# Introduction to React

- What is React?
- Working of DOM
- Problems with JS
- Working of React
- JS Vs React
- Intro to Components

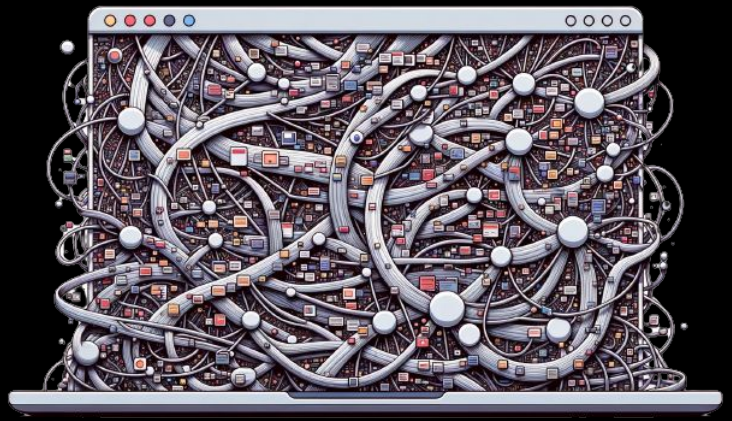
React

# What is React


1. JavaScript library to build Dynamic and interactive user interfaces
2. Developed at Facebook in 2011.
3. Currently most widely used JS library for front-end development.
4. Used to create single page application (page does not re-load).

React

# Working of DOM



1. Browser takes HTML and create DOM.
2. JS helps us modify DOM based on user actions or events.
3. In big applications, Working with DOM becomes complicated.

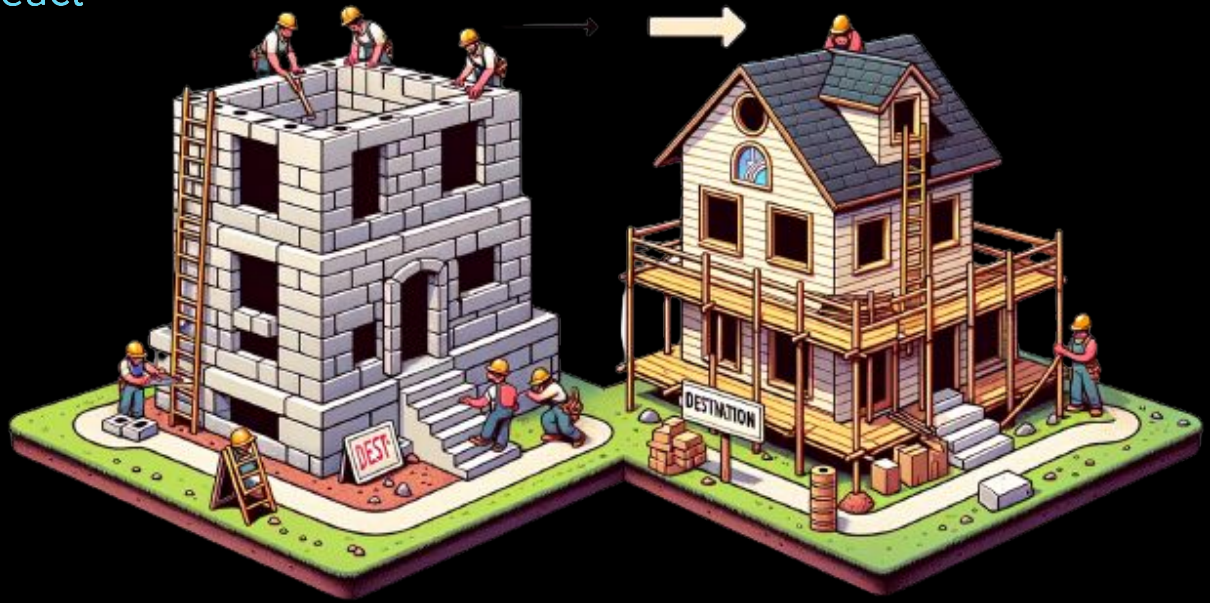
# Problems with JavaScript

1. React has a simpler mental model
2. JS is cumbersome
3. JS is Error-prone
4. JS is Hard to maintain

# Working of React


1. No need to worry about querying and updating DOM elements.
2. React creates a web page with small and reusable components
3. React will take care of creating and updating DOM elements.
4. IT saves a lot of time, cheezein aasan hai, pahele se likhi hui hain

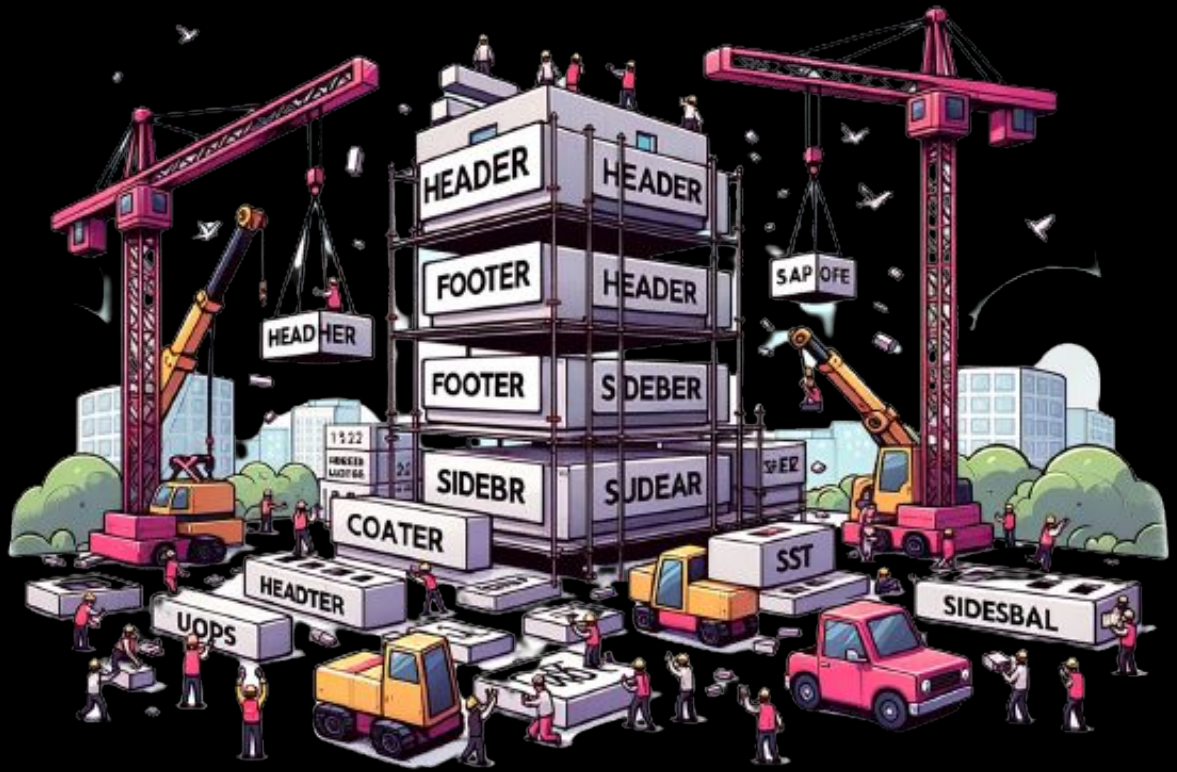
# JS Vs React




1. JS is imperative: You define steps to reach your desired state.
2. React is Declarative: You define the target UI state and then react figures out how to reach that state.

React

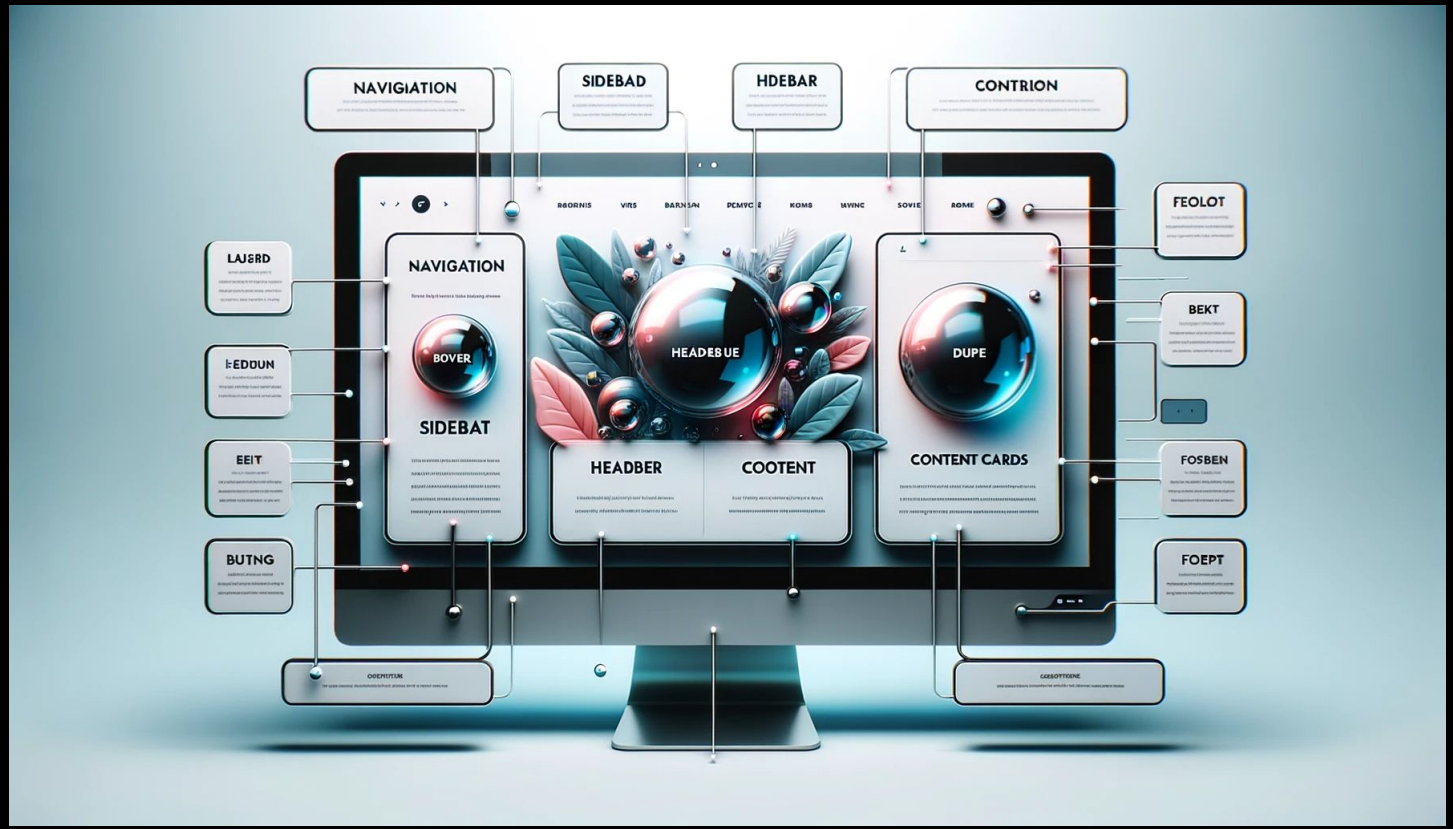
# Introduction to Components



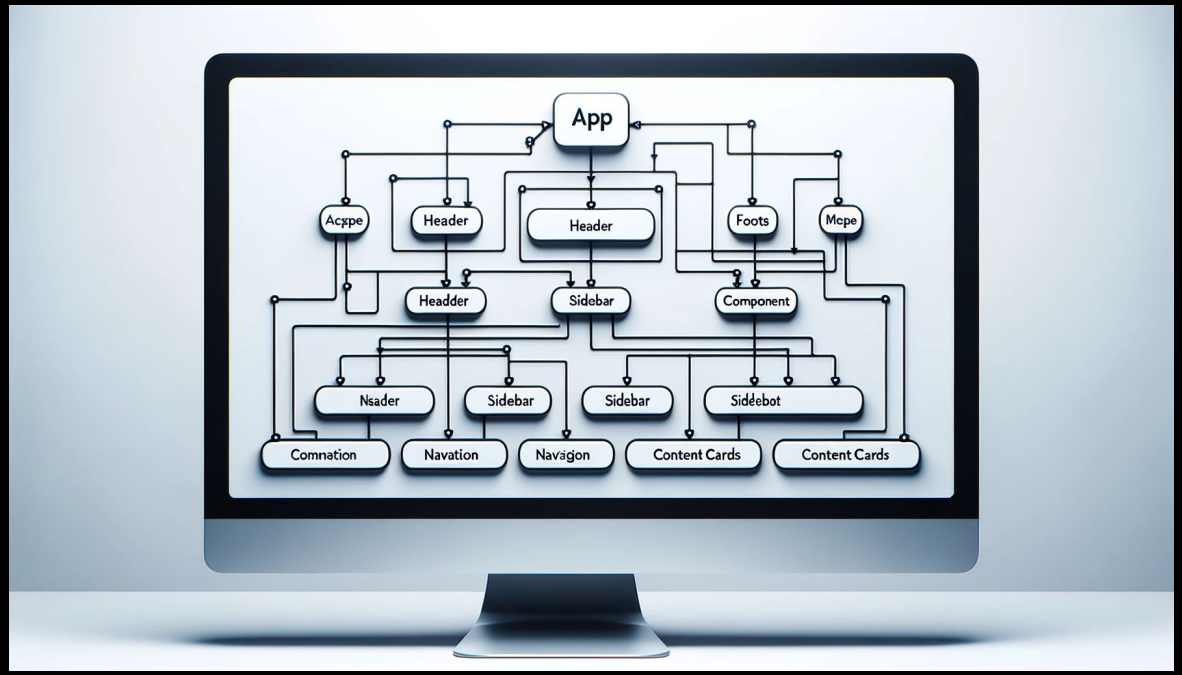

Components help us write reusable, modular and better organized code.

React

# Introduction to Components



# Introduction to Components




React application is a tree of components with App Component as the root bringing everything together.

# Create a React App

Official tool is CRA(Create React APP)

## Quick Start

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

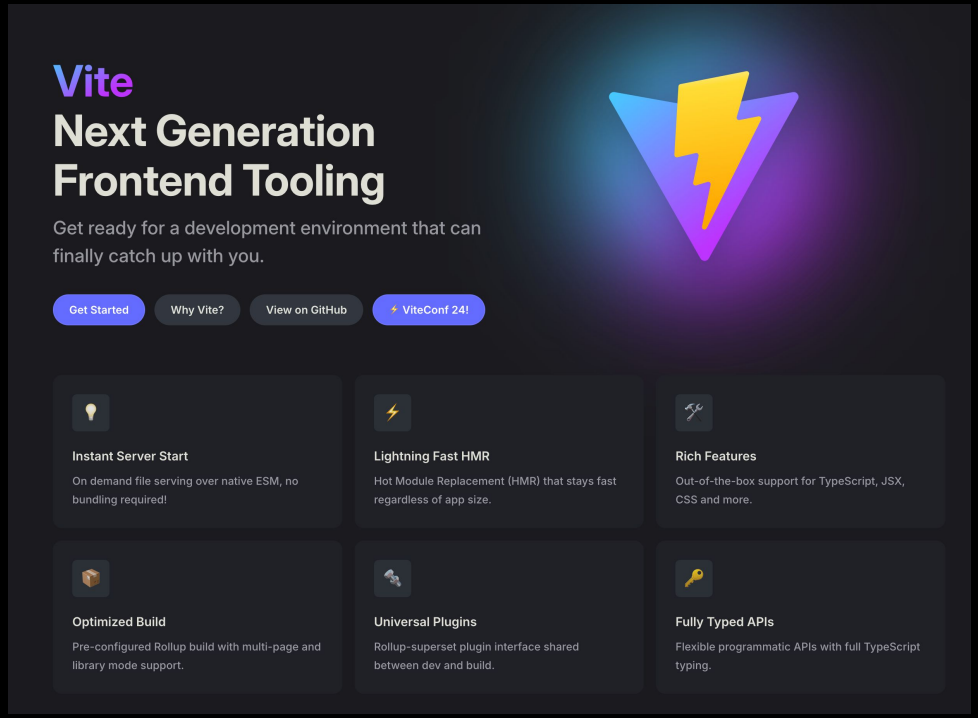

- npm: You need to install a package first (using npm install) before you can use it.
- npx: You can directly run a package without installing it, using npx <package-name>.

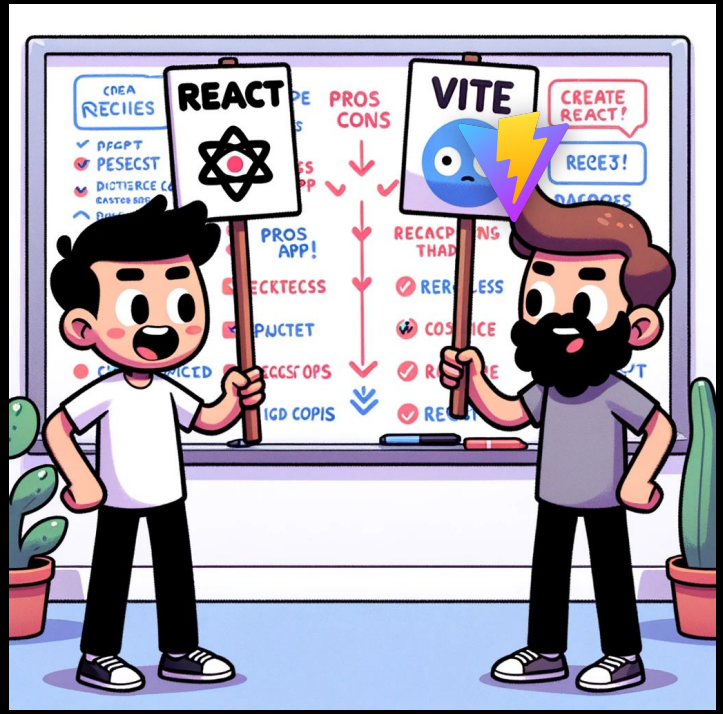
# Project Structure

React

- node_modules
- public
  - favicon.ico
  - index.html
  - logo192.png
  - logo512.png
  - manifest.json
  - robots.txt
- src
  - App.css
  - App.js
  - App.test.js
  - index.css
  - index.js
  - logo.svg
  - reportWebVitals.js
  - setupTests.js
- .gitignore
- package-lock.json
- package.json
- README.md

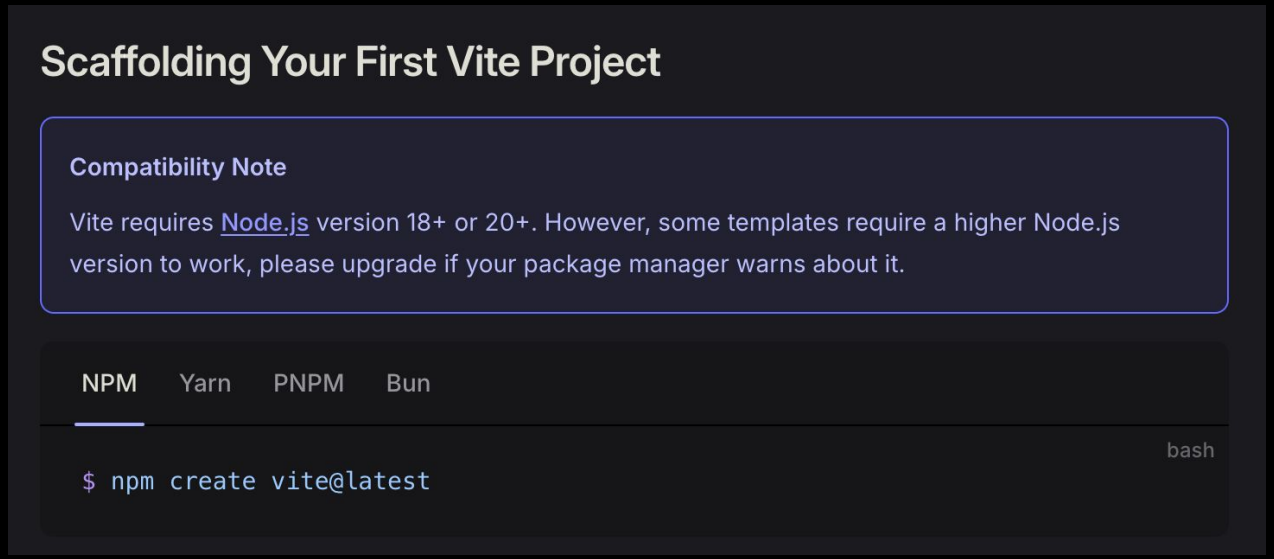
# Create a React App




1. Vite is a modern tool to create React Project.
2. Vite produces Quick and Small bundle size.

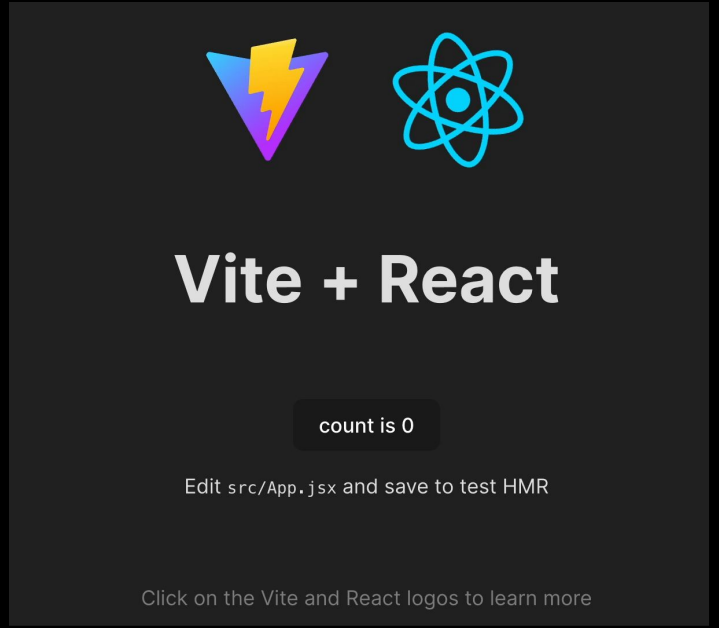
# Create a React App




1. Vite is a modern tool to create React Project.
2. Vite produces Quick and Small bundle size.

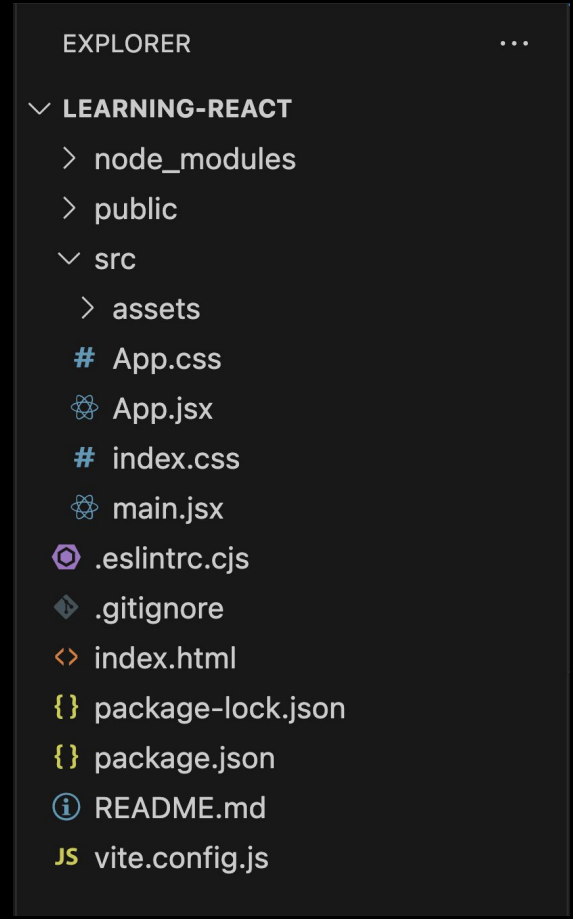
# Create a React App

```
Scaffolding project in /Users/prashantjain/workspace/Test Project/react/test2...

Done. Now run:

  cd test2
  npm install
  npm run dev

prashantjain@Prashants-Mac-mini react % cd test2
prashantjain@Prashants-Mac-mini test2 % npm install

added 215 packages, and audited 216 packages in 26s

99 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
prashantjain@Prashants-Mac-mini test2 % npm run dev

> test2@0.0.0 dev
> vite
```



# Project Structure

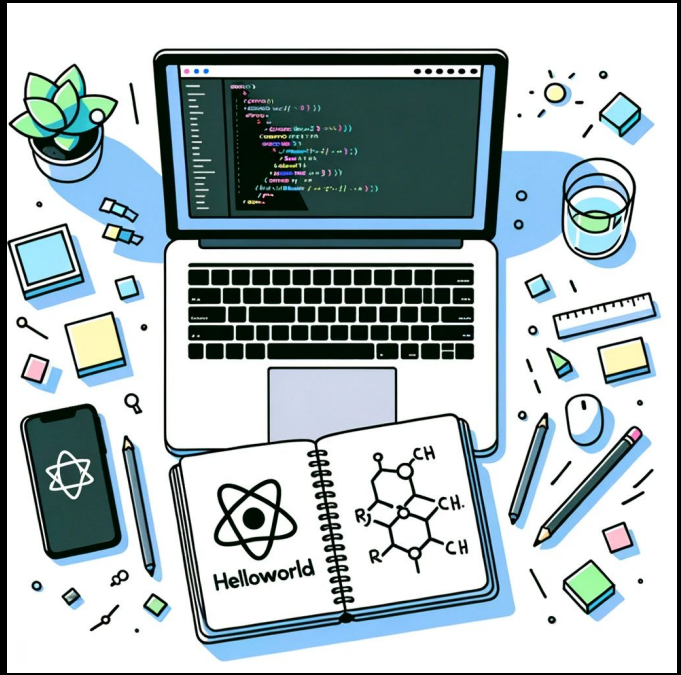
1. node_modules/ has all the installed node packages
2. public/ Directory: Contains static files that don't change.
3. src/ Directory: Main folder for the React code.
    1. components/: Reusable parts of the UI, like buttons or headers.
    2. assets/: Images, fonts, and other static files.
    3. styles/: CSS or stylesheets.
4. package.json contains information about this project like name, version, dependencies on other react packages.
5. vite.config.js contains vite config.




