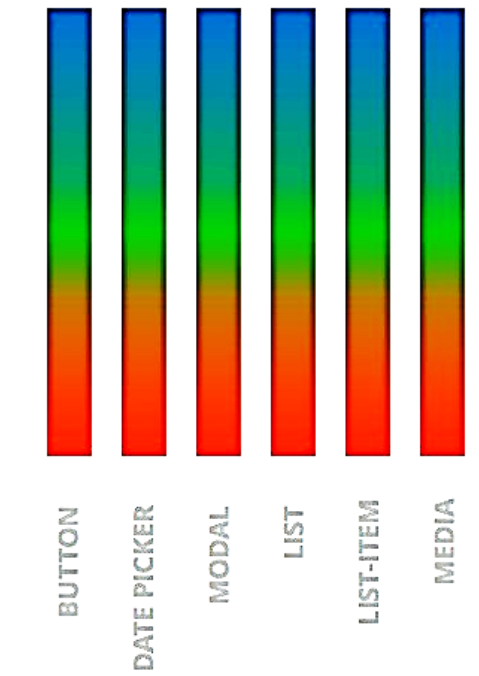


Why React.js over Vanilla JavaScript?

- Oppose to Vanilla JavaScript, React has a concept called **components** which combines all HTML, CSS and JavaScript by features instead of separating HTML, CSS and JavaScript completely.
- React.js manages updating the DOM or Document Object Model with the components written by us.

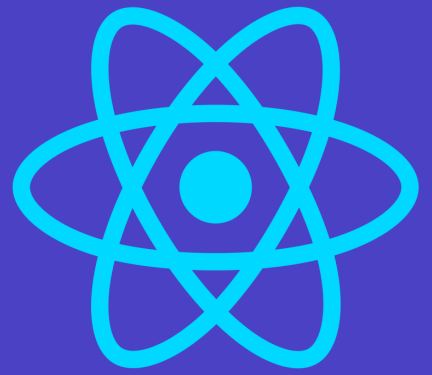


Traditional Way



React.js

Why React.js?



- React is the Most Popular JavaScript library for building user interfaces.
- Component-Based Architecture
- Declarative UI
- Rich Ecosystem – npm packages
- Strong Community Support – Online / Github