React

# First React Component

- File Extensions
- What is JSX?
- Class vs Function Components
- Exporting component
- Other important Points
- Dynamic Components
- Reusable Components



React

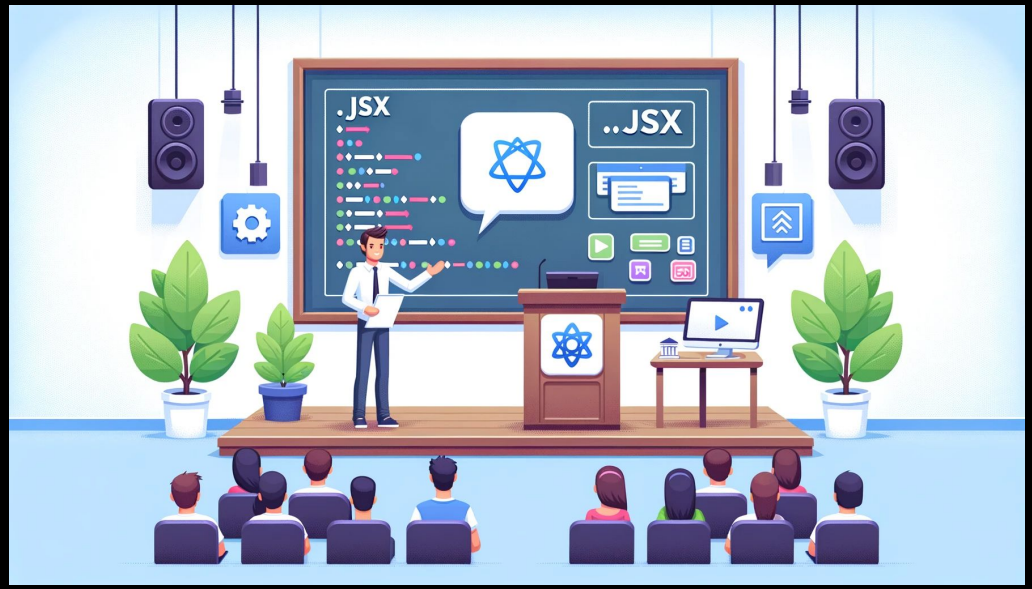
# File Extensions


## .JS

- Stands for JavaScript
- Contains regular JavaScript code
- Used for general logic and components

## .JSX

- Stands for JavaScript XML
- Combines JavaScript with HTML-like tags
- Makes it easier to design UI components

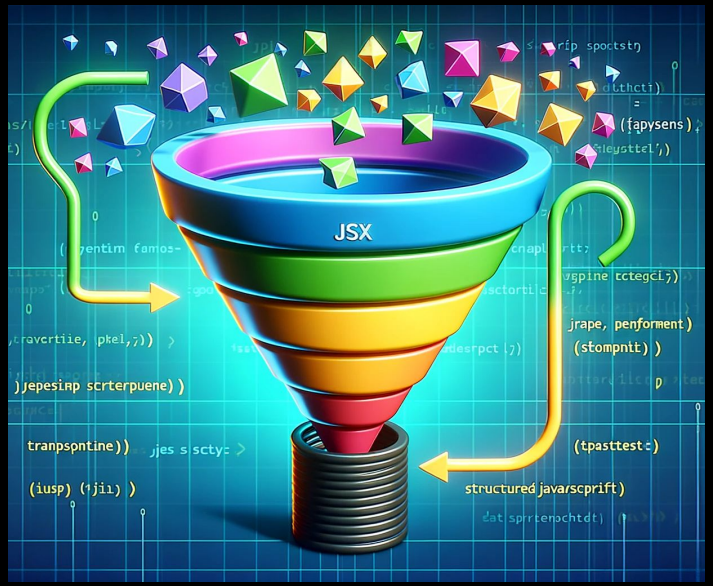



Technically, there is no difference between .js and .jsx files in terms of functionality. The separation is purely a convention to differentiate between files that predominantly contain JSX from those that are purely JavaScript.

# What is JSX?

React

1. Definition: JSX determines how the UI will look wherever the component is used.
2. Not HTML: Though it resembles HTML, you're actually writing JSX, which stands for JavaScript XML.
3. Conversion: JSX gets converted to regular JavaScript.
4. Babeljs.io/repl is a tool that allows you to see how JSX is transformed into JavaScript.




# JSX Example

```jsx
function App() {
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <p>
          Edit <code>src/App.js</code> and save to reload.
        </p>
        <a
          className="App-link"
          href="https://reactjs.org"
          target="_blank"
          rel="noopener noreferrer"
        >
          Learn React
        </a>
      </header>
    </div>
  );
}
```

React

# Class vs Function Components

## Class Components

- Stateful: Can manage state.
- Lifecycle: Access to lifecycle methods.
- Verbose: More boilerplate code.
- Not Preferred anymore.

## Functional Components

- Initially stateless.
- Can use Hooks for state and effects.
- Simpler and more concise.
- More Popular.


## Class Components

```
import React, { Component } from 'react';

class Heading extends Component {
  render() {
    return <h1>{this.props.title}</h1>;
  }
}

export default Heading;
```

## Function Components

```
import React from 'react';

const Heading = (props) => {
  return <h1>{props.title}</h1>;
};

export default Heading;
```

# Using CSS


est1 > src > # KGButton.css > ...

```css
.custom-button {
  background-color: #4CAF50;
  color: white;
  padding: 10px 20px;
  border: none;
  border-radius: 5px;
  font-size: 16px;
  cursor: pointer;
  transition: background-color 0.3s ease;
}

.custom-button:hover {
  background-color: #45a049;
}
```

```jsx
import React from 'react';
import './KGButton.css';

const KGButton = () => {
  return (
    <button className="custom-button">
      Click Me!
    </button>
  );
};

export default KGButton;
```

# Exporting/Importing components

Component.js

```
export default
function
Button() {
  ...
}
```

one default export

Components.js

```
export function
Slider() {
  ...
}
```

```
export function
Checkbox() {
  ...
}
```

multiple named exports

MixedComponents.js

```
export function
Avatar() {
  ...
}
```

```
export default
function
FriendsList() {
  ...
}
```

named export(s)
and one default export


1. Enables the use of a component in other parts.
2. Default Export: Allows exporting a single component as the default from a module.
3. Named Export: Allows exporting multiple items from a module.
4. Importing: To use an exported component, you need to import it in the destination file using import syntax.

# Exporting/Importing components

```jsx
import React from 'react';
import './Paragraph.css';

const Paragraph = () => {
  return <p className="custom-paragraph">
    This is a styled paragraph component demonstrating
     how to apply external CSS in React.
  </p>;
};

export default Paragraph;
```

```css
.custom-paragraph {
  font-size: 18px;
  color: #333;
  line-height: 1.6;
  margin-bottom: 20px;
  font-family: 'Arial', sans-serif;
}
```

```jsx
import React from 'react';
import './Heading.css';

const Heading = () => {
  return <h1 className="custom-heading">
    Custom Components
  </h1>;
};

export default Heading;
```

```css
.custom-heading {
  font-size: 32px;
  color: #4CAF50;
  margin: 20px 0;
  font-weight: bold;
  font-family: 'Arial', sans-serif;
}
```

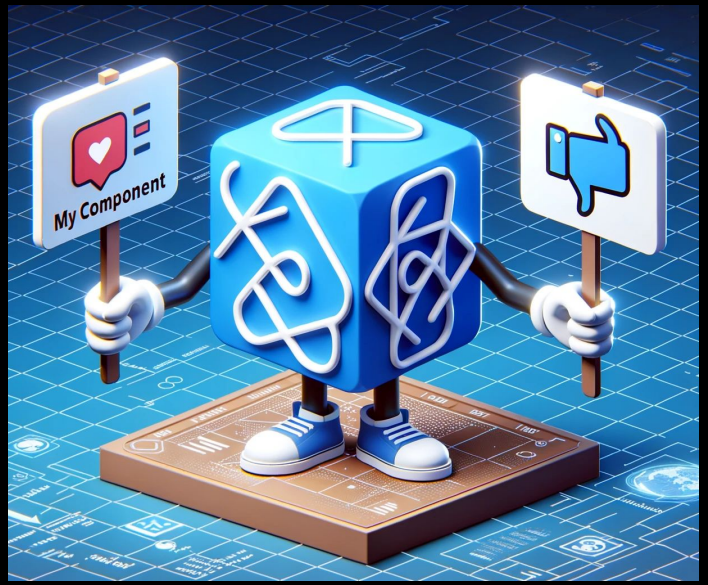

# Exporting/Importing components

```
import React from 'react';
import Button from './Button';
import Paragraph from './Paragraph';
import Heading from './Heading';

function App() {

  return (
    <div>
      <Heading/>
      <Paragraph />
      <Button />
    </div>
  );
}

export default App;
```


React

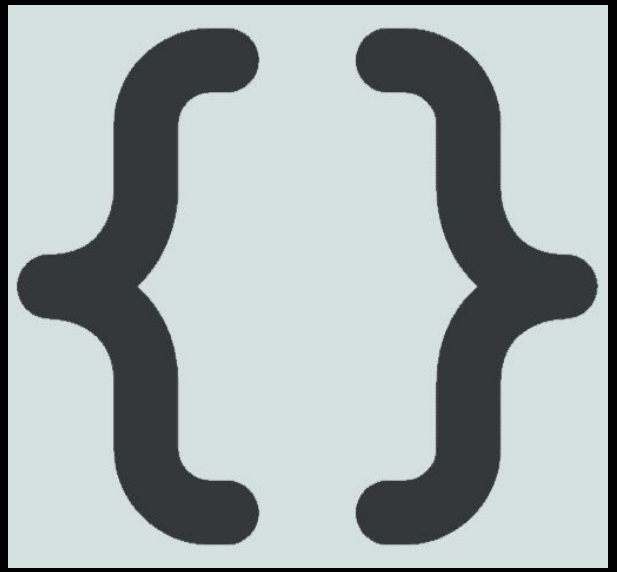
# Important Points

1. Naming: Must be capitalized; lowercase for default HTML.
2. HTML: Unlike vanilla JS where you can't directly write HTML, in React, you can embed HTML-like syntax using JSX.
3. CSS: In React, CSS can be directly imported into component files, allowing for modular and component-specific styling.



# Dynamic Components

1. Dynamic Content: JSX allows the creation of dynamic and interactive UI components.
2. JavaScript Expressions: Using {}, we can embed any JS expression directly within JSX. This includes variables, function calls, and more.




# Dynamic Components

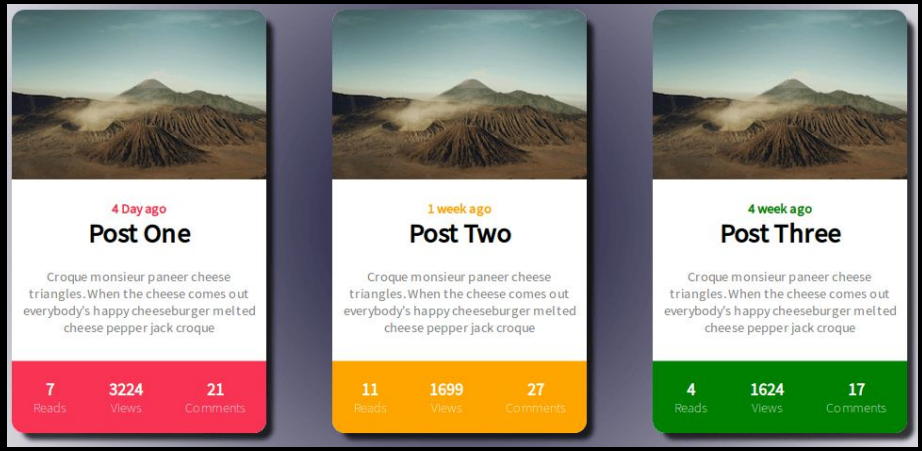
```
function Hello() {
  let myName = 'Prashant';
  let number = 456;
  let fullName = () => {
    return 'Prashant Jain';
  }

  return <p>
    MessageNo: {number} {myName} your master {fullName()}
  </p>
}

export default Hello;
```


# Dynamic Components

```jsx
import React from 'react';

const fruits = ['Apple', 'Banana', 'Orange', 'Grapes', 'Mango'];

const List = () => {
  return (
    <ul>
      {fruits.map((fruit, index) => (
        <li key={index}>{fruit}</li>
      ))}
    </ul>
  );
};

export default List;
```

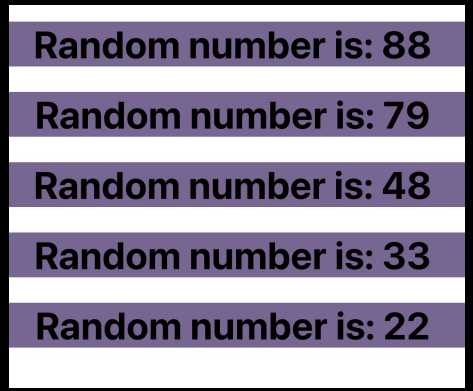
# Reusable Components

1. Modularity: Components are modular, allowing for easy reuse across different parts of an application.
2. Consistency: Reusing components ensures UI consistency and reduces the chance of discrepancies.
3. Efficiency: Reduces development time and effort by avoiding duplication of code.
4. Maintainability: Changes made to a reused component reflect everywhere it's used, simplifying updates and bug fixes.



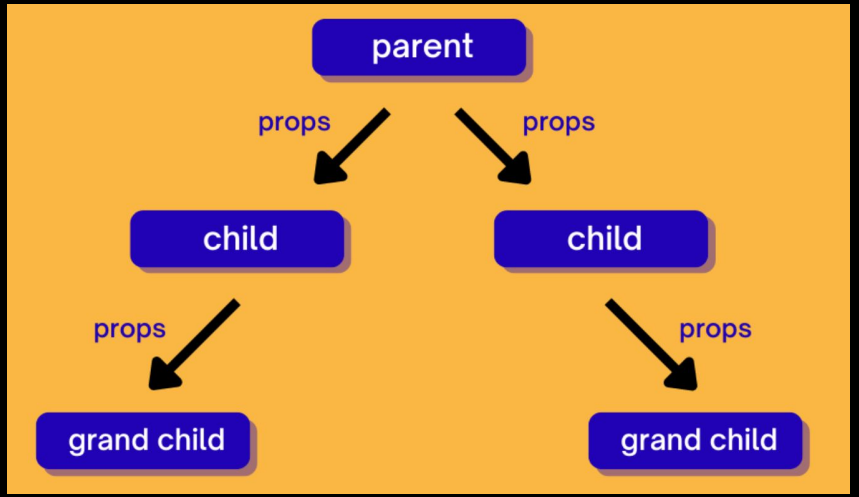
# Reusable Components

```jsx
function Random() {
  let number = Math.random() * 100;
  return <h1 style={{'background-color': '#776691'}}>
    Random number is: {Math.round(number)}
  </h1>
}

export default Random;
```




```jsx
function App() {
  return (
    <div className="App">
      <Random></Random>
      <Random></Random>
      <Random></Random>
      <Random></Random>
      <Random></Random>
    </div>
  );
}
```

# Passing Data via Props




## Props in React

- Short for properties
- Mechanism for passing data.
- Read-only by default

## Usage

- Pass data from parent to child component.
- Makes components reusable.
- Defined as attributes in JSX.

## Key Points

- Data flows one-way (downwards).
- Props are immutable.
- Used for communication between components.

## Examples

```
<Header title="My App" />
```


# Using Props

```jsx
function Heading(props) {
  return <h1>{props.text}</h1>;
}
function Paragraph(props) {
  return <p>{props.content}</p>;
}
function Button(props) {
  return <button>{props.label}</button>;
}
```

```jsx
<>
  <Heading text="Welcome to My Website" />
  <Paragraph content="This is a simple
  paragraph." />
  <Button label="Click Me" />
</>
```

1. Purpose: Parameters (props) are used to pass data from a parent component to a child component in React.
2. Access: In a component, you access these parameters via props.
3. Dynamic Content: Props allow components to render dynamic content based on the passed data.


# Using Props

```jsx
import Button from "./Button";

const Post = () => {
  const handleClick = () => {
    alert("Button clicked!");
  };

  return (
    <div>
      <h2>Post Title</h2>
      <p>This is a post content.</p>
      <Button text="Click Me" type="success-btn"
       onClick={handleClick} />
    </div>
  );
};

export default Post;
```

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

const Button = ({ text, type, onClick }) => {
  return (
    <button className={type} onClick={onClick}>
      {text}
    </button>
  );
};

export default Button;
```


React

React

# What is Bootstrap

1. Responsive: Mobile-first design for all device sizes.
2. Components: Pre-styled elements like buttons and navbars.
3. Customizable: Modify default styles as needed.
4. Cross-Browser: Consistent look across browsers.
5. Open-Source: Free with community support.

React

Including Bootstrap

# Get started any way you want

Jump right into building with Bootstrap—use the CDN, install it via package manager, or download the source code.

Read installation docs →

## Install via package manager

Install Bootstrap's source Sass and JavaScript files via npm, RubyGems, Composer, or Meteor. Package managed installs don't include documentation or our full build scripts. You can also use any demo from our Examples repo to quickly jumpstart Bootstrap projects.

```
$ npm install bootstrap@5.3.3
```

```
$ gem install bootstrap -v 5.3.3
```

Read our installation docs for more info and additional package managers.

## Include via CDN

When you only need to include Bootstrap's compiled CSS or JS, you can use jsDelivr. See it in action with our simple quick start, or browse the examples to jumpstart your next project. You can also choose to include Popper and our JS separately.

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist
```

```
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dis
```

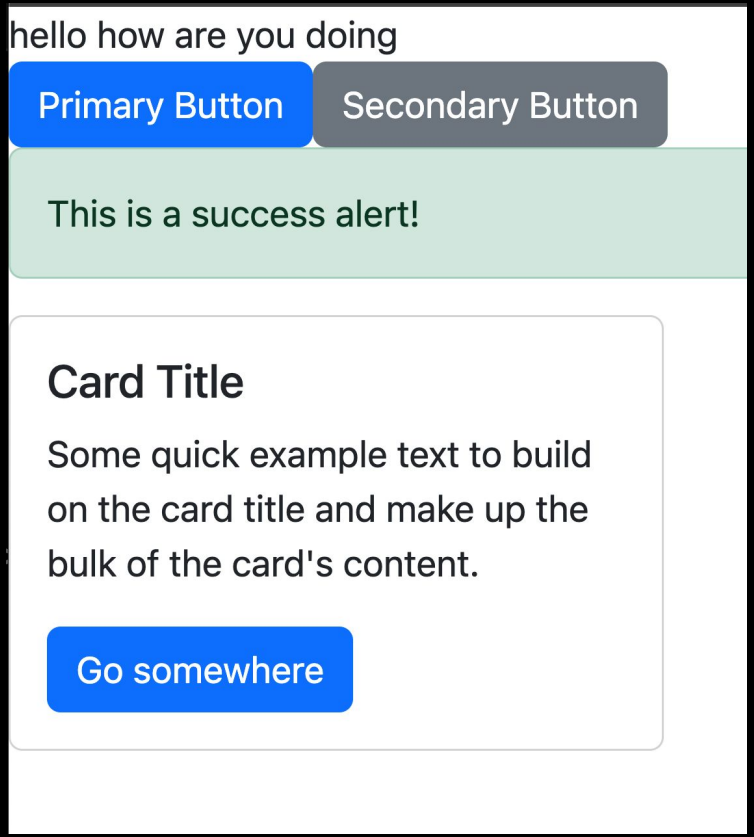

# Including Bootstrap with CRA

1. Install:
   npm i bootstrap@5.3.3
2. import
   import ‘bootstrap/dist/css/bootstrap.min.css’;

React

# Using Bootstrap

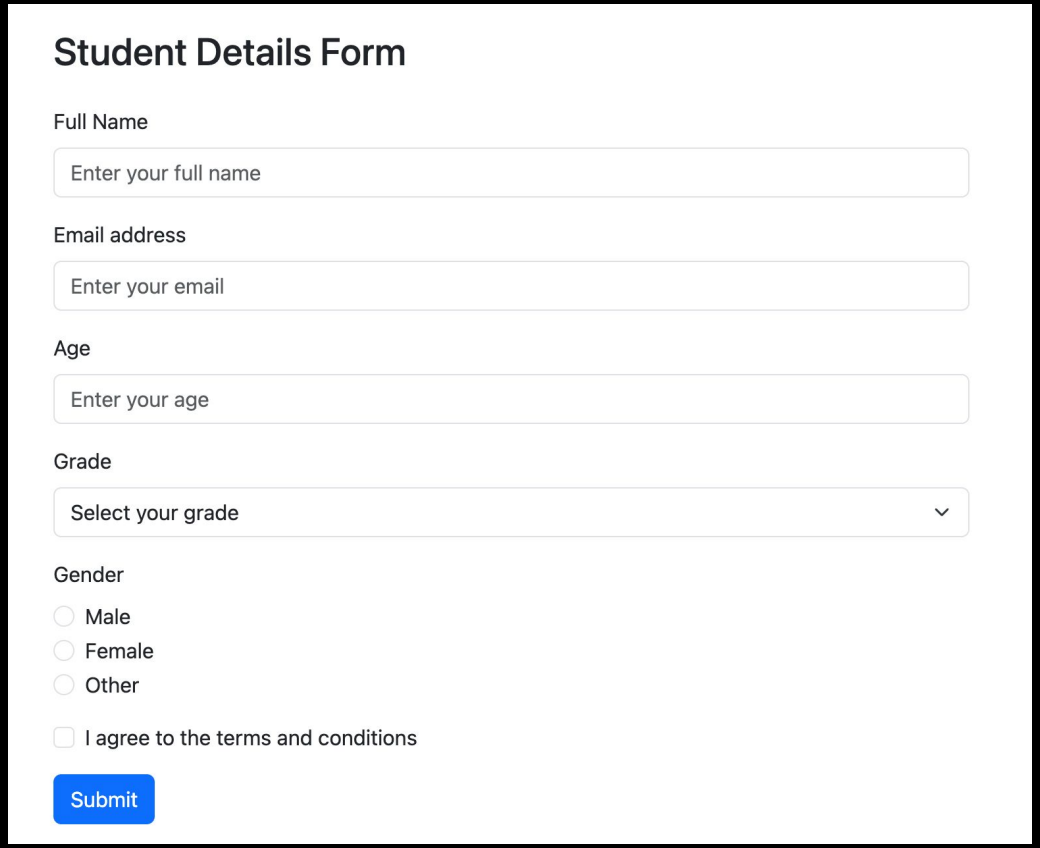
```jsx
function App() {
  return (
    <>
      <div>hello how are you doing</div>
      <button className="btn btn-primary">Primary Button</button>
      <button className="btn btn-secondary">Secondary Button</
      button>
      <div className="alert alert-success" role="alert">
        This is a success alert!
      </div>
      <div className="card" style={{ width: '18rem' }}>
        <div className="card-body">
          <h5 className="card-title">Card Title</h5>
          <p className="card-text">Some quick example text to
          build on the card title and make up the bulk of the
          card's content.</p>
          <a href="#" className="btn btn-primary">Go somewhere</a>
        </div>
      </div>
    </>
  )
}
```



# Project: TODO App UI

# Exercise: Create Form using Bootstrap

React

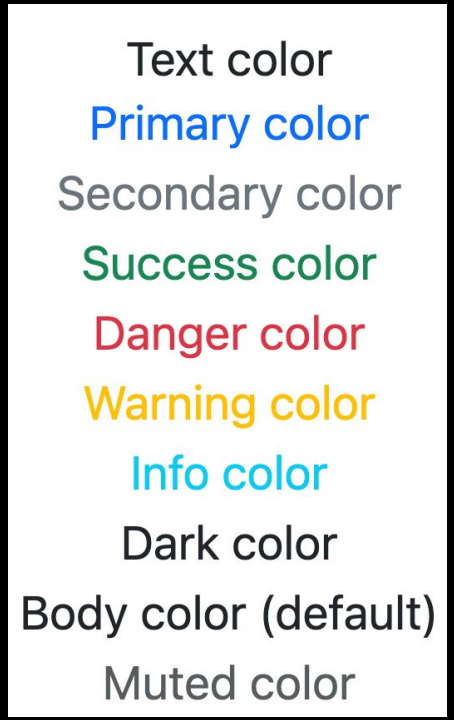



# Using Bootstrap CSS

1. Text color
2. Background color
3. Margin
4. Padding
5. Font size
6. Font weight
7. Width
8. Height
9. Borders
10. Text alignment
11. Text decoration
12. Text transform
13. Opacity
14. Display
15. Line height
16. List-Group
17. Position
18. Shadow
19. Tables
20. FlexBox
21. Grid
22. Badges
23. Alerts
24. Spinner
25. Progress-Bar
26. Drop-Down
27. Card
28. Form and Form-Control

# 1. Using TextColor

```
<div>
  <h2>Text color</h2>
  <p className="text-primary">Primary color</p>
  <p className="text-secondary">Secondary color</p>
  <p className="text-success">Success color</p>
  <p className="text-danger">Danger color</p>
  <p className="text-warning">Warning color</p>
  <p className="text-info">Info color</p>
  <p className="text-dark">Dark color</p>
  <p className="text-body">Body color (default)</p>
  <p className="text-muted">Muted color</p>
</div>
```

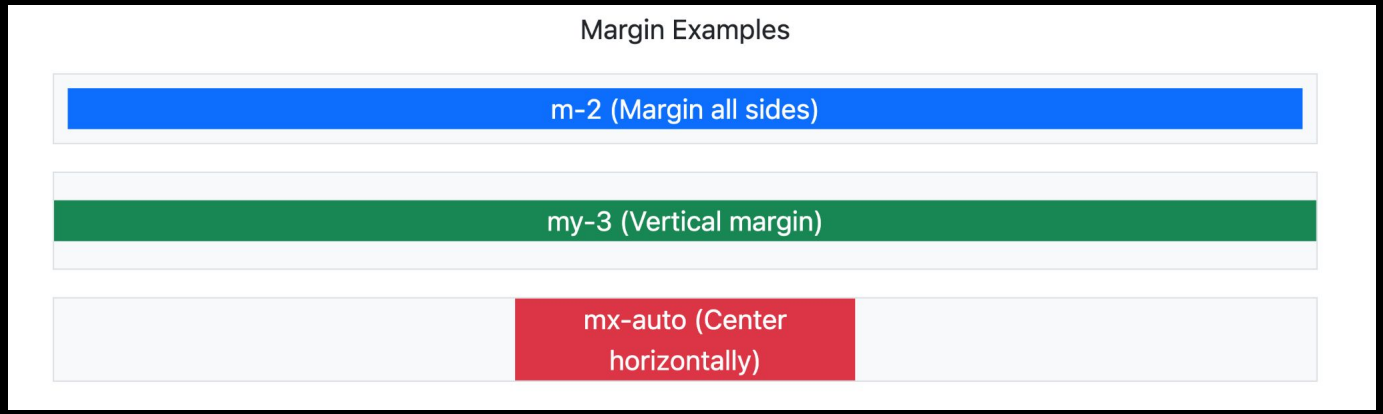


# 2. Using Background Color

```jsx
<div className="bg-primary">.bg-primary</div>
<div className="bg-primary-subtle">.bg-primary-subtle</div>
<div className="bg-secondary">.bg-secondary</div>
<div className="bg-secondary-subtle">.bg-secondary-subtle</div>
<div className="bg-success">.bg-success</div>
<div className="bg-success-subtle">.bg-success-subtle</div>
<div className="bg-danger">.bg-danger</div>
<div className="bg-danger-subtle">.bg-danger-subtle</div>
<div className="bg-warning">.bg-warning</div>
<div className="bg-warning-subtle">.bg-warning-subtle</div>
<div className="bg-info">.bg-info</div>
<div className="bg-info-subtle">.bg-info-subtle</div>
<div className="bg-light">.bg-light</div>
<div className="bg-light-subtle">.bg-light-subtle</div>
<div className="bg-dark">.bg-dark</div>
<div className="bg-dark-subtle">.bg-dark-subtle</div>
<div className="bg-body-secondary">.bg-body-secondary</div>
<div className="bg-body-tertiary">.bg-body-tertiary</div>
<div className="bg-body">.bg-body</div>
<div className="bg-black">.bg-black</div>
<div className="bg-white">.bg-white</div>
```

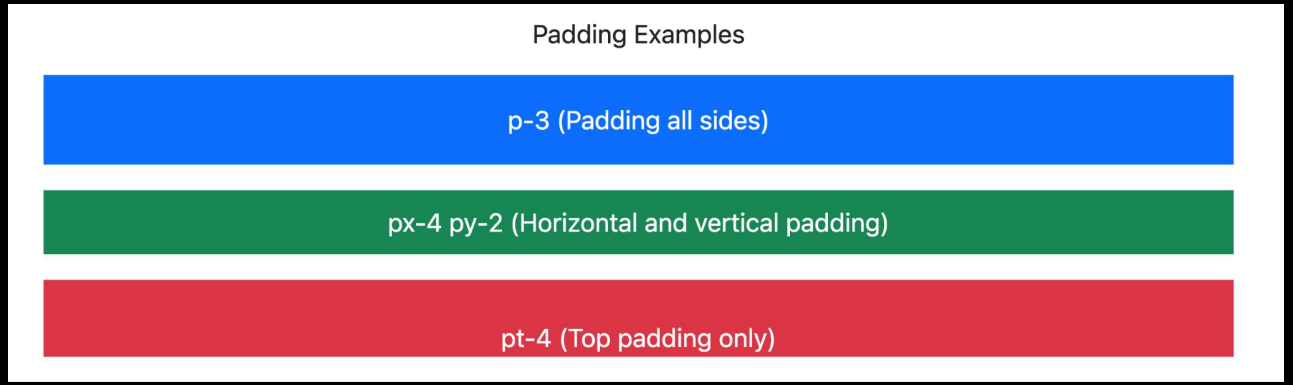
.bg-primary
.bg-primary-subtle
.bg-secondary
.bg-secondary-subtle
.bg-success
.bg-success-subtle
.bg-danger
.bg-danger-subtle
.bg-warning
.bg-warning-subtle
.bg-info
.bg-info-subtle
.bg-light
.bg-light-subtle
.bg-dark
.bg-dark-subtle
.bg-body-secondary
.bg-body-tertiary
.bg-body
.bg-black
.bg-white

# 3. Using Margin




```
<h2 className="mb-3">Margin Examples</h2>
<div className="bg-light border">
  <div className="bg-primary text-white m-2">m-2 (Margin all sides)</div>
</div>
<div className="bg-light border mt-3">
  <div className="bg-success text-white my-3">my-3 (Vertical margin)</div>
</div>
<div className="bg-light border mt-3">
  <div className="bg-danger text-white mx-auto" style={{width: '200px'}}>mx-auto (Center horizontally)</div>
</div>
```

# 4. Using Padding




```
<h2 className="mb-3">Padding Examples</h2>
<div className="bg-primary text-white p-3 mb-3">p-3 (Padding all sides)</div>
<div className="bg-success text-white px-4 py-2 mb-3">px-4 py-2 (Horizontal and vertical padding)</div>
<div className="bg-danger text-white pt-4 mb-3">pt-4 (Top padding only)</div>
```

# 5. Using Font-Size


```
<div className="container mt-4">
  <h1 className="mb-4">Bootstrap Font Size Examples</h1>
  <p className="fs-1">fs-1: This is the largest font size (2.5rem / 40px)</p>
  <p className="fs-2">fs-2: This is the second largest font size (2rem / 32px)</p>
  <p className="fs-3">fs-3: This is the third largest font size (1.75rem / 28px)</p>
  <p className="fs-4">fs-4: This is the fourth largest font size (1.5rem / 24px)</p>
  <p className="fs-5">fs-5: This is the fifth largest font size (1.25rem / 20px)</p>
  <p className="fs-6">fs-6: This is the smallest predefined font size (1rem / 16px)</p>
</div>
```

# 6. Using Font-Weight

Bootstrap Font Weight Examples

**fw-bold: This text is bold (font-weight: 700)**

**fw-bolder: This text is bolder relative to its parent**

fw-normal: This text has normal weight (font-weight: 400)

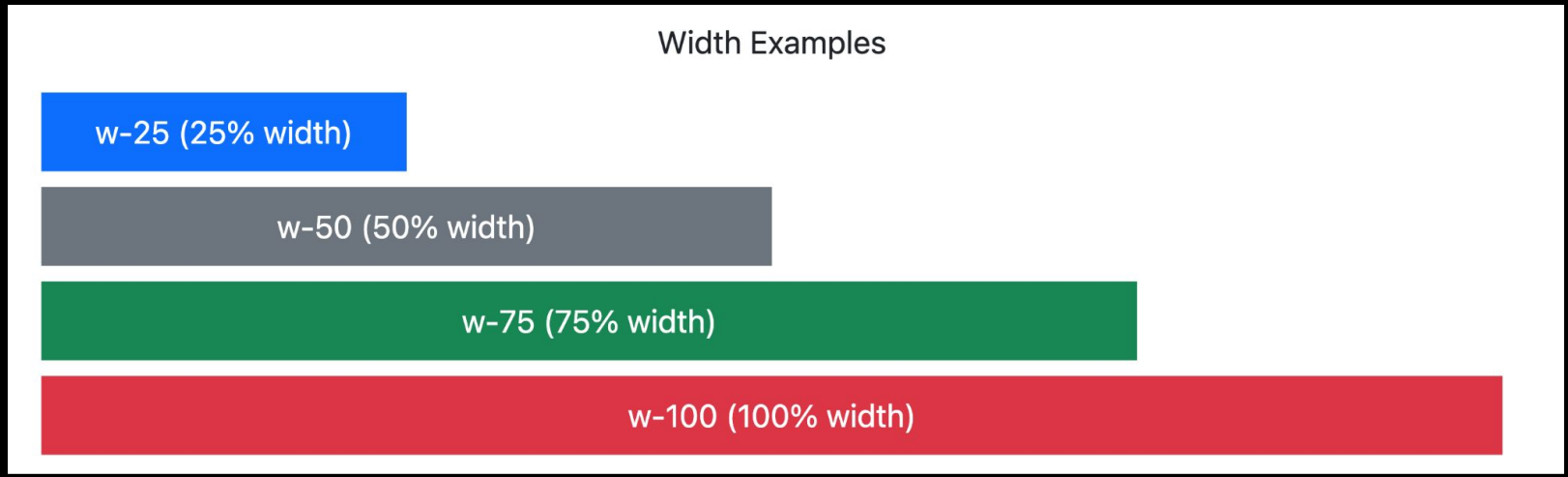
fw-light: This text is light weight (font-weight: 300)

fw-lighter: This text is lighter relative to its parent

*fst-italic: This text is italicized*

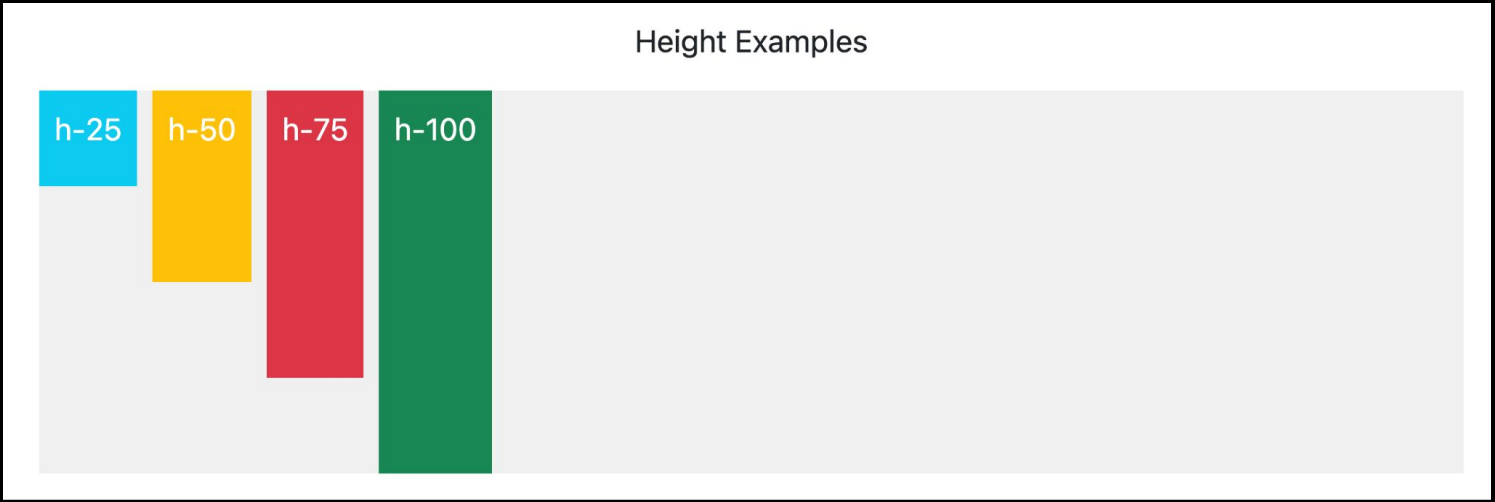
```
<h1 className="mb-4">Bootstrap Font Weight Examples</h1>

<p className="fw-bold">fw-bold: This text is bold (font-weight: 700)</p>
<p className="fw-bolder">fw-bolder: This text is bolder relative to its parent</p>
<p className="fw-normal">fw-normal: This text has normal weight (font-weight: 400)</p>
<p className="fw-light">fw-light: This text is light weight (font-weight: 300)</p>
<p className="fw-lighter">fw-lighter: This text is lighter relative to its parent</p>
<p className="fst-italic">fst-italic: This text is italicized</p>
```

# 7. Using Width




```
<h2 className="mb-3">Width Examples</h2>
<div className="mb-4">
  <div className="bg-primary text-white p-2 w-25 mb-2">w-25 (25% width)</div>
  <div className="bg-secondary text-white p-2 w-50 mb-2">w-50 (50% width)</div>
  <div className="bg-success text-white p-2 w-75 mb-2">w-75 (75% width)</div>
  <div className="bg-danger text-white p-2 w-100">w-100 (100% width)</div>
</div>
```

# 8. Using Height

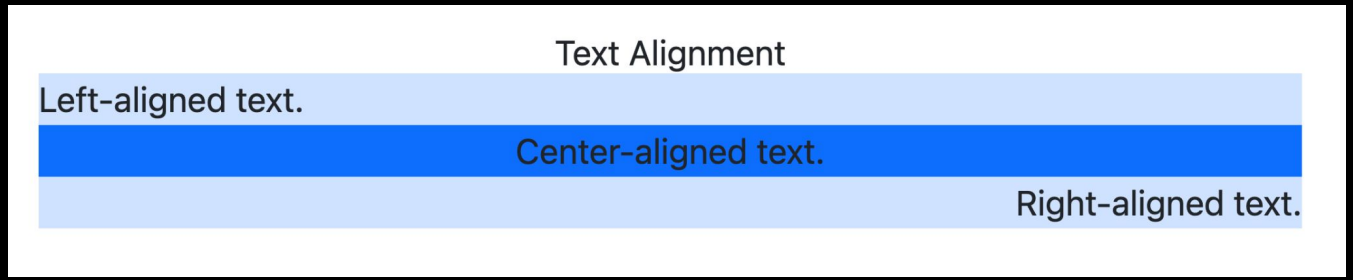



```
<h2 className="mb-3">Height Examples</h2>
<div className="mb-4 d-flex" style={{height: '200px', background: '#f0f0f0'}}>
  <div className="bg-info text-white p-2 h-25 me-2">h-25</div>
  <div className="bg-warning text-white p-2 h-50 me-2">h-50</div>
  <div className="bg-danger text-white p-2 h-75 me-2">h-75</div>
  <div className="bg-success text-white p-2 h-100">h-100</div>
</div>
```

# 9. Using Border

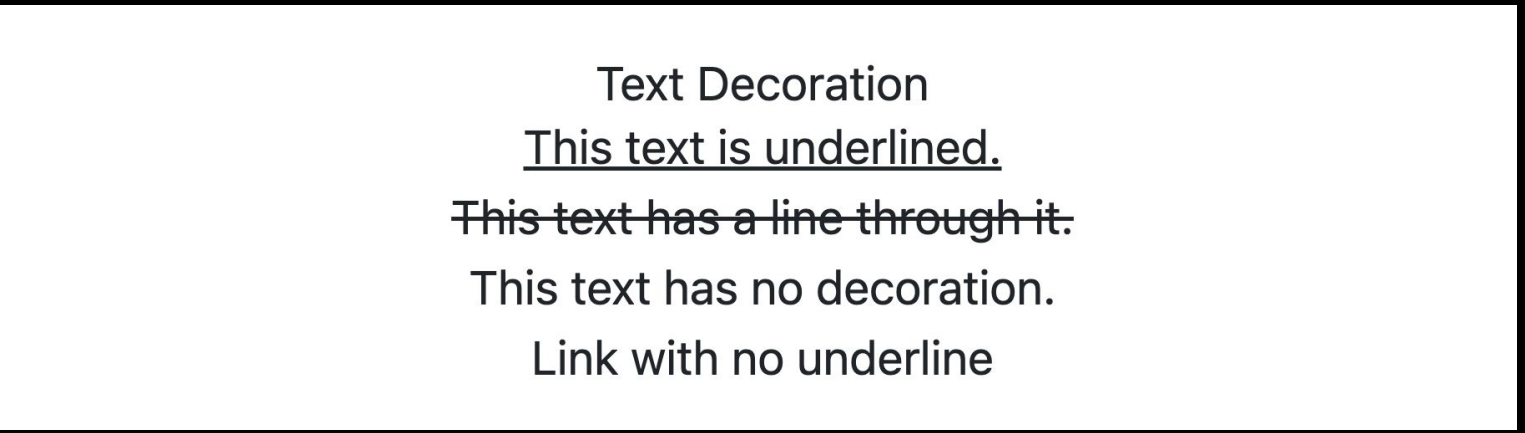

```
<h1 className="mb-4">Bootstrap Border Examples</h1>

<div className="d-flex flex-wrap gap-3">
  <div className="p-3 border">Default</div>
  <div className="p-3 border border-primary">Primary</div>
  <div className="p-3 border-top">Top only</div>
  <div className="p-3 border border-2">Thick</div>
  <div className="p-3 border rounded">Rounded</div>
  <div className="p-3 border rounded-circle" style={{width: '100px', height: '100px'}}>Circle</div>
</div>
```

# 10 Using Text-Alignment



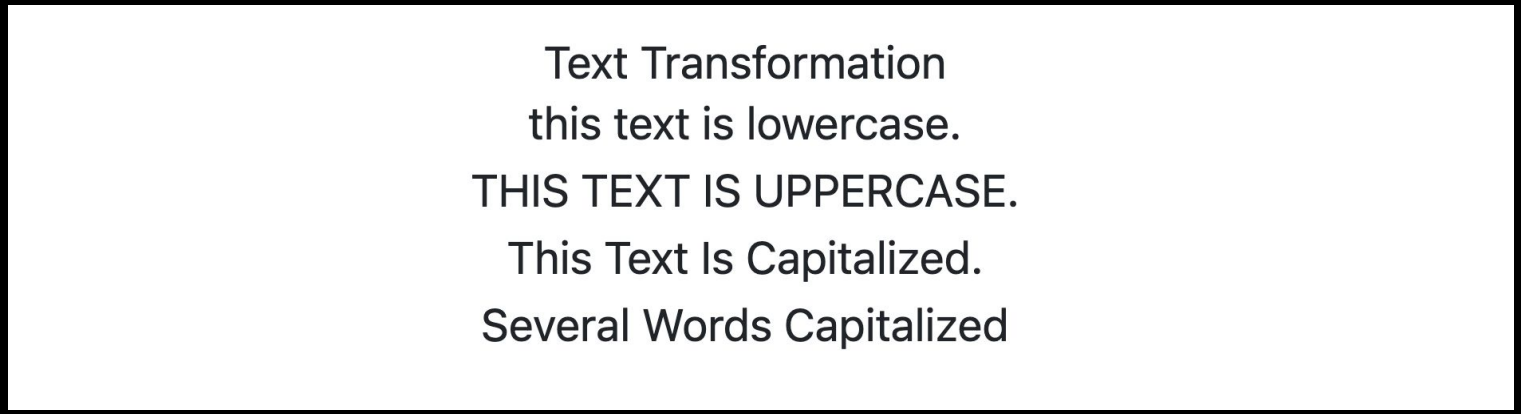

```
<h2>Text Alignment</h2>
<p className="text-start bg-primary-subtle">Left-aligned text.</p>
<p className="text-center bg-primary">Center-aligned text.</p>
<p className="text-end bg-primary-subtle">Right-aligned text.</p>
```

# 11. Using Text-Decoration




```
<h2>Text Decoration</h2>
<p className="text-decoration-underline">This text is underlined.</p>
<p className="text-decoration-line-through">This text has a line through it.</p>
<p className="text-decoration-none">This text has no decoration.</p>
<a href="#" className="text-decoration-none">Link with no underline</a>
```

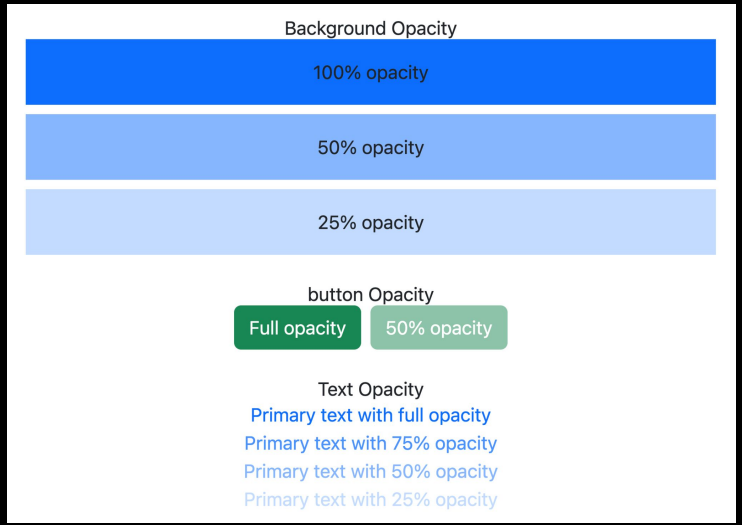
# 12. Using Text-Transform



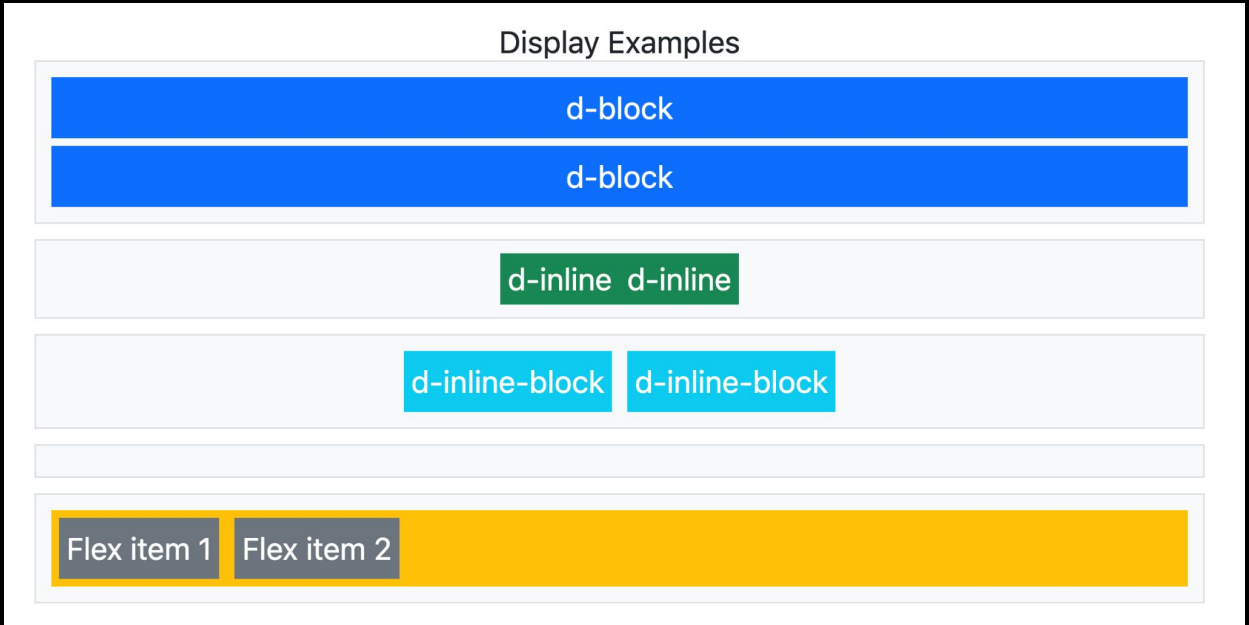

```
<h2>Text Transformation</h2>
<p className="text-lowercase">THIS TEXT IS LOWERCASE.</p>
<p className="text-uppercase">This text is uppercase.</p>
<p className="text-capitalize">this text is capitalized.</p>
<p className="text-capitalize">several words capitalized</p>
```

# 13. Using Opacity

```
<h2 className="mt-4">Background Opacity</h2>
<div className="p-3 mb-2 bg-primary bg-opacity-100">100% opacity</div>
<div className="p-3 mb-2 bg-primary bg-opacity-50">50% opacity</div>
<div className="p-3 bg-primary bg-opacity-25">25% opacity</div>

<h2 className="mt-4">button Opacity</h2>
<button className="btn btn-success me-2">Full opacity</button>
<button className="btn btn-success opacity-50">50% opacity</button>

<h2 className="mt-4">Text Opacity</h2>
<p className="text-primary">Primary text with full opacity</p>
<p className="text-primary text-opacity-75">Primary text with 75% opacity</p>
<p className="text-primary text-opacity-50">Primary text with 50% opacity</p>
<p className="text-primary text-opacity-25">Primary text with 25% opacity</p>
```



# 14. Using Display



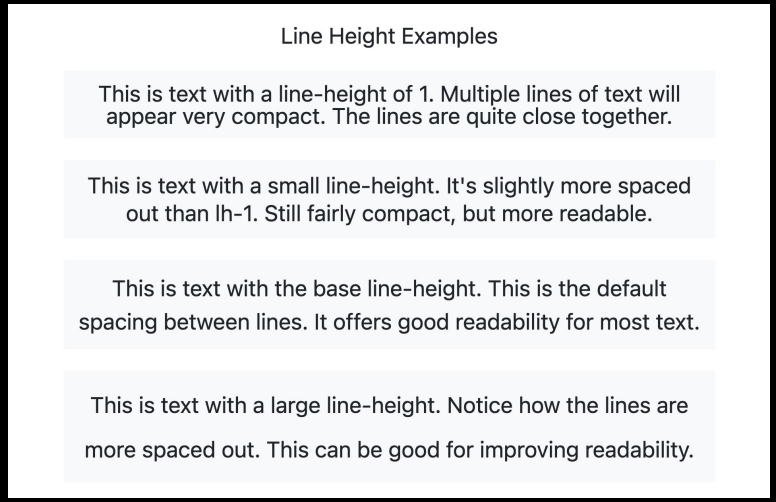

`d-block`: Makes an element a block-level element.
`d-inline`: Makes an element an inline element.
`d-inline-block`: Combines characteristics of both block and inline.
`d-none`: Hides an element.
`d-flex`: Creates a flexbox container.


# 14. Using Display

```
<div className="bg-light border p-2 mb-2">
  <span className="d-block bg-primary text-white p-1 mb-1">d-block</span>
  <span className="d-block bg-primary text-white p-1">d-block</span>
</div>

<div className="bg-light border p-2 mb-2">
  <span className="d-inline bg-success text-white p-1">d-inline</span>
  <span className="d-inline bg-success text-white p-1">d-inline</span>
</div>

<div className="bg-light border p-2 mb-2">
  <div className="d-inline-block bg-info text-white p-1 me-2">d-inline-block</div>
  <div className="d-inline-block bg-info text-white p-1">d-inline-block</div>
</div>

<div className="bg-light border p-2 mb-2">
  <div className="d-none bg-danger text-white p-1">d-none (hidden)</div>
</div>

<div className="bg-light border p-2">
  <div className="d-flex bg-warning p-1">
    <div className="p-1 bg-secondary text-white">Flex item 1</div>
    <div className="p-1 bg-secondary text-white ms-2">Flex item 2</div>
  </div>
</div>
```

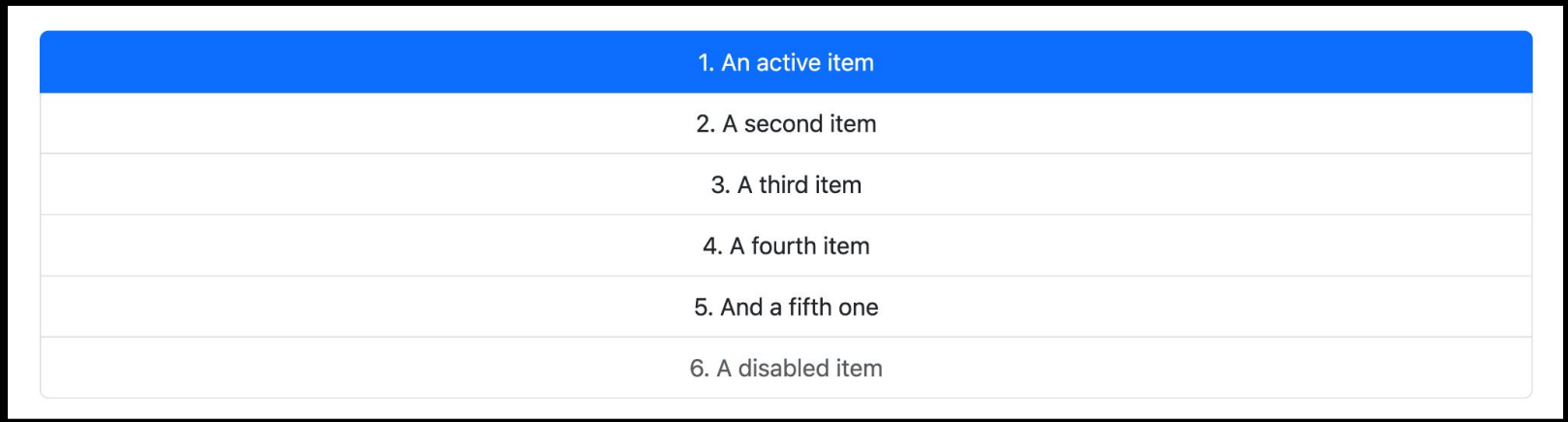
# 15. Using Line-Height

```
<div className="container mt-3">
  <h2 className="mb-3">Line Height Examples</h2>

  <p className="lh-1 bg-light p-2 mb-3">
    This is text with a line-height of 1.
    Multiple lines of text will appear very compact.
    The lines are quite close together.
  </p>

  <p className="lh-sm bg-light p-2 mb-3">
    This is text with a small line-height.
    It's slightly more spaced out than lh-1.
    Still fairly compact, but more readable.
  </p>

  <p className="lh-base bg-light p-2 mb-3">
    This is text with the base line-height.
    This is the default spacing between lines.
    It offers good readability for most text.
  </p>

  <p className="lh-lg bg-light p-2">
    This is text with a large line-height.
    Notice how the lines are more spaced out.
    This can be good for improving readability.
  </p>
</div>
```



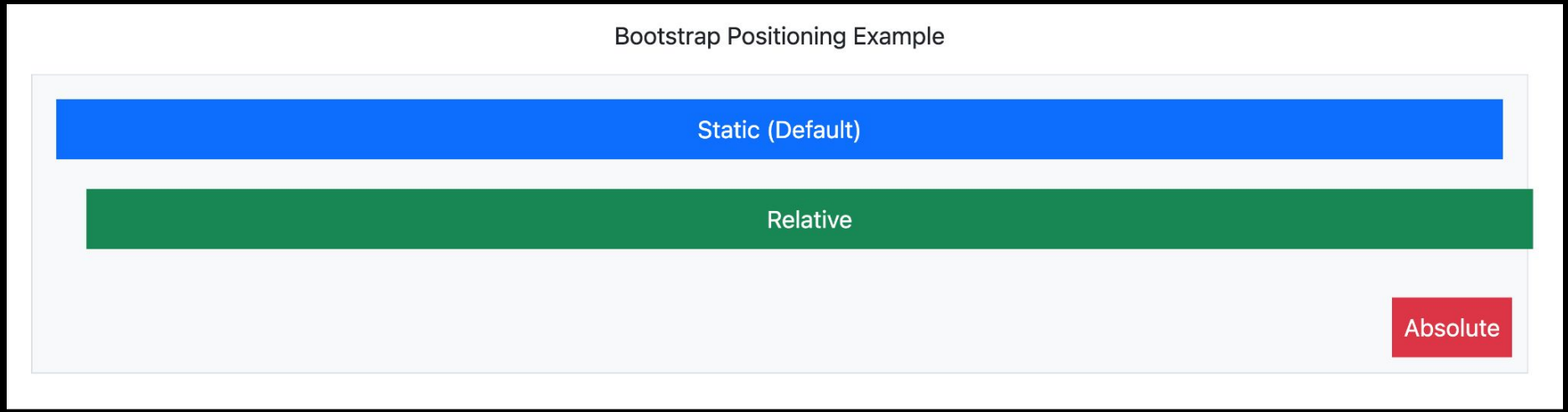

- lh-1 creates very compact text, which might be useful for headings or where space is at a premium.
- lh-sm offers slightly more breathing room than lh-1, but is still relatively compact.
- lh-base is the default and generally provides good readability for body text.
- lh-lg creates more spacious text, which can be helpful for improving readability, especially for longer passages of text.

# 16. Using List-Group



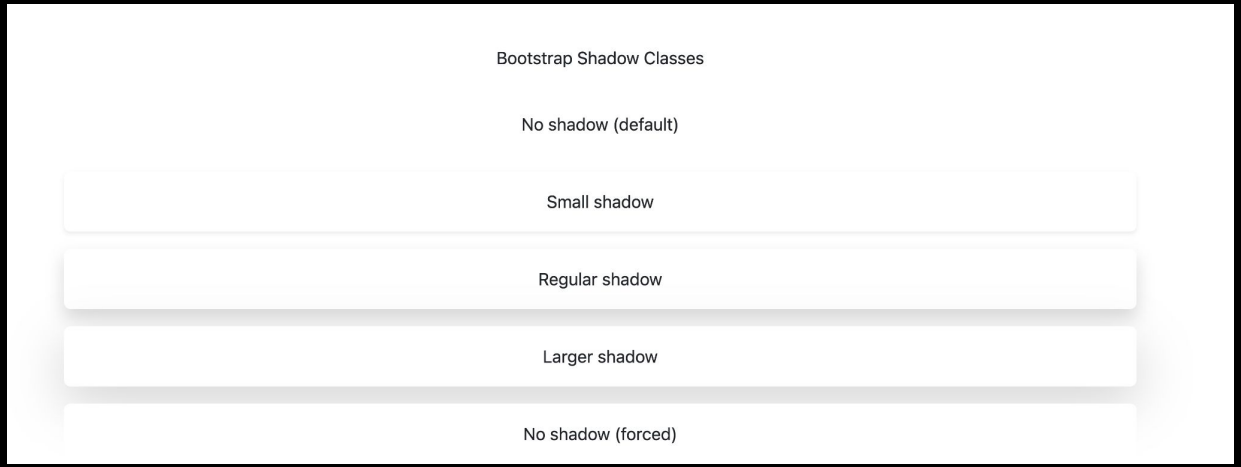

```
<ul class="list-group list-group-numbered">
  <li class="list-group-item active" aria-current="true">An active item</li>
  <li class="list-group-item">A second item</li>
  <li class="list-group-item">A third item</li>
  <li class="list-group-item">A fourth item</li>
  <li class="list-group-item">And a fifth one</li>
  <li class="list-group-item disabled" aria-disabled="true">A disabled item</li>
</ul>
```

# 17. Using Position




```
<h2 className="mb-3">Bootstrap Positioning Example</h2>
<div className="position-relative bg-light border p-3" style={{ height: '200px' }}>
  <div className="p-2 bg-primary text-white"> Static (Default) </div>
  <div className="position-relative p-2 bg-success text-white" style={{ top: '20px', left: '20px' }}> Relative </div>
  <div className="position-absolute p-2 bg-danger text-white" style={{ bottom: '10px', right: '10px' }}> Absolute </div>
</div>
```

# 18. Using Shadow




```
<h2 className="mb-4">Bootstrap Shadow Classes</h2>

<div className="p-3 mb-3 bg-white rounded">No shadow (default)</div>
<div className="p-3 mb-3 bg-white rounded shadow-sm">Small shadow</div>
<div className="p-3 mb-3 bg-white rounded shadow">Regular shadow</div>
<div className="p-3 mb-3 bg-white rounded shadow-lg">Larger shadow</div>
<div className="p-3 mb-3 bg-white rounded shadow-none">No shadow (forced)</div>
```

# 19. Using Tables

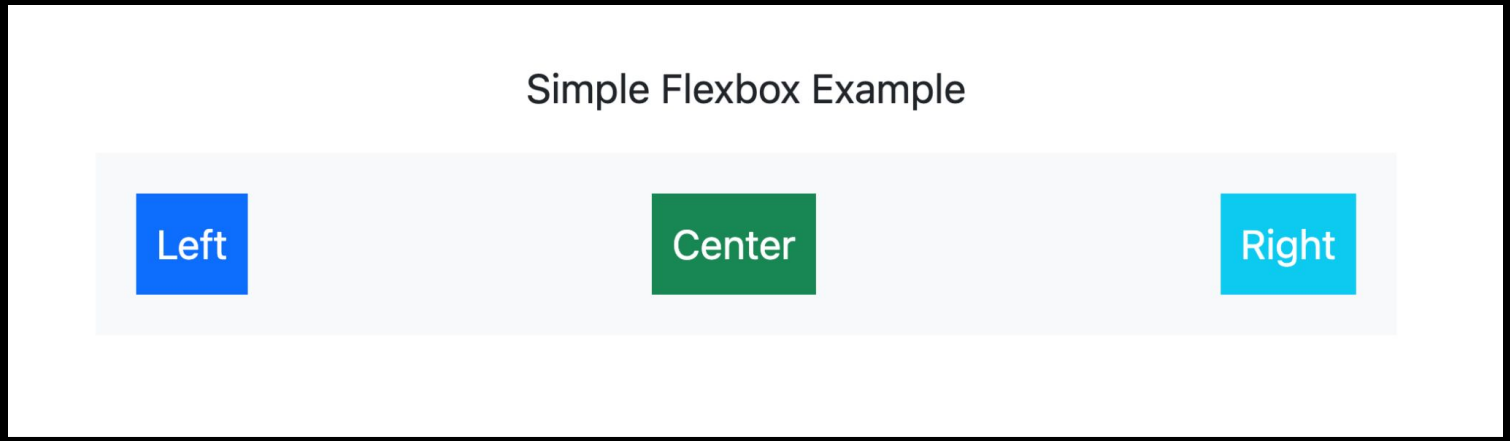
React

```
<h2 className="mb-4">Simple Bootstrap Table</h2>

<table className="table table-striped table-hover">
  <thead className="table-dark">
    <tr>
      <th scope="col">#</th>
      <th scope="col">First Name</th>
      <th scope="col">Last Name</th>
      <th scope="col">Email</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th scope="row">1</th>
      <td>John</td>
      <td>Doe</td>
      <td>john@example.com</td>
    </tr>
    <tr>
      <th scope="row">2</th>
      <td>Jane</td>
      <td>Smith</td>
      <td>jane@example.com</td>
    </tr>
  </tbody>
</table>
```

Simple Bootstrap Table

| # | First Name | Last Name | Email |
|---|---|---|---|
| 1 | John | Doe | john@example.com |
| 2 | Jane | Smith | jane@example.com |

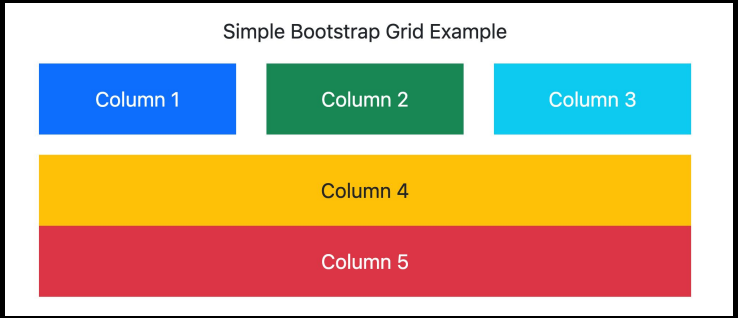
# 20. Using FlexBox




```
<div className="container mt-4">
  <h3 className="mb-3">Simple Flexbox Example</h3>

  <div className="d-flex justify-content-between align-items-center bg-light p-3">
    <div className="p-2 bg-primary text-white">Left</div>
    <div className="p-2 bg-success text-white">Center</div>
    <div className="p-2 bg-info text-white">Right</div>
  </div>
</div>
```

# 21. Using Grid

```
<h3 className="mb-3">Simple Bootstrap Grid Example</h3>

<div className="row">
  <div className="col-sm-4">
    <div className="p-3 bg-primary text-white">Column 1</div>
  </div>
  <div className="col-sm-4">
    <div className="p-3 bg-success text-white">Column 2</div>
  </div>
  <div className="col-sm-4">
    <div className="p-3 bg-info text-white">Column 3</div>
  </div>
</div>

<div className="row mt-3">
  <div className="col-md-6">
    <div className="p-3 bg-warning text-dark">Column 4</div>
  </div>
  <div className="col-md-6">
    <div className="p-3 bg-danger text-white">Column 5</div>
  </div>
</div>
```

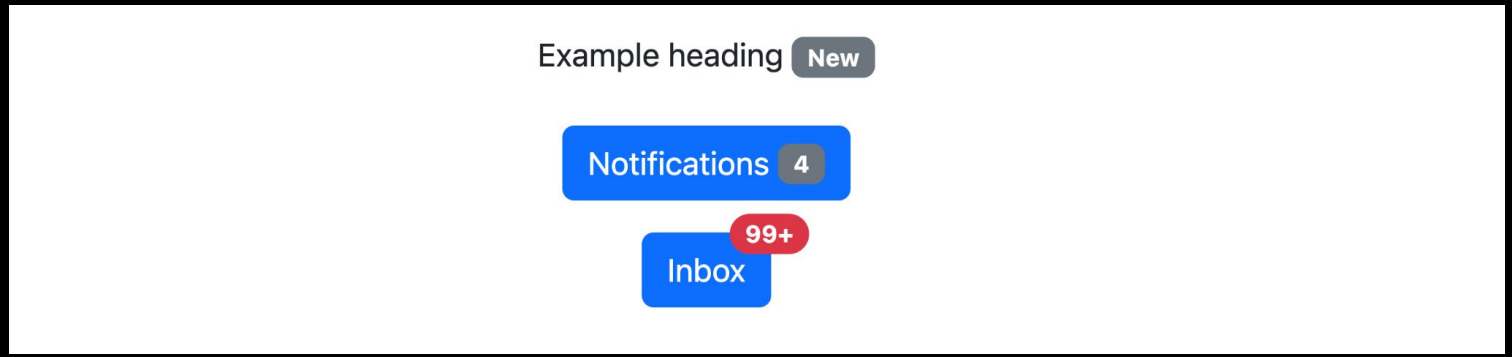



1. **Rows**: `row`: Creates a horizontal group of columns
2. **Columns**:
   - `col-sm-4`: Creates a column that spans 4 out of 12 grid columns on small screens and up
   - `col-md-6`: Creates a column that spans 6 out of 12 grid columns on medium screens and up

Key Bootstrap grid classes:

- `container`: Creates a responsive container
- `row`: Defines a row in the grid system
- `col-sm-4`: Defines a column that takes up 1/3 of the row width on small screens and up
- `col-md-6`: Defines a column that takes up 1/2 of the row width on medium screens and up

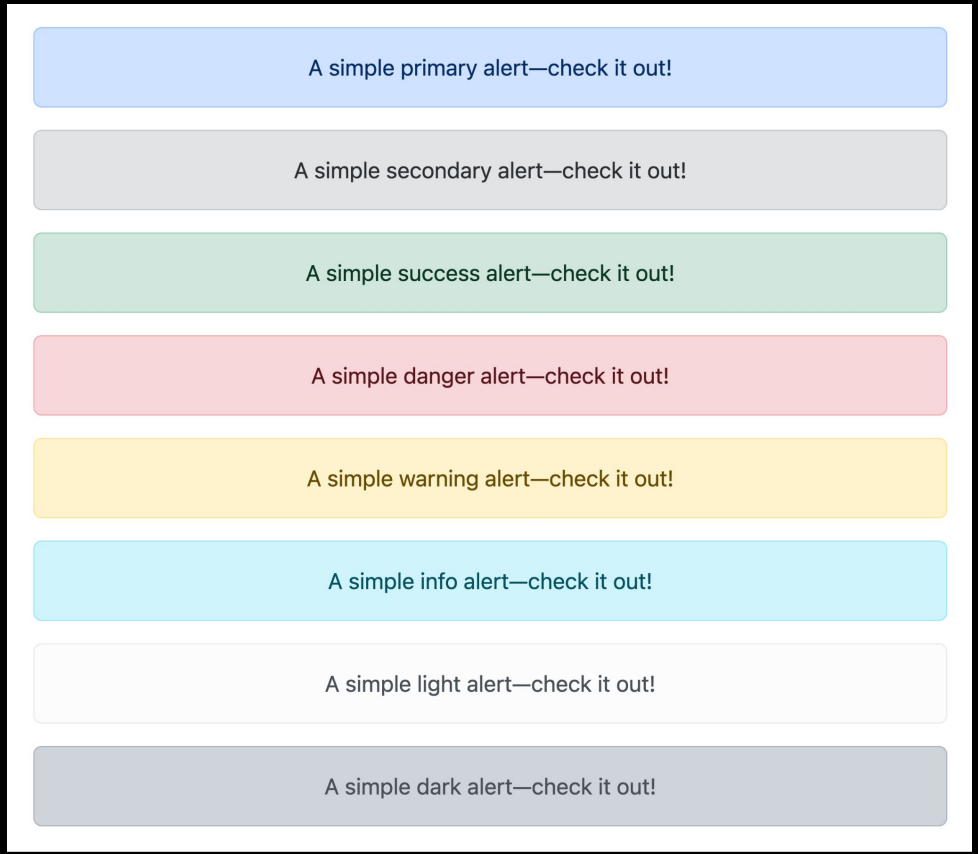
# 22. Using Badges




```
<h4>Example heading <span class="badge text-bg-secondary">New</span></h4>
<br />
<button type="button" class="btn btn-primary">
  Notifications <span class="badge text-bg-secondary">4</span>
</button><br />
<button type="button" class="mt-3 btn btn-primary position-relative">
  Inbox
  <span class="position-absolute top-0 start-100 translate-middle badge rounded-pill bg-danger">
    99+
    <span class="visually-hidden">unread messages</span>
  </span>
</button>
```

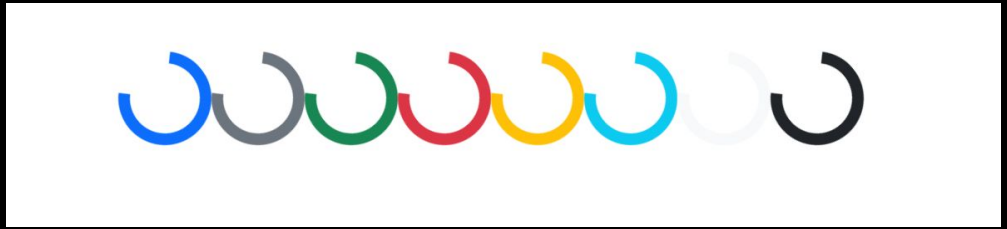
# 23. Using Alerts

```
<div class="alert alert-primary" role="alert">
  A simple primary alert—check it out!
</div>
<div class="alert alert-secondary" role="alert'
  A simple secondary alert—check it out!
</div>
<div class="alert alert-success" role="alert">
  A simple success alert—check it out!
</div>
<div class="alert alert-danger" role="alert">
  A simple danger alert—check it out!
</div>
<div class="alert alert-warning" role="alert">
  A simple warning alert—check it out!
</div>
<div class="alert alert-info" role="alert">
  A simple info alert—check it out!
</div>
<div class="alert alert-light" role="alert">
  A simple light alert—check it out!
</div>
<div class="alert alert-dark" role="alert">
  A simple dark alert—check it out!
</div>
```

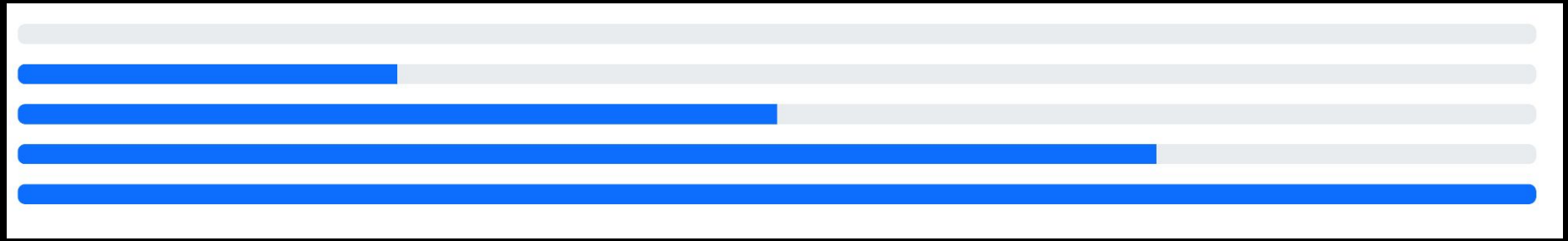


# 24. Using Spinner

```
<div class="spinner-border text-primary" role="status">
  <span class="visually-hidden">Loading...</span>
</div>
<div class="spinner-border text-secondary" role="status">
  <span class="visually-hidden">Loading...</span>
</div>
<div class="spinner-border text-success" role="status">
  <span class="visually-hidden">Loading...</span>
</div>
<div class="spinner-border text-danger" role="status">
  <span class="visually-hidden">Loading...</span>
</div>
<div class="spinner-border text-warning" role="status">
  <span class="visually-hidden">Loading...</span>
</div>
<div class="spinner-border text-info" role="status">
  <span class="visually-hidden">Loading...</span>
</div>
<div class="spinner-border text-light" role="status">
  <span class="visually-hidden">Loading...</span>
</div>
<div class="spinner-border text-dark" role="status">
  <span class="visually-hidden">Loading...</span>
</div>
```

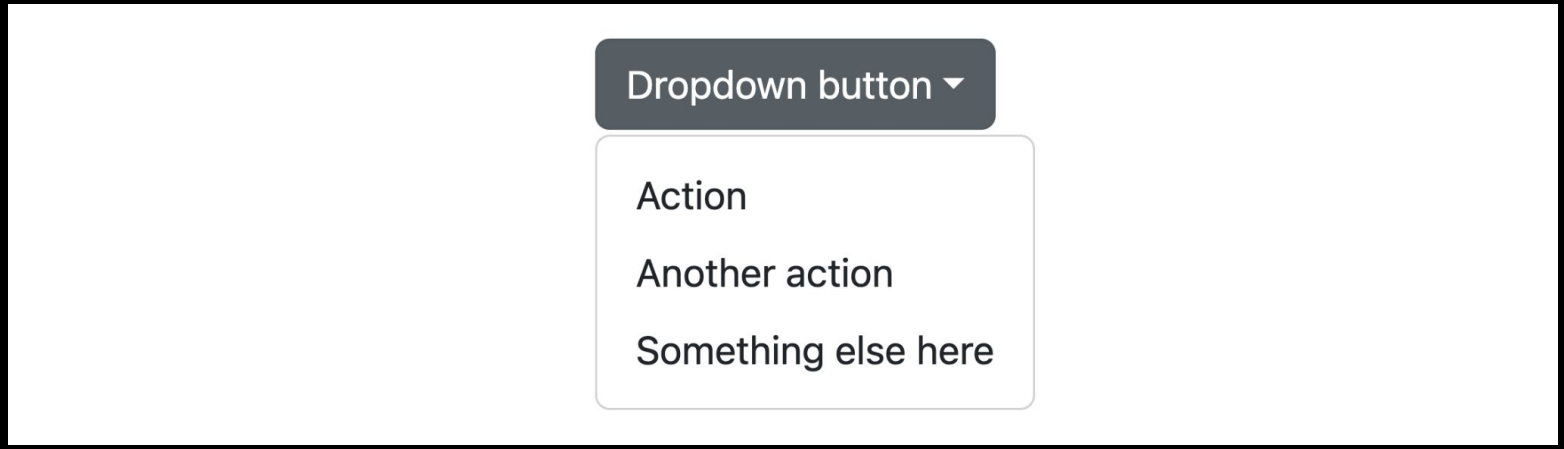


# 25. Using Progress-Bar



```
<div className="progress mt-3" role="progressbar" aria-label="Basic example" aria-valuenow="0" aria-valuemin="0" aria-valuemax="100">
  <div className="progress-bar" style={{ width: "0%" }}></div>
</div>
<div className="progress mt-3" role="progressbar" aria-label="Basic example" aria-valuenow="25" aria-valuemin="0" aria-valuemax="100">
  <div className="progress-bar" style={{ width: "25%" }}></div>
</div>
<div className="progress mt-3" role="progressbar" aria-label="Basic example" aria-valuenow="50" aria-valuemin="0" aria-valuemax="100">
  <div className="progress-bar" style={{ width: "50%" }}></div>
</div>
<div className="progress mt-3" role="progressbar" aria-label="Basic example" aria-valuenow="75" aria-valuemin="0" aria-valuemax="100">
  <div className="progress-bar" style={{ width: "75%" }}></div>
</div>
<div className="progress mt-3" role="progressbar" aria-label="Basic example" aria-valuenow="100" aria-valuemin="0" aria-valuemax="100">
  <div className="progress-bar" style={{ width: "100%" }}></div>
</div>
```

# 26. Using Drop-Down




```
<div class="dropdown">
  <button class="btn btn-secondary dropdown-toggle" type="button" data-bs-toggle="dropdown" aria-expanded="false">
    Dropdown button
  </button>
  <ul class="dropdown-menu">
    <li><a class="dropdown-item" href="#">Action</a></li>
    <li><a class="dropdown-item" href="#">Another action</a></li>
    <li><a class="dropdown-item" href="#">Something else here</a></li>
  </ul>
</div>
```

# 27. Using Card

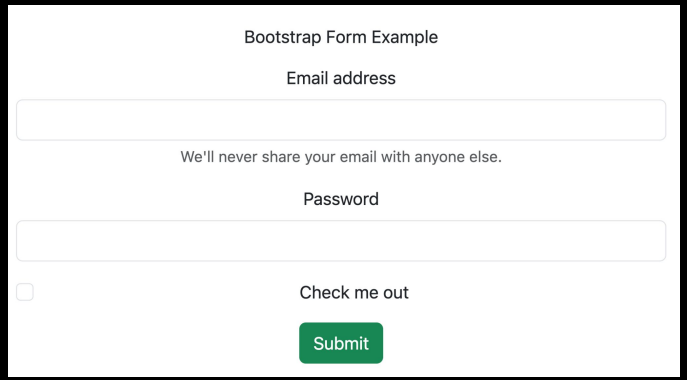
```
<div className="card" style={{ width: "18rem" }}>
  <img src="https://a0.muscache.com/im/pictures/
  1ff6d909-5ba6-42f3-9d2c-fa2327780936.jpg"
  className="card-img-top" alt="..." />
  <div className="card-body">
    <h5 className="card-title">Card title</h5>
    <p className="card-text">Some quick example text to
    build on the card title and make up the bulk of the
    card's content.</p>
    <a href="#" className="btn btn-primary">Go somewhere</a>
  </div>
</div>
```

Card title

Some quick example text to build on the card title and make up the bulk of the card's content.

Go somewhere

# 28. Using Form & Form-Control




The `form-control` class is a fundamental part of Bootstrap's form styling. It's applied to form input elements to give them a consistent, styled appearance across different browsers and devices.

```
<form>
  <div className="mb-3">
    <label htmlFor="exampleInputEmail1" className="form-label">Email address</label>
    <input type="email" className="form-control" id="exampleInputEmail1" aria-describedby="emailHelp" />
    <div id="emailHelp" className="form-text">We'll never share your email with anyone else.</div>
  </div>
  <div className="mb-3">
    <label htmlFor="exampleInputPassword1" className="form-label">Password</label>
    <input type="password" className="form-control" id="exampleInputPassword1" />
  </div>
  <div className="mb-3 form-check">
    <input type="checkbox" className="form-check-input" id="exampleCheck1" />
    <label className="form-check-label" htmlFor="exampleCheck1">Check me out</label>
  </div>
  <button type="submit" className="btn btn-success">Submit</button>
</form>
```


React

# Exercise: Create Form using Bootstrap


## Student Details Form

Full Name

Enter your full name

Email address

Enter your email

Age

Enter your age

Grade

Select your grade

Gender

- Male
- Female
- Other

I agree to the terms and conditions

Submit


# Bootstrap: Classes to be used

`container`

- Creates a responsive fixed-width container
- Centers the content and provides padding on the sides
-

`form-label`

- Styles form labels
- Typically adds some bottom margin and makes the text bold

`form-control`

- Styles form inputs and textareas
- Provides full-width, rounded corners, and consistent padding
- Adds focus styles and hover effects

`form-select`

- Styles select dropdowns
- Similar to form-control but with a custom dropdown arrow

`form-check`

- Container class for custom radio buttons and checkboxes
- Provides proper spacing and alignment

`form-check-input`

- Styles the actual input for radio buttons and checkboxes
- Works with the form-check class to create custom-styled inputs

`form-check-label`

- Styles labels for radio buttons and checkboxes
- Provides proper alignment with the custom inputs

```jsx
import React from 'react';

const StudentDetailsForm = () => {
  return (
    <div className="container mt-5">
      <h2 className="mb-4">Student Details Form</h2>
      <form>
        <div className="mb-3">
          <label htmlFor="fullName" className="form-label">Full Name</label>
          <input type="text" className="form-control" id="fullName" placeholder="Enter your full name" />
        </div>
        <div className="mb-3">
          <label htmlFor="email" className="form-label">Email address</label>
          <input type="email" className="form-control" id="email" placeholder="Enter your email" />
        </div>
        <div className="mb-3">
          <label htmlFor="age" className="form-label">Age</label>
          <input type="number" className="form-control" id="age" placeholder="Enter your age" />
        </div>
        <div className="mb-3">
          <label htmlFor="grade" className="form-label">Grade</label>
          <select className="form-select" id="grade">
            <option selected>Select your grade</option>
            <option value="9">9th Grade</option>
            <option value="10">10th Grade</option>
            <option value="11">11th Grade</option>
            <option value="12">12th Grade</option>
          </select>
        </div>
        <div className="mb-3">
          <label className="form-label">Gender</label>
```

```
32            <div className="form-check">
33              <input className="form-check-input" type="radio" name="gender" id="male" />
34              <label className="form-check-label" htmlFor="male">
35                Male
36              </label>
37            </div>
38            <div className="form-check">
39              <input className="form-check-input" type="radio" name="gender" id="female" />
40              <label className="form-check-label" htmlFor="female">
41                Female
42              </label>
43            </div>
44            <div className="form-check">
45              <input className="form-check-input" type="radio" name="gender" id="other" />
46              <label className="form-check-label" htmlFor="other">
47                Other
48              </label>
49            </div>
50          </div>
51          <div className="mb-3 form-check">
52            <input type="checkbox" className="form-check-input" id="termsCheck" />
53            <label className="form-check-label" htmlFor="termsCheck">I agree to the terms and conditions</label>
54          </div>
55          <button type="submit" className="btn btn-primary">Submit</button>
56        </form>
57      </div>
58    );
59  };
60
61  export default StudentDetailsForm;
```


React

# Installing Extension

## Bootstrap 4, Font awesome 4, Font Aweso

Ashok Koyi | 3,257,489 | ★★★★☆ (28)

Bootstrap 4 snippets based on documentation + Font awesome 4 + Font A...

Install | Auto Update

DETAILS FEATURES CHANGELOG

### Bootstrap 4, Font awesome 4, Font Awesome 5 Free & Pro snippets for Visual studio code

Visual studio code plugin containing Bootstrap 4, Font awesome 4 & Font Awesome 5 Free & Pro snippets. This plugin works in both in the stable & the insiders build

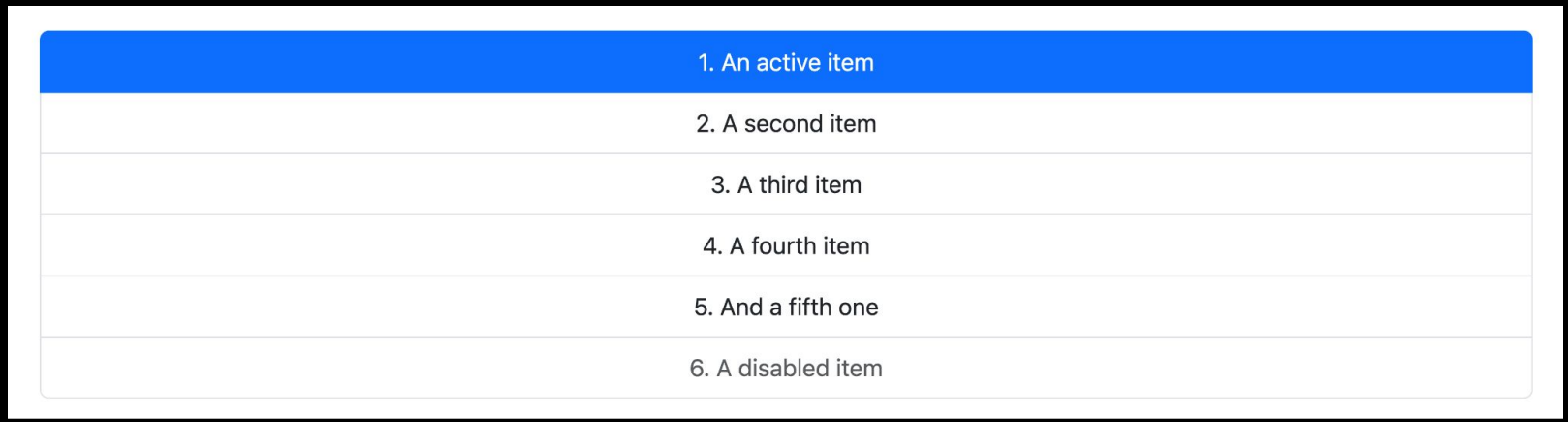
Categories

Snippets

Resources

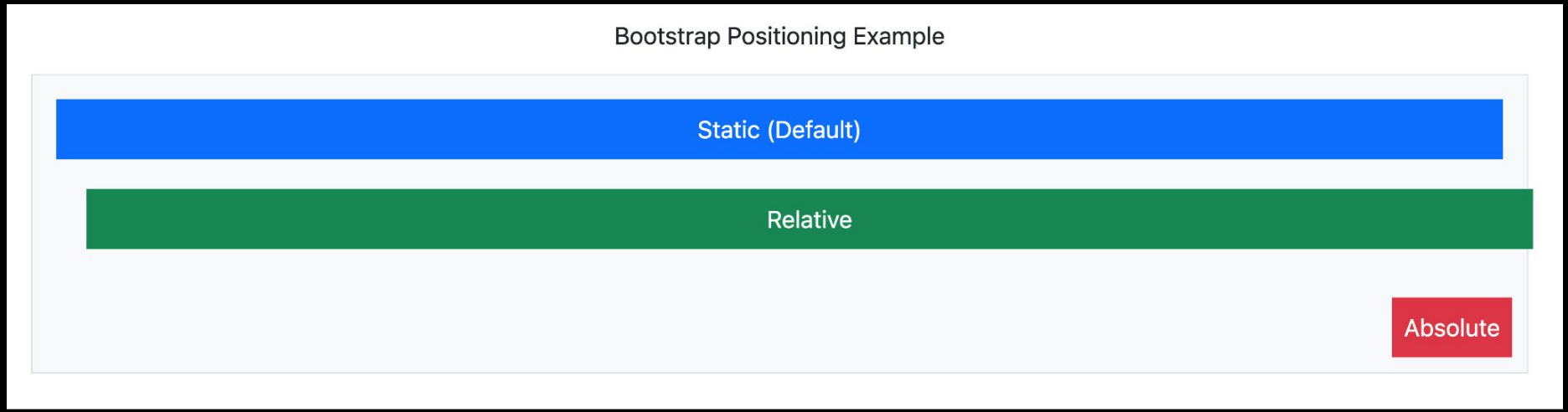
Marketplace
Repository
Ashok Koyi

# 16. Using List-Group



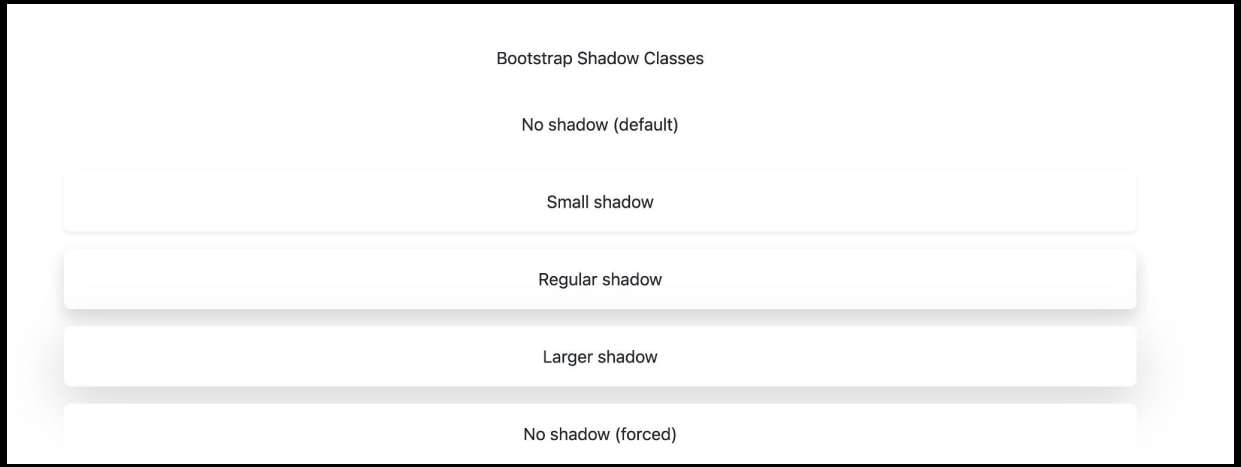

```
<ul class="list-group list-group-numbered">
  <li class="list-group-item active" aria-current="true">An active item</li>
  <li class="list-group-item">A second item</li>
  <li class="list-group-item">A third item</li>
  <li class="list-group-item">A fourth item</li>
  <li class="list-group-item">And a fifth one</li>
  <li class="list-group-item disabled" aria-disabled="true">A disabled item</li>
</ul>
```

# 17. Using Position




```
<h2 className="mb-3">Bootstrap Positioning Example</h2>
<div className="position-relative bg-light border p-3" style={{ height: '200px' }}>
  <div className="p-2 bg-primary text-white"> Static (Default) </div>
  <div className="position-relative p-2 bg-success text-white" style={{ top: '20px', left: '20px' }}> Relative </div>
  <div className="position-absolute p-2 bg-danger text-white" style={{ bottom: '10px', right: '10px' }}> Absolute </div>
</div>
```

# 18. Using Shadow




```
<h2 className="mb-4">Bootstrap Shadow Classes</h2>

<div className="p-3 mb-3 bg-white rounded">No shadow (default)</div>
<div className="p-3 mb-3 bg-white rounded shadow-sm">Small shadow</div>
<div className="p-3 mb-3 bg-white rounded shadow">Regular shadow</div>
<div className="p-3 mb-3 bg-white rounded shadow-lg">Larger shadow</div>
<div className="p-3 mb-3 bg-white rounded shadow-none">No shadow (forced)</div>
```

# 19. Using Tables

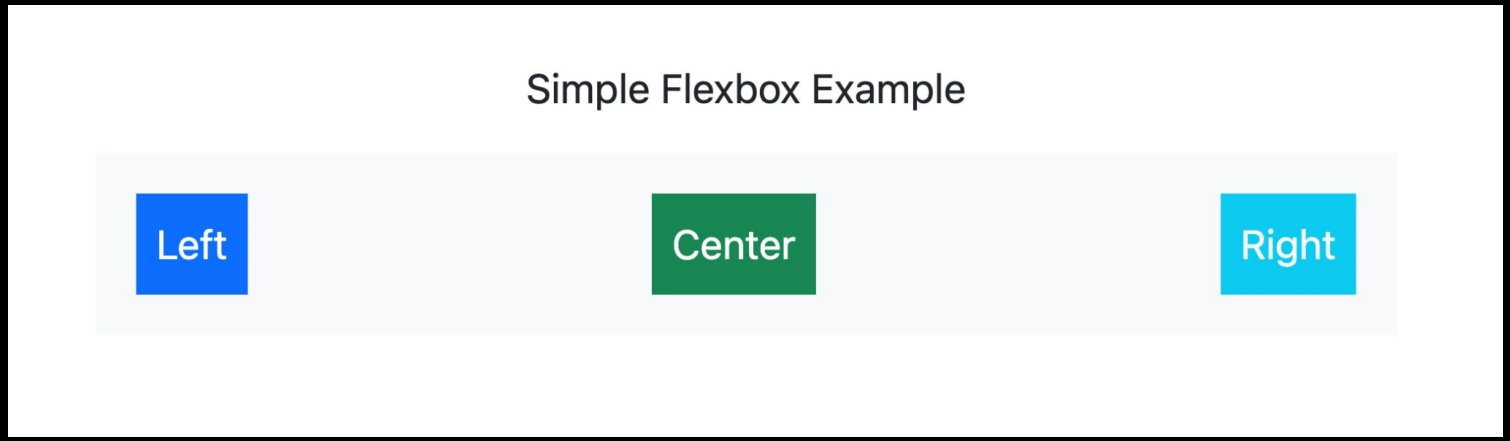
React

```
<h2 className="mb-4">Simple Bootstrap Table</h2>

<table className="table table-striped table-hover">
  <thead className="table-dark">
    <tr>
      <th scope="col">#</th>
      <th scope="col">First Name</th>
      <th scope="col">Last Name</th>
      <th scope="col">Email</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th scope="row">1</th>
      <td>John</td>
      <td>Doe</td>
      <td>john@example.com</td>
    </tr>
    <tr>
      <th scope="row">2</th>
      <td>Jane</td>
      <td>Smith</td>
      <td>jane@example.com</td>
    </tr>
  </tbody>
</table>
```

Simple Bootstrap Table

| # | First Name | Last Name | Email |
|---|---|---|---|
| 1 | John | Doe | john@example.com |
| 2 | Jane | Smith | jane@example.com |

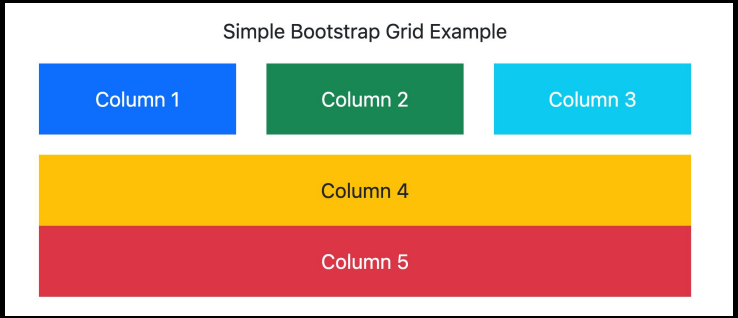
# 20. Using FlexBox




```
<div className="container mt-4">
  <h3 className="mb-3">Simple Flexbox Example</h3>

  <div className="d-flex justify-content-between align-items-center bg-light p-3">
    <div className="p-2 bg-primary text-white">Left</div>
    <div className="p-2 bg-success text-white">Center</div>
    <div className="p-2 bg-info text-white">Right</div>
  </div>
</div>
```

# 21. Using Grid

```
<h3 className="mb-3">Simple Bootstrap Grid Example</h3>

<div className="row">
  <div className="col-sm-4">
    <div className="p-3 bg-primary text-white">Column 1</div>
  </div>
  <div className="col-sm-4">
    <div className="p-3 bg-success text-white">Column 2</div>
  </div>
  <div className="col-sm-4">
    <div className="p-3 bg-info text-white">Column 3</div>
  </div>
</div>

<div className="row mt-3">
  <div className="col-md-6">
    <div className="p-3 bg-warning text-dark">Column 4</div>
  </div>
  <div className="col-md-6">
    <div className="p-3 bg-danger text-white">Column 5</div>
  </div>
</div>
```

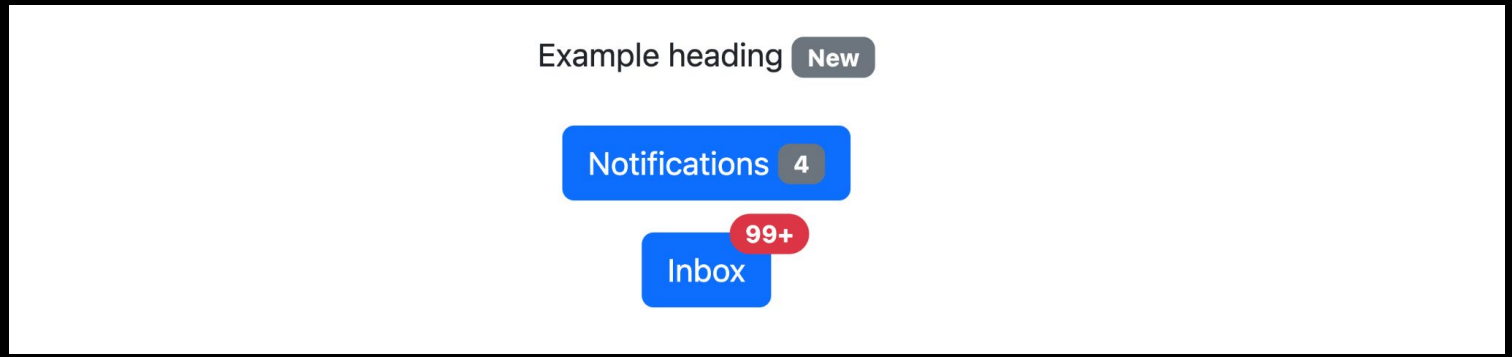



1. **Rows**: `row`: Creates a horizontal group of columns
2. **Columns**:
   - `col-sm-4`: Creates a column that spans 4 out of 12 grid columns on small screens and up
   - `col-md-6`: Creates a column that spans 6 out of 12 grid columns on medium screens and up

Key Bootstrap grid classes:

- `container`: Creates a responsive container
- `row`: Defines a row in the grid system
- `col-sm-4`: Defines a column that takes up 1/3 of the row width on small screens and up
- `col-md-6`: Defines a column that takes up 1/2 of the row width on medium screens and up

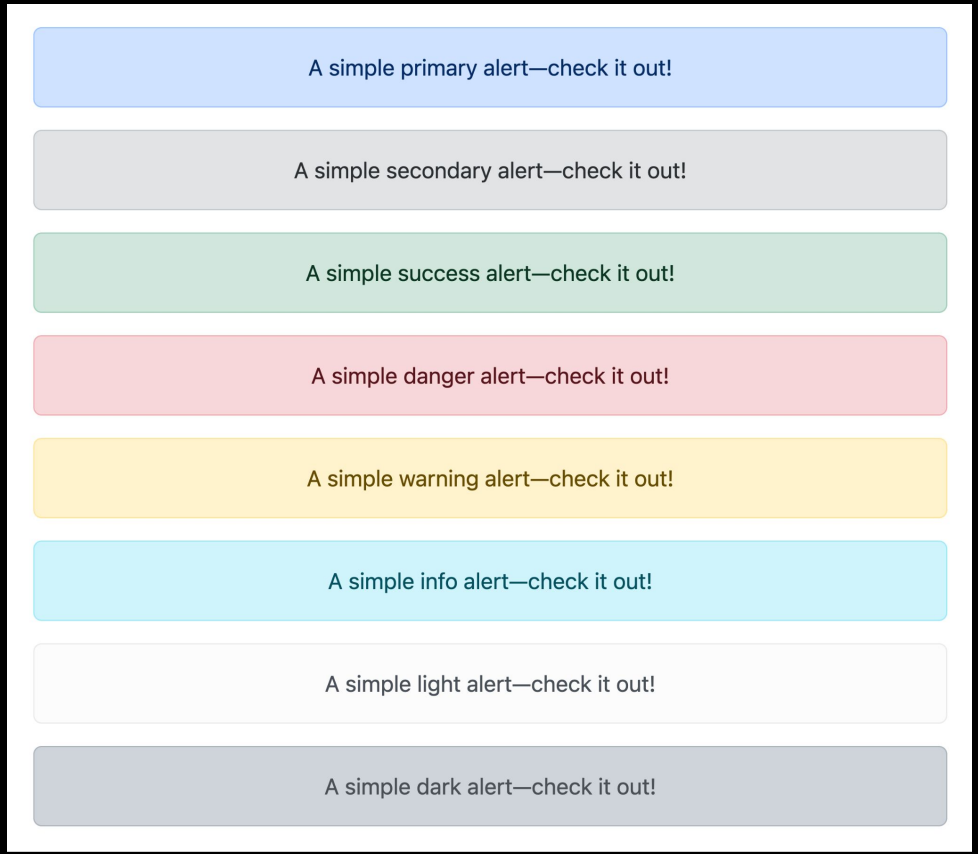
# 22. Using Badges




```
<h4>Example heading <span class="badge text-bg-secondary">New</span></h4>
<br />
<button type="button" class="btn btn-primary">
  Notifications <span class="badge text-bg-secondary">4</span>
</button><br />
<button type="button" class="mt-3 btn btn-primary position-relative">
  Inbox
  <span class="position-absolute top-0 start-100 translate-middle badge rounded-pill bg-danger">
    99+
    <span class="visually-hidden">unread messages</span>
  </span>
</button>
```

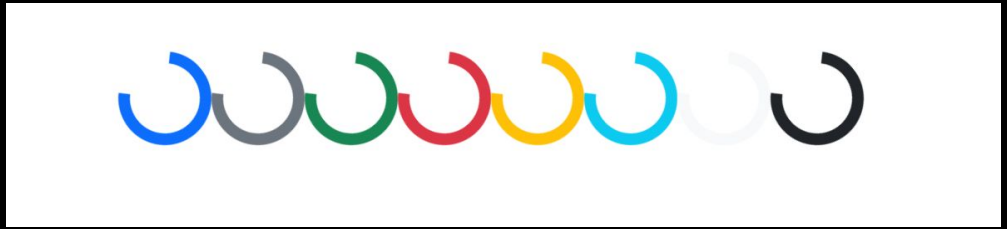
# 23. Using Alerts

```
<div class="alert alert-primary" role="alert">
  A simple primary alert—check it out!
</div>
<div class="alert alert-secondary" role="alert'
  A simple secondary alert—check it out!
</div>
<div class="alert alert-success" role="alert">
  A simple success alert—check it out!
</div>
<div class="alert alert-danger" role="alert">
  A simple danger alert—check it out!
</div>
<div class="alert alert-warning" role="alert">
  A simple warning alert—check it out!
</div>
<div class="alert alert-info" role="alert">
  A simple info alert—check it out!
</div>
<div class="alert alert-light" role="alert">
  A simple light alert—check it out!
</div>
<div class="alert alert-dark" role="alert">
  A simple dark alert—check it out!
</div>
```

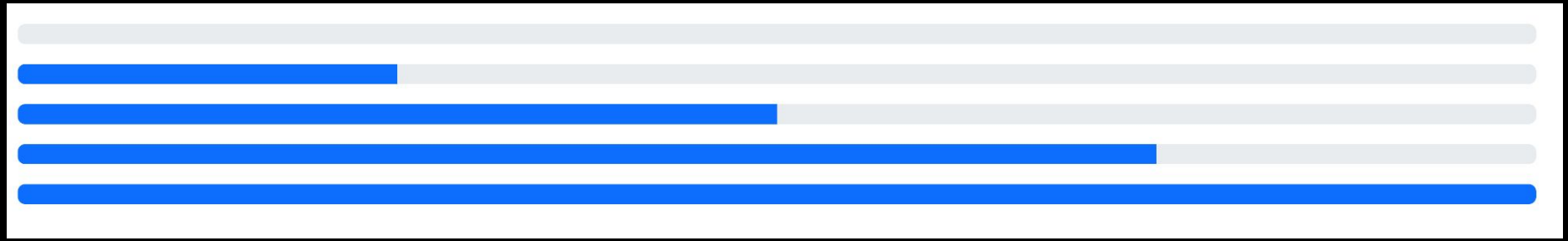


# 24. Using Spinner

```
<div class="spinner-border text-primary" role="status">
  <span class="visually-hidden">Loading...</span>
</div>
<div class="spinner-border text-secondary" role="status">
  <span class="visually-hidden">Loading...</span>
</div>
<div class="spinner-border text-success" role="status">
  <span class="visually-hidden">Loading...</span>
</div>
<div class="spinner-border text-danger" role="status">
  <span class="visually-hidden">Loading...</span>
</div>
<div class="spinner-border text-warning" role="status">
  <span class="visually-hidden">Loading...</span>
</div>
<div class="spinner-border text-info" role="status">
  <span class="visually-hidden">Loading...</span>
</div>
<div class="spinner-border text-light" role="status">
  <span class="visually-hidden">Loading...</span>
</div>
<div class="spinner-border text-dark" role="status">
  <span class="visually-hidden">Loading...</span>
</div>
```

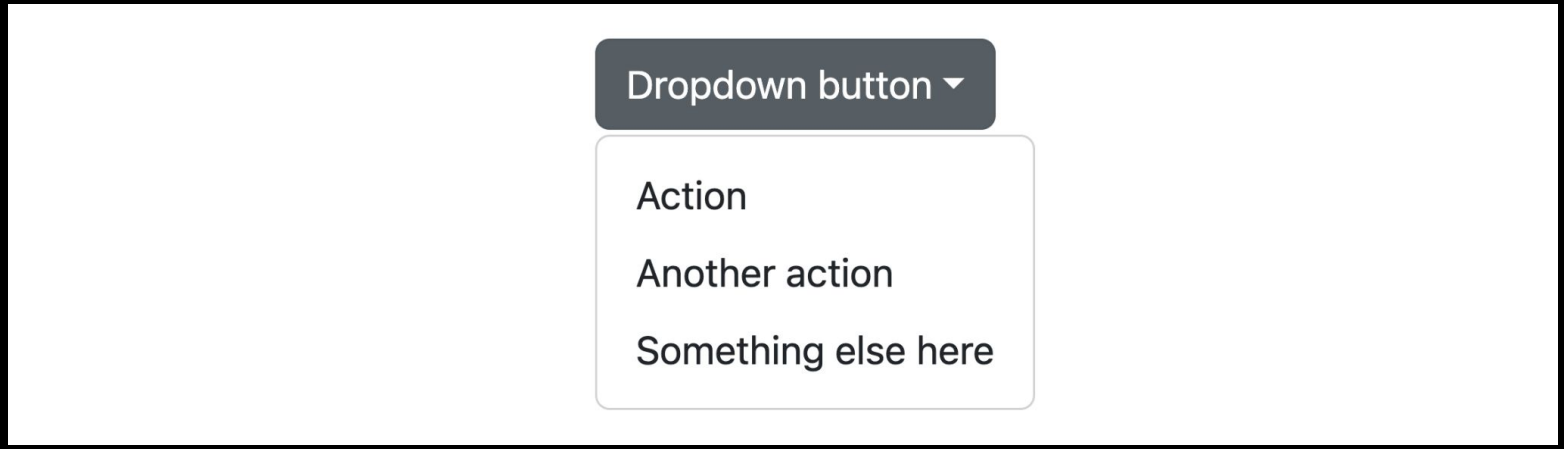


# 25. Using Progress-Bar



```
<div className="progress mt-3" role="progressbar" aria-label="Basic example" aria-valuenow="0" aria-valuemin="0" aria-valuemax="100">
  <div className="progress-bar" style={{ width: "0%" }}></div>
</div>
<div className="progress mt-3" role="progressbar" aria-label="Basic example" aria-valuenow="25" aria-valuemin="0" aria-valuemax="100">
  <div className="progress-bar" style={{ width: "25%" }}></div>
</div>
<div className="progress mt-3" role="progressbar" aria-label="Basic example" aria-valuenow="50" aria-valuemin="0" aria-valuemax="100">
  <div className="progress-bar" style={{ width: "50%" }}></div>
</div>
<div className="progress mt-3" role="progressbar" aria-label="Basic example" aria-valuenow="75" aria-valuemin="0" aria-valuemax="100">
  <div className="progress-bar" style={{ width: "75%" }}></div>
</div>
<div className="progress mt-3" role="progressbar" aria-label="Basic example" aria-valuenow="100" aria-valuemin="0" aria-valuemax="100">
  <div className="progress-bar" style={{ width: "100%" }}></div>
</div>
```

# 26. Using Drop-Down




```
<div class="dropdown">
  <button class="btn btn-secondary dropdown-toggle" type="button" data-bs-toggle="dropdown" aria-expanded="false">
    Dropdown button
  </button>
  <ul class="dropdown-menu">
    <li><a class="dropdown-item" href="#">Action</a></li>
    <li><a class="dropdown-item" href="#">Another action</a></li>
    <li><a class="dropdown-item" href="#">Something else here</a></li>
  </ul>
</div>
```

# 27. Using Card

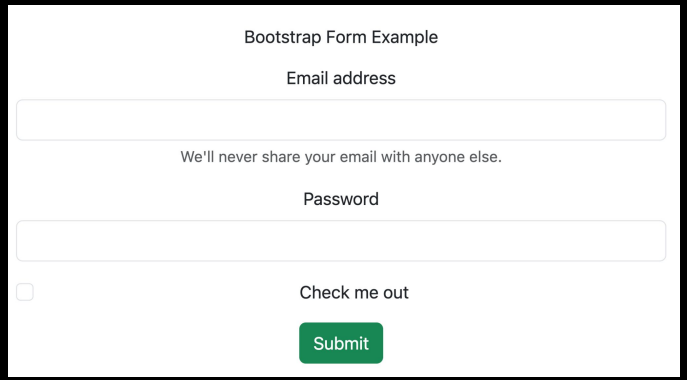
```
<div className="card" style={{ width: "18rem" }}>
  <img src="https://a0.muscache.com/im/pictures/
  1ff6d909-5ba6-42f3-9d2c-fa2327780936.jpg"
  className="card-img-top" alt="..." />
  <div className="card-body">
    <h5 className="card-title">Card title</h5>
    <p className="card-text">Some quick example text to
    build on the card title and make up the bulk of the
    card's content.</p>
    <a href="#" className="btn btn-primary">Go somewhere</a>
  </div>
</div>
```

Card title

Some quick example text to build on the card title and make up the bulk of the card's content.

Go somewhere

# 28. Using Form & Form-Control




The `form-control` class is a fundamental part of Bootstrap's form styling. It's applied to form input elements to give them a consistent, styled appearance across different browsers and devices.

```
<form>
  <div className="mb-3">
    <label htmlFor="exampleInputEmail1" className="form-label">Email address</label>
    <input type="email" className="form-control" id="exampleInputEmail1" aria-describedby="emailHelp" />
    <div id="emailHelp" className="form-text">We'll never share your email with anyone else.</div>
  </div>
  <div className="mb-3">
    <label htmlFor="exampleInputPassword1" className="form-label">Password</label>
    <input type="password" className="form-control" id="exampleInputPassword1" />
  </div>
  <div className="mb-3 form-check">
    <input type="checkbox" className="form-check-input" id="exampleCheck1" />
    <label className="form-check-label" htmlFor="exampleCheck1">Check me out</label>
  </div>
  <button type="submit" className="btn btn-success">Submit</button>
</form>
```


React

# Exercise: Create Form using Bootstrap


## Student Details Form

Full Name

Enter your full name

Email address

Enter your email

Age

Enter your age

Grade

Select your grade

Gender

- Male
- Female
- Other

I agree to the terms and conditions

Submit


# Bootstrap: Classes to be used

`container`

- Creates a responsive fixed-width container
- Centers the content and provides padding on the sides
-

`form-label`

- Styles form labels
- Typically adds some bottom margin and makes the text bold

`form-control`

- Styles form inputs and textareas
- Provides full-width, rounded corners, and consistent padding
- Adds focus styles and hover effects

`form-select`

- Styles select dropdowns
- Similar to form-control but with a custom dropdown arrow

`form-check`

- Container class for custom radio buttons and checkboxes
- Provides proper spacing and alignment

`form-check-input`

- Styles the actual input for radio buttons and checkboxes
- Works with the form-check class to create custom-styled inputs

`form-check-label`

- Styles labels for radio buttons and checkboxes
- Provides proper alignment with the custom inputs

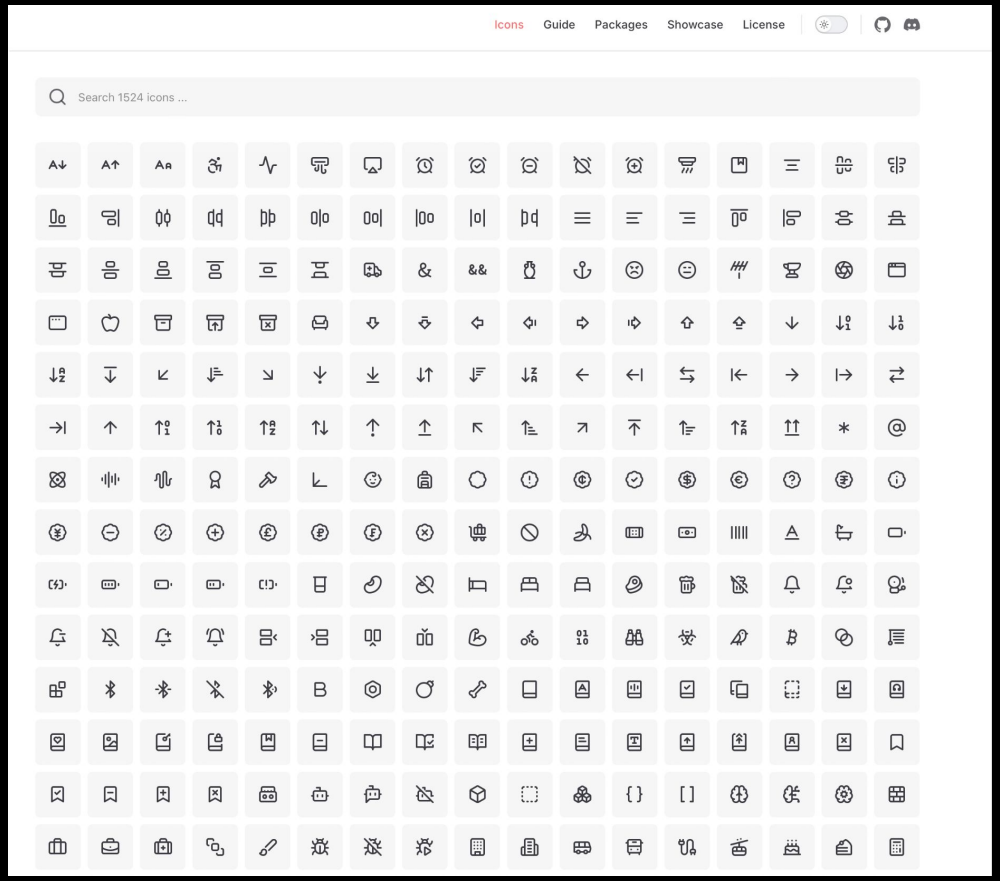
```jsx
import React from 'react';

const StudentDetailsForm = () => {
  return (
    <div className="container mt-5">
      <h2 className="mb-4">Student Details Form</h2>
      <form>
        <div className="mb-3">
          <label htmlFor="fullName" className="form-label">Full Name</label>
          <input type="text" className="form-control" id="fullName" placeholder="Enter your full name" />
        </div>
        <div className="mb-3">
          <label htmlFor="email" className="form-label">Email address</label>
          <input type="email" className="form-control" id="email" placeholder="Enter your email" />
        </div>
        <div className="mb-3">
          <label htmlFor="age" className="form-label">Age</label>
          <input type="number" className="form-control" id="age" placeholder="Enter your age" />
        </div>
        <div className="mb-3">
          <label htmlFor="grade" className="form-label">Grade</label>
          <select className="form-select" id="grade">
            <option selected>Select your grade</option>
            <option value="9">9th Grade</option>
            <option value="10">10th Grade</option>
            <option value="11">11th Grade</option>
            <option value="12">12th Grade</option>
          </select>
        </div>
        <div className="mb-3">
          <label className="form-label">Gender</label>
```

```
32            <div className="form-check">
33              <input className="form-check-input" type="radio" name="gender" id="male" />
34              <label className="form-check-label" htmlFor="male">
35                Male
36              </label>
37            </div>
38            <div className="form-check">
39              <input className="form-check-input" type="radio" name="gender" id="female" />
40              <label className="form-check-label" htmlFor="female">
41                Female
42              </label>
43            </div>
44            <div className="form-check">
45              <input className="form-check-input" type="radio" name="gender" id="other" />
46              <label className="form-check-label" htmlFor="other">
47                Other
48              </label>
49            </div>
50          </div>
51          <div className="mb-3 form-check">
52            <input type="checkbox" className="form-check-input" id="termsCheck" />
53            <label className="form-check-label" htmlFor="termsCheck">I agree to the terms and conditions</label>
54          </div>
55          <button type="submit" className="btn btn-primary">Submit</button>
56        </form>
57      </div>
58    );
59  };
60
61  export default StudentDetailsForm;
```


React

# Lucide React

# Lucide React




# Including Lucide React

## Lucide React

Implementation of the lucide icon library for react applications

---

## Installation

pnpm yarn npm

sh

```
npm install lucide-react
```


# Using Lucide React

## How to use

Lucide is built with ES Modules, so it's completely tree-shakable.

Each icon can be imported as a React component, which renders an inline SVG element.
This way, only the icons that are imported into your project are included in the final bundle.
The rest of the icons are tree-shaken away.

## Example

Additional props can be passed to adjust the icon:

```jsx
import { Camera } from 'lucide-react';

// Usage
const App = () => {
  return <Camera color="red" size={48} />;
};

export default App;
```


# Using Lucide: Props

## Props

| name | type | default |
| --- | --- | --- |
| `size` | *number* | 24 |
| `color` | *string* | currentColor |
| `strokeWidth` | *number* | 2 |
| `absoluteStrokeWidth` | *boolean* | false |

## Applying props

To customize the appearance of an icon, you can pass custom properties as props directly to the component. The component accepts all SVG attributes as props, which allows flexible styling of the SVG elements. See the list of SVG Presentation Attributes on MDN.

jsx

```jsx
// Usage
const App = () => {
  return <Camera size={48} fill="red" />;
};
```


# Using Lucide Lab

## With Lucide lab or custom icons

Lucide lab is a collection of icons that are not part of the Lucide main library.

They can be used by using the `Icon` component. All props like regular lucide icons can be passed to adjust the icon appearance.

### Using the `Icon` component

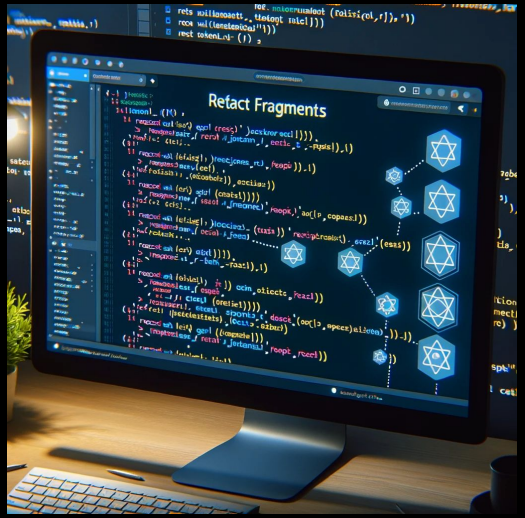
This creates a single icon based on the iconNode passed and renders a Lucide icon component.

```jsx
import { Icon } from 'lucide-react';
import { burger } from '@lucide/lab';

const App = () => (
  <Icon iconNode={burger} />
);
```

npm install @lucide/lab

# Fragments

1. What?
   Allows grouping of multiple elements without extra DOM nodes.
2. Why?
   - Return multiple elements without a wrapping parent.
   - Cleaner DOM and consistent styling.
3. How? Two syntaxes:
   1. <React.Fragment>...</React.Fragment>
   2. Short: <>...</>



# Map Method

1. Purpose: Render lists from array data.
2. JSX Elements: Transform array items into JSX.
3. Inline Rendering: Directly inside JSX

```
{
  items.map(item =>
    <li key={item.id}>{item.name}</li>)
}
```

4. Key Prop: Assign unique key for optimized re-renders.

```
<div key={item.id}>{item.name}</div>
```

# Conditional Rendering

React

## Conditional Rendering

- Displaying content based on certain conditions.
- Allows for dynamic user interfaces.

## Methods

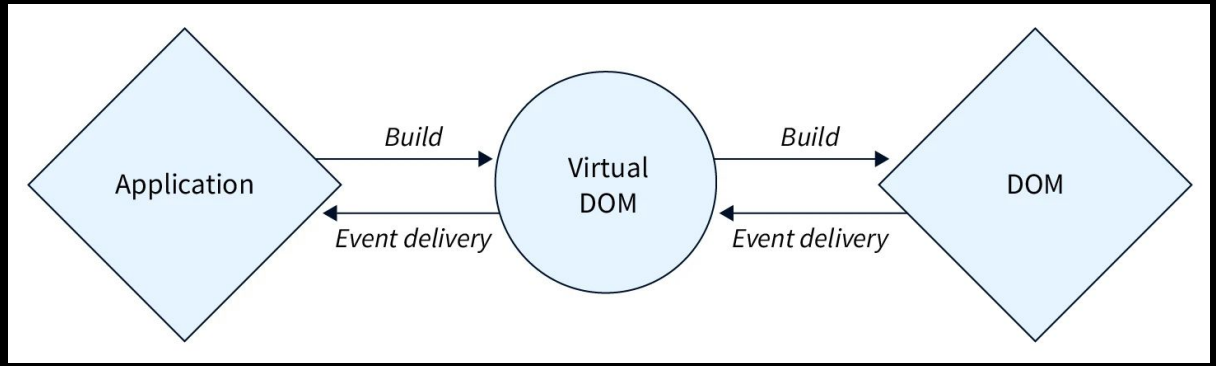
- If-else statements: Choose between two blocks of content.
- Ternary operators: Quick way to choose between two options.
- Logical operators: Useful for rendering content when a condition is true.

## Benefits

- Enhances user experience.
- Reduces unnecessary rendering.
- Makes apps more interactive and responsive.

```
{ condition && <div>Write something</div> }

{ !condition ? <div>Error do it again</div> :
               <div>Congratulations</div> }
```

# Handling Events




1. React events use camelCase, e.g., onClick.
2. Uses synthetic events, not direct browser events.
3. Event handlers can be functions or arrow functions.
4. Use onChange for controlled form inputs.
5. Avoid inline arrow functions in JSX for performance.

React

# CSS Modules


1. Localized class names to avoid global conflicts.
2. Styles are scoped to individual components.
3. Helps in creating component-specific styles.
4. Automatically generates unique class names.
5. Promotes modular and maintainable CSS.
6. Can use alongside global CSS when needed.


# Passing Children

```jsx
function Container(props) {
  return (
    <div className="container-style">
      {props.children}
    </div>
  );
}
```

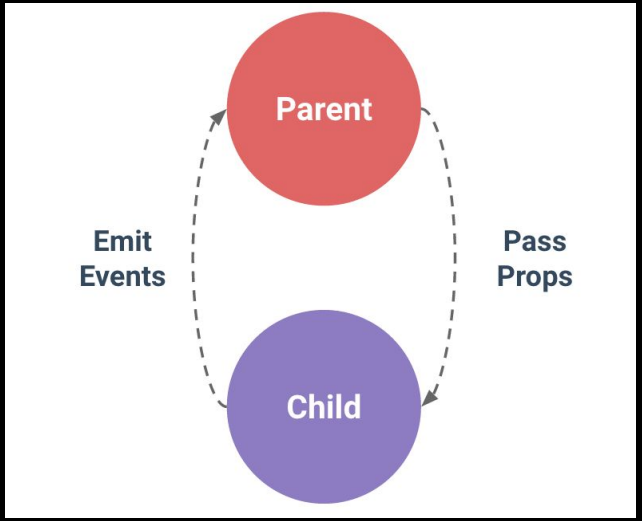
```jsx
<Container>
  <h1>Welcome to My App</h1>
  <p>This content is passed as children to the
  Container component.</p>
</Container>
```

1. children is a special prop for passing elements into components.
2. Used for flexible and reusable component designs.
3. Common in layout or container components.
4. Accessed with props.children.
5. Can be any content: strings, numbers, JSX, or components.
6. Enhances component composability and reusability.

# Passing Functions via Props

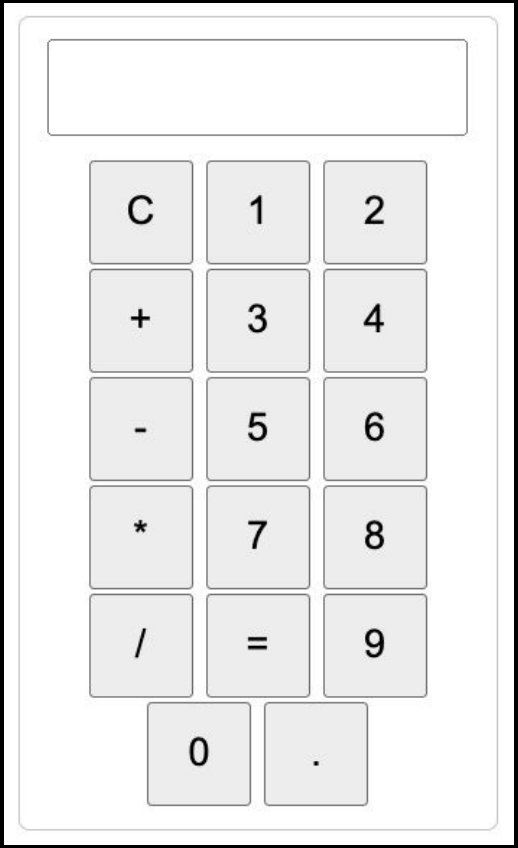
1. Pass dynamic behaviour between components.
2. Enables upward communication from child to parent.
3. Commonly used for event handling.
4. Parent defines a function, child invokes it.
5. Enhances component interactivity.
6. Example:

```
<Button onClick={handleClick} />
```

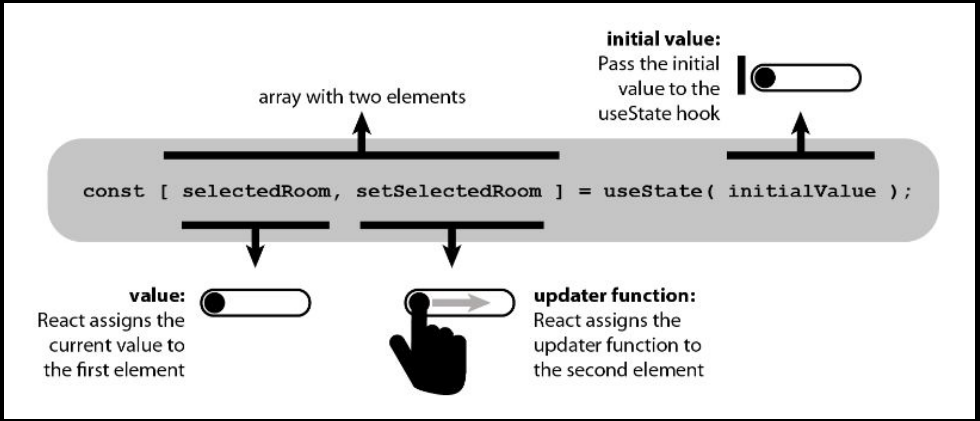


React

# Project Calculator



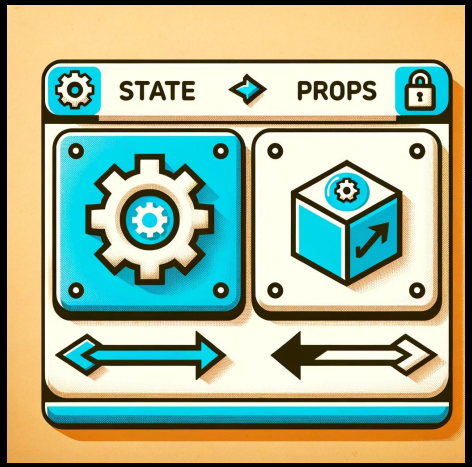
# Managing State




1. State represents data that changes over time.
2. State is local and private to the component.
3. State changes cause the component to re-render.
4. For functional components, use the useState hook.
5. React Functions that start with word use are called hooks
6. Hooks should only be used inside components
7. Parent components can pass state down to children via props.
8. Lifting state up: share state between components by moving it to their closest common ancestor.

React

# State vs Props




## State:

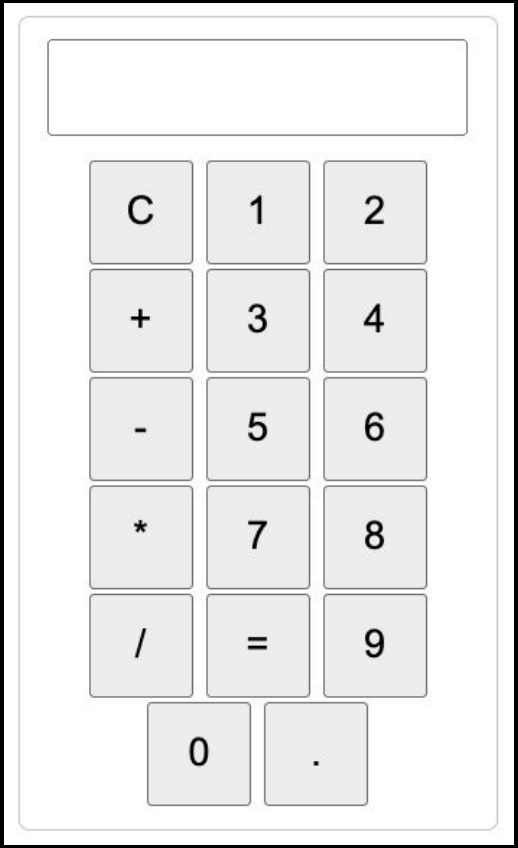
- Local and mutable data within a component.
- Initialized within the component.
- Can change over time.
- Causes re-render when updated.
- Managed using useState in functional components.

## Props:

- Passed into a component from its parent.
- Read-only (immutable) within the receiving component.
- Allow parent-to-child component communication.
- Changes in props can also cause a re-render.

React

# Project Calculator



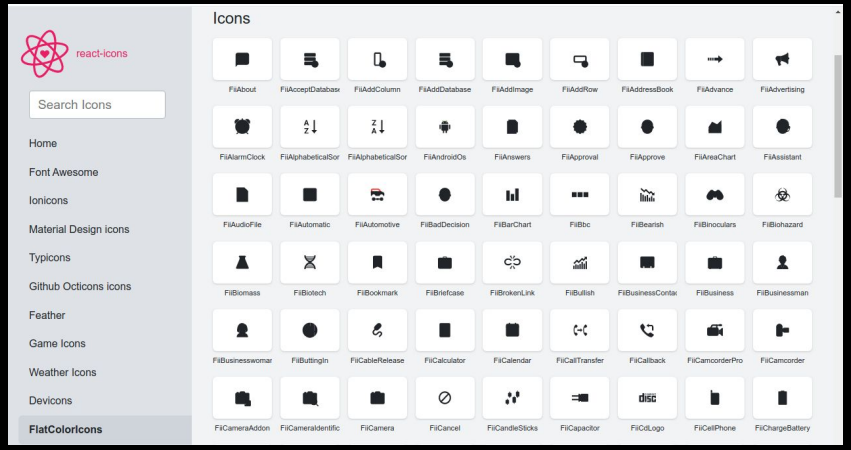
# React-icon Library

1. You can use a lot of icons without managing them.
2. Install Package

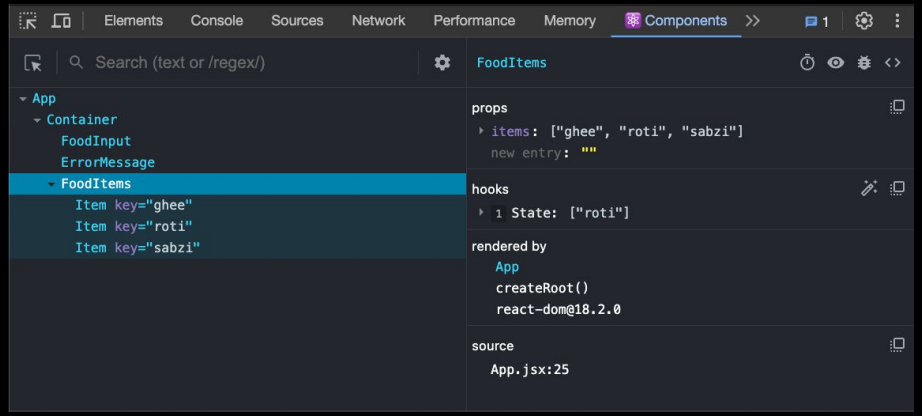
```
npm install react-icons -save
```

3. Use icon:

```
import {IconName} from
"react-icons/fc";
```



# Inspecting with React Dev Tools


1. Inspection: Allows inspection of React component hierarchies.
2. State & Props: View and edit the current state and props of components.
3. Performance: Analyze component re-renders and performance bottlenecks.
4. Navigation: Conveniently navigate through the entire component tree.
5. Filtering: Filter components by name or source to locate them quickly.
6. Real-time Feedback: See live changes as you modify state or props.

# How React Works

## Root Component:

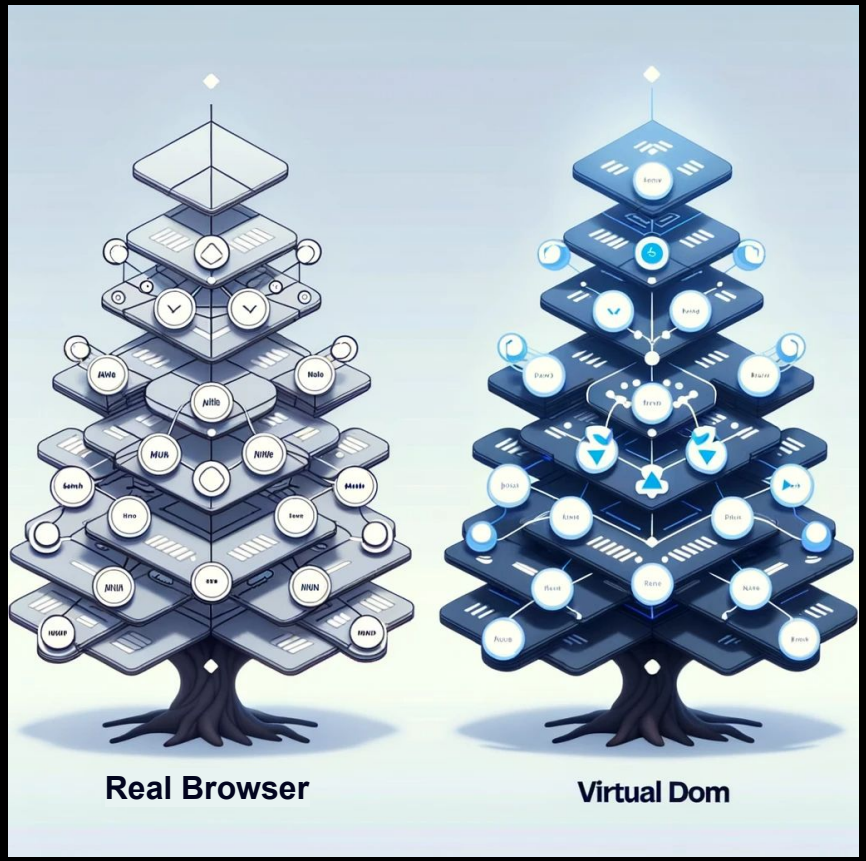
- The App is the main or root component of a React application.
- It's the starting point of your React component tree.

## Virtual DOM:

- React creates an in-memory structure called the virtual DOM.
- Different from the actual browser DOM.
- It's a lightweight representation where each node stands for a component and its attributes.

## Reconciliation Process:

- When component data changes, React updates the virtual DOM's state to mirror these changes.
- React then compares the current and previous versions of the virtual DOM.
- It identifies the specific nodes that need updating.
- Only these nodes are updated in the real browser DOM, making it efficient.



# How React Works

## Root Component:

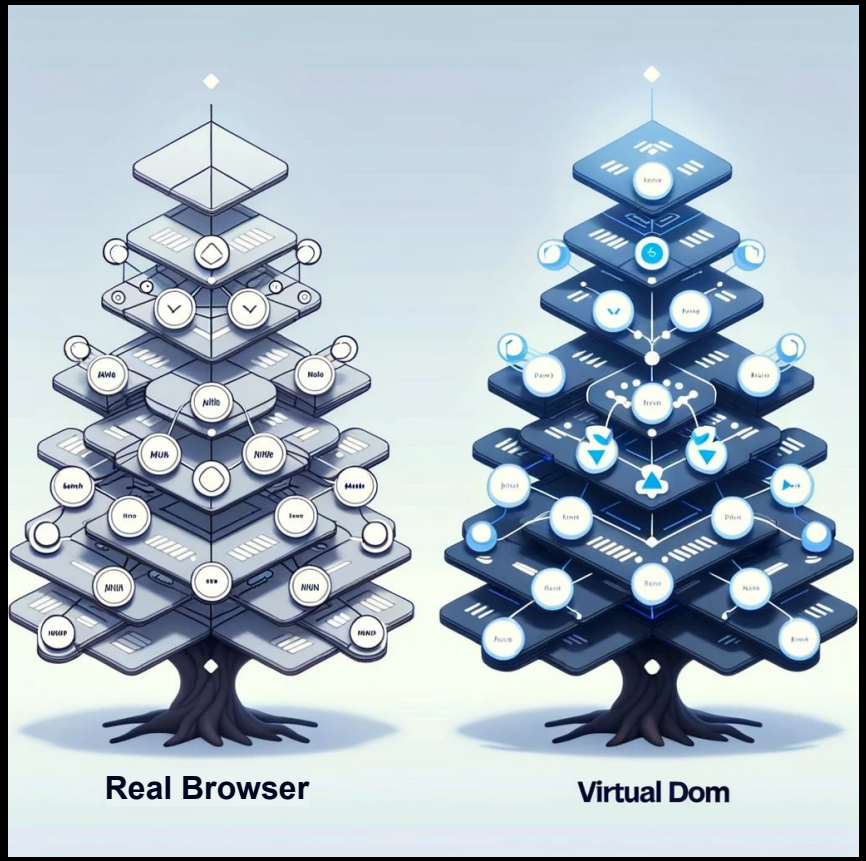
- The App is the main or root component of a React application.
- It's the starting point of your React component tree.

## Virtual DOM:

- React creates an in-memory structure called the virtual DOM.
- Different from the actual browser DOM.
- It's a lightweight representation where each node stands for a component and its attributes.

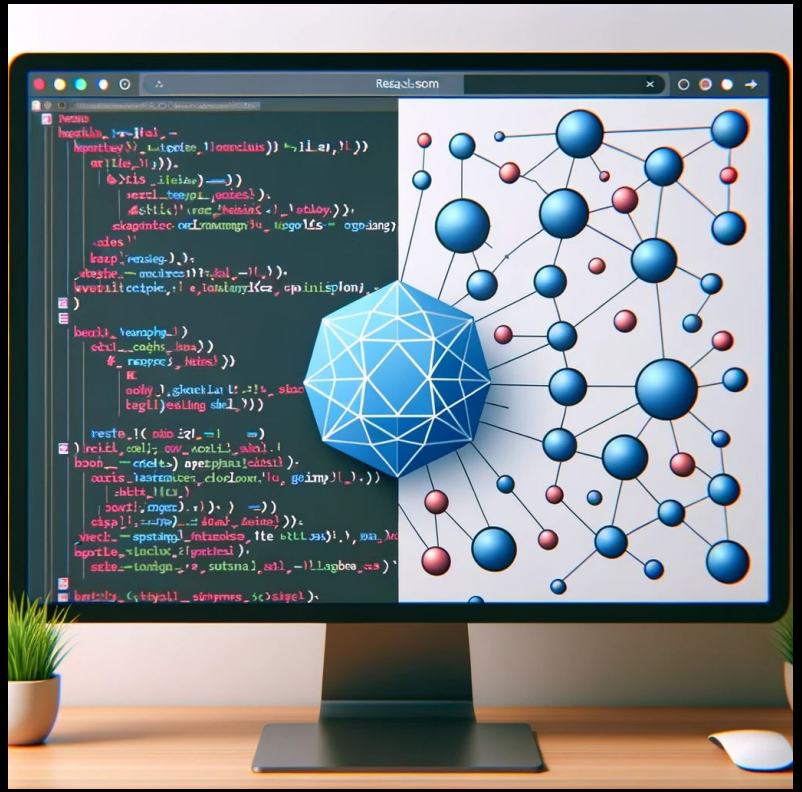
## Reconciliation Process:

- When component data changes, React updates the virtual DOM's state to mirror these changes.
- React then compares the current and previous versions of the virtual DOM.
- It identifies the specific nodes that need updating.
- Only these nodes are updated in the real browser DOM, making it efficient.



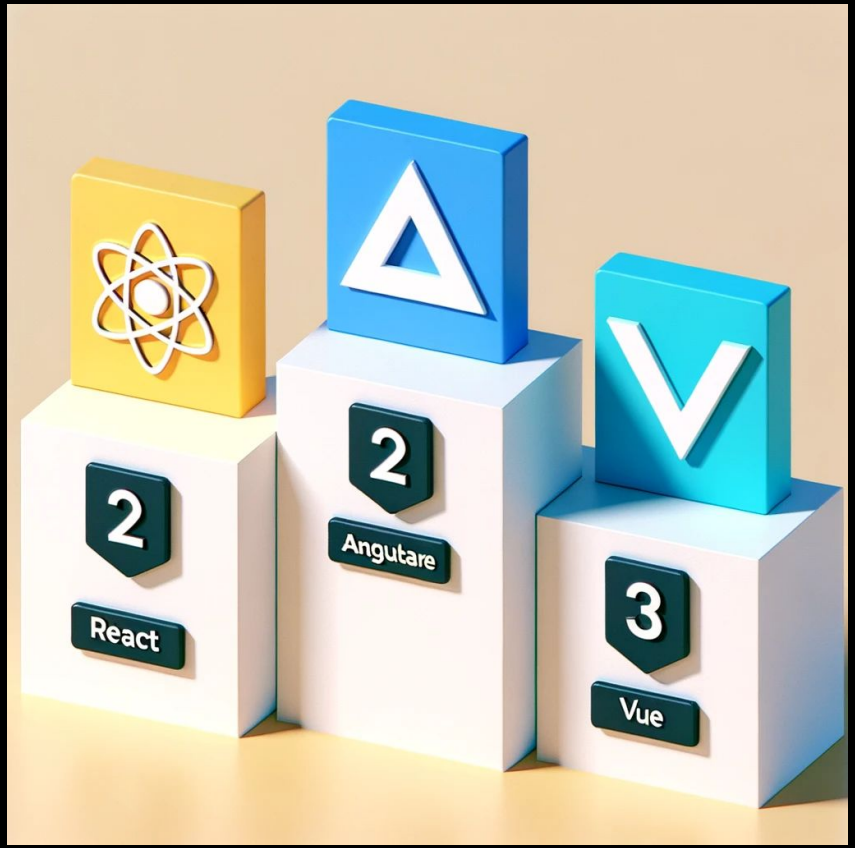
# How React Works

- **React and ReactDOM:**
  - The actual updating of the browser's DOM isn't done by React itself.
  - It's handled by a companion library called react-dom.
- **Root Element:**
  - The root div acts as a container for the React app.
  - The script tag is where the React app starts executing.
  - If you check main.tsx, the component tree is rendered inside this root element.
- **Strict Mode Component:**
  - It's a special component in React.
  - Doesn't have a visual representation.
  - Its purpose is to spot potential issues in your React app.
- **Platform Independence:**
  - React's design allows it to be platform-agnostic.
  - While react-dom helps build web UIs using React, ReactNative can be used to craft mobile app UIs.



# React Vs Angular vs Vue

- **React, Angular, and Vue:**
  - React is a library, while Angular and Vue.js are frameworks.
  - React focuses on UI; Angular and Vue.js offer comprehensive tools for full app development.
- **Library vs. Framework:**
  - A library offers specific functionality.
  - A framework provides a set of tools and guidelines.
  - In simpler terms: React is a tool; Angular and Vue.js are toolsets.
- **React's Specialty:**
  - React's main role is crafting dynamic, interactive UIs.
  - It doesn't handle routing, HTTP calls, state management, and more.
- **React's Flexibility:**
  - React doesn't dictate tool choices for other app aspects.
  - Developers pick what fits their project best.
- **About Angular and Vue.js:**
  - Angular, developed by Google, provides a robust framework with a steep learning curve.
  - Vue.js is known for its simplicity and ease of integration, making it beginner-friendly.



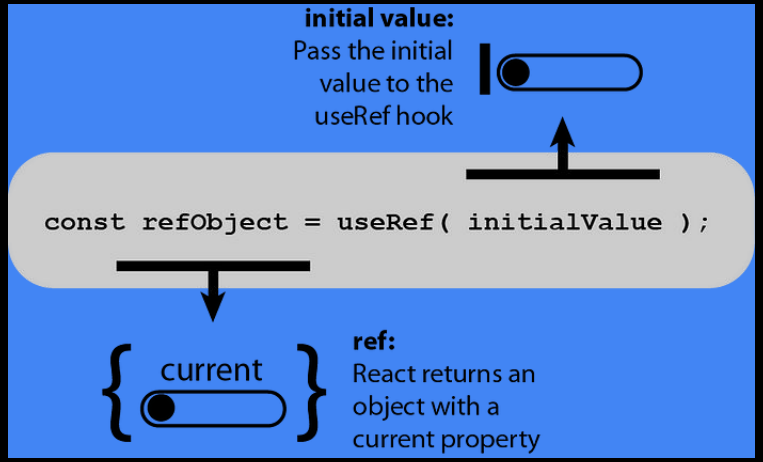
# Using Forms

React

1. State Management: Each input's state is stored in the component's state.
2. Handling Changes: Use onChange to detect input changes.
3. Submission: Utilize onSubmit for form submissions and prevent default with event.preventDefault().
4. Validation: Implement custom validation or use third-party libraries.

# Use Ref

1. useRef allows access to DOM elements and retains mutable values without re-renders.
2. Used with the ref attribute for direct DOM interactions.
3. Can hold previous state or prop values.
4. Not limited to DOM references; can hold any value.
5. Refs can be passed as props also



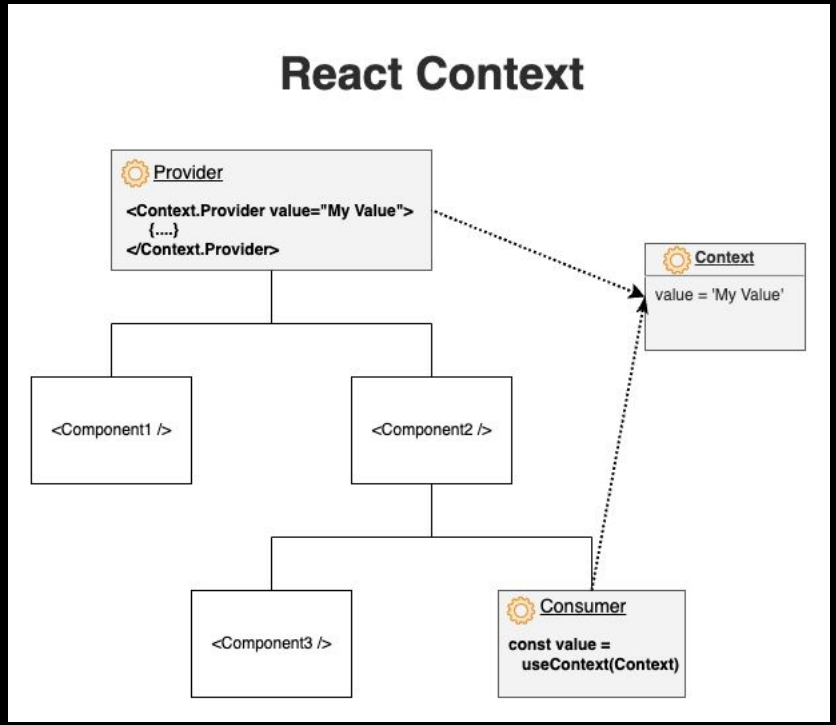
# Update state from Previous State

- **Spread Operator**: Use to maintain immutability when updating arrays or objects.
- **Functional Updates**: Use (existingPosts) => [postData, ...existingPosts] to avoid stale values during asynchronous updates.

# Project: TODO App

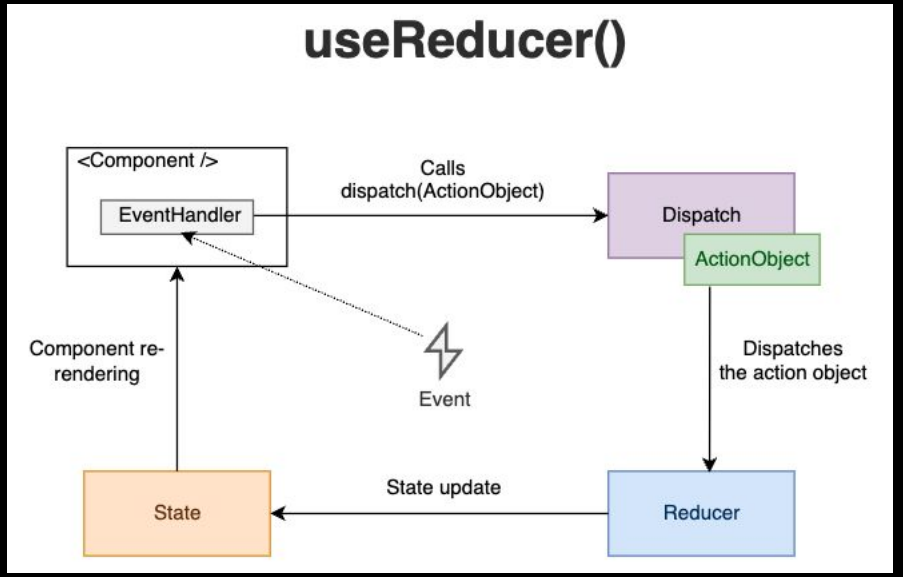
# Context API

1. Prop Drilling: Context API addresses prop drilling; component composition is an alternative.
2. Folder Setup: Use a store folder for context files.
3. Initialization: Use React.createContext with initial state and export it.
4. Provider: Implement with contextName.Provider in components.
5. Access Value: Use the useContext hook.
6. Dynamic Data: Combine context value with state.
7. Export Functions: Context can also export functions for actions
8. Logic Separation: This helps keep the UI and business logic separate from each other.

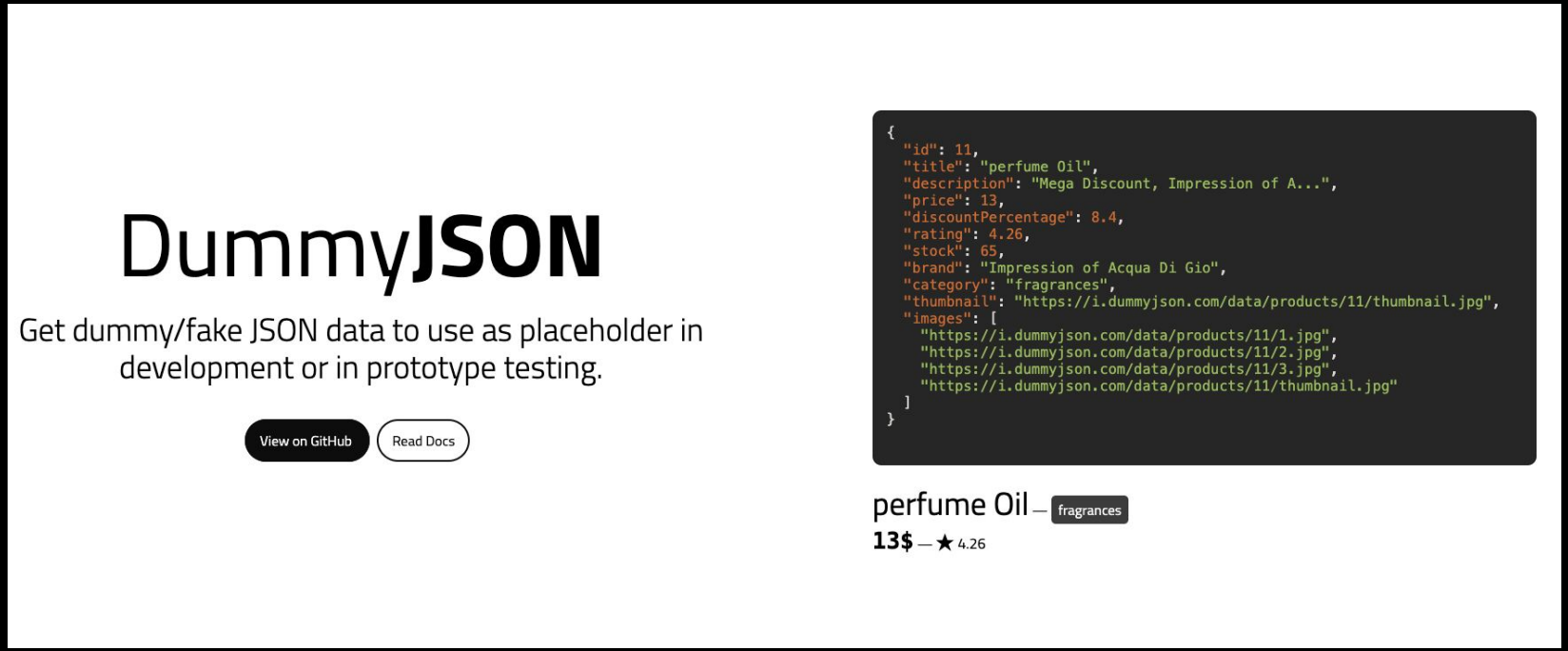


# Use Reducer

1. useReducer is a hook in React that offers more control over state operations compared to useState, especially for complex state logic.
2. Components: It involves two main components:
   - Reducer: A pure function that takes the current state and an action and returns a new state.
   - Action: An object describing what happened, typically having a type property.
3. Initialization: It's invoked as `const [state, dispatch] = useReducer(reducer, initialState)`.
4. Dispatch: Actions are dispatched using the dispatch function, which invokes the reducer with the current state and the given action.
5. Use Cases: Particularly useful for managing state in large components or when the next state depends on the previous one.
6. Predictable State Management: Due to its strict structure, it leads to more predictable and maintainable state management.

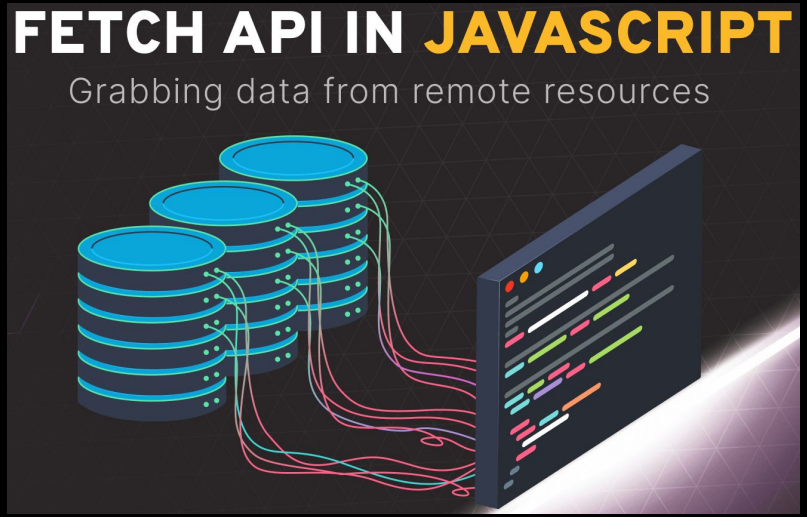


# Introducing Dummy API



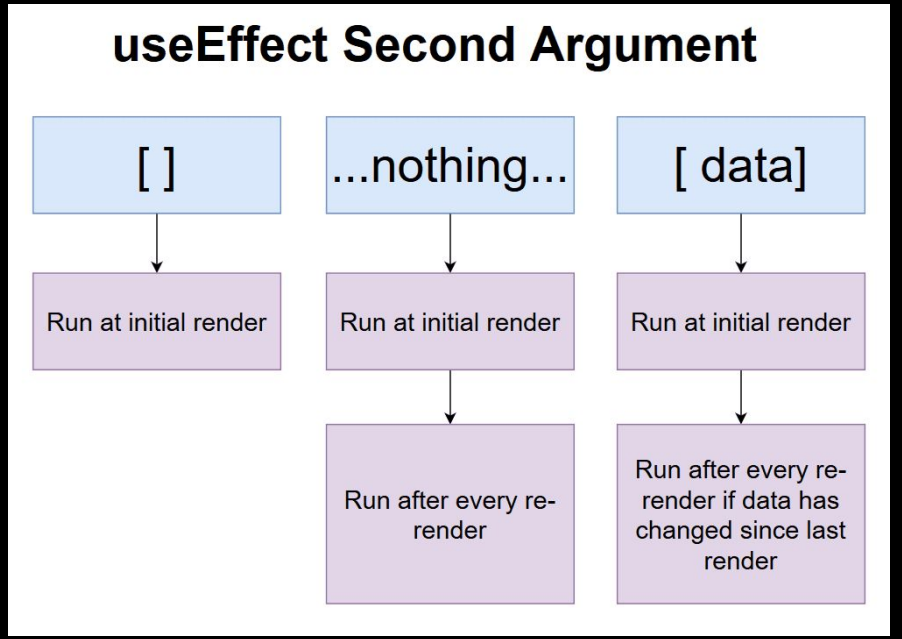
# Data fetching using Fetch

1. fetch: Modern JavaScript API for network requests.
2. Promise-Based: Returns a Promise with a Response object.
3. Usage: Default is GET. For POST use method: 'POST'
4. Response: Use .then() and response.json() for JSON data.
5. Errors: Doesn't reject on HTTP errors. Check response.ok.
6. Headers: Managed using the Headers API.



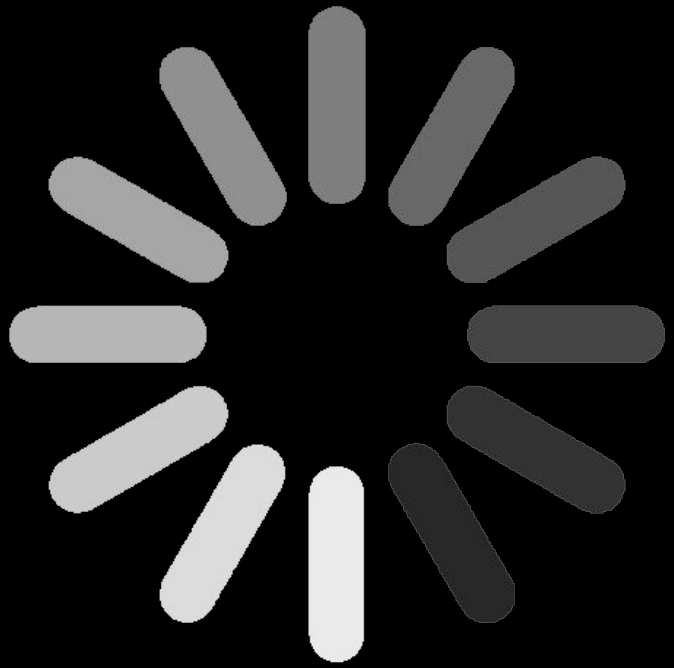
# The useEffect Hook

1. In function-based components, useEffect handles side effects like data fetching or event listeners.
2. useEffect runs automatically after every render by default.
3. By providing a dependency array, useEffect will only run when specified variables change. An empty array means the effect runs once.
4. Multiple useEffect hooks can be used in a single component for organizing different side effects separately.

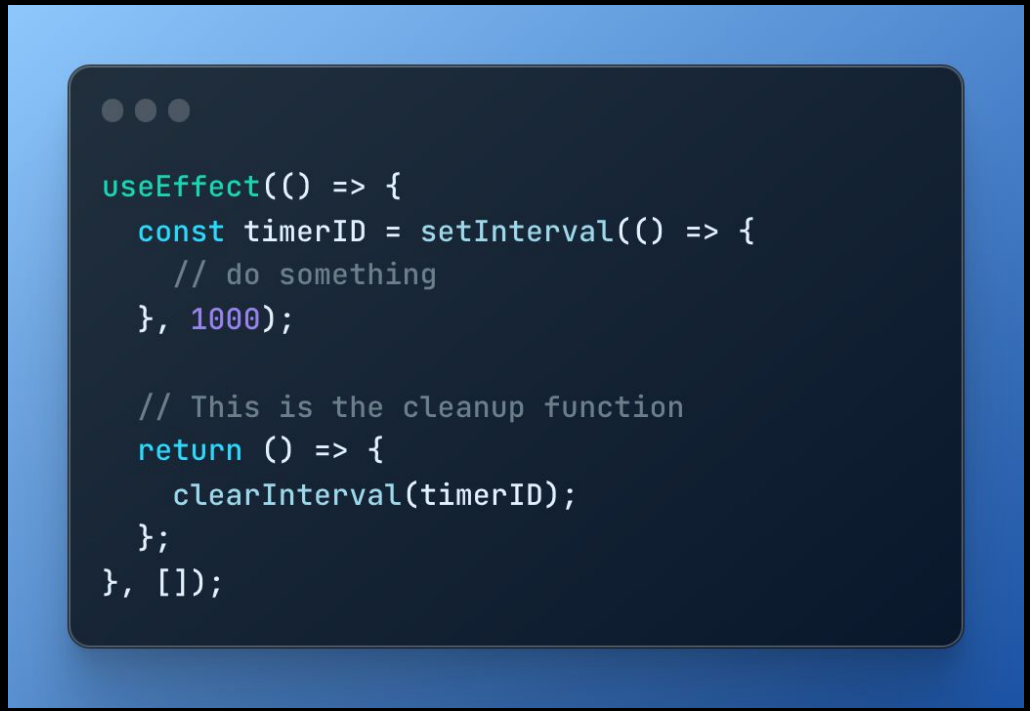


React

# Handling Loading State



# The useEffect Hook Cleanup



```
useEffect(() => {
  const timerID = setInterval(() => {
    // do something
  }, 1000);

  // This is the cleanup function
  return () => {
    clearInterval(timerID);
  };
}, []);
```

Returning a function from `useEffect` allows for cleanup, ideal for removing event listeners.

# Advanced useEffect

React

## Junior

```
useEffect(() => {
  fetch(`/api/users/${id}`)
    .then((res) => res.json(
    .then((data) => {
      setUser(data);
    });
}, [id]);
```

## Pro

```
useEffect(() => {
  const controller = new AbortController();
  const signal = controller.signal;

  fetch(`/api/users/${id}`, { signal })
    .then((res) => res.json())
    .then((data) => {
      setUser(data);
    });

  return () => {
    controller.abort();
  };
}, [id]);
```