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;
}
```

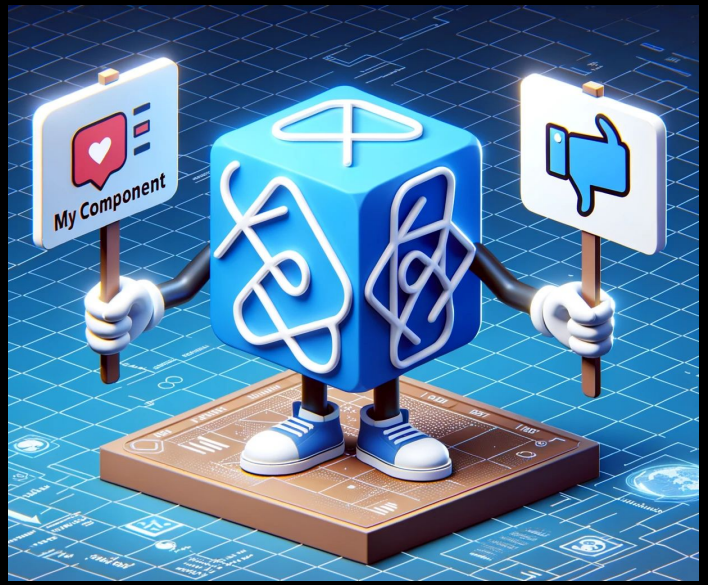
Exporting/Importing components

```
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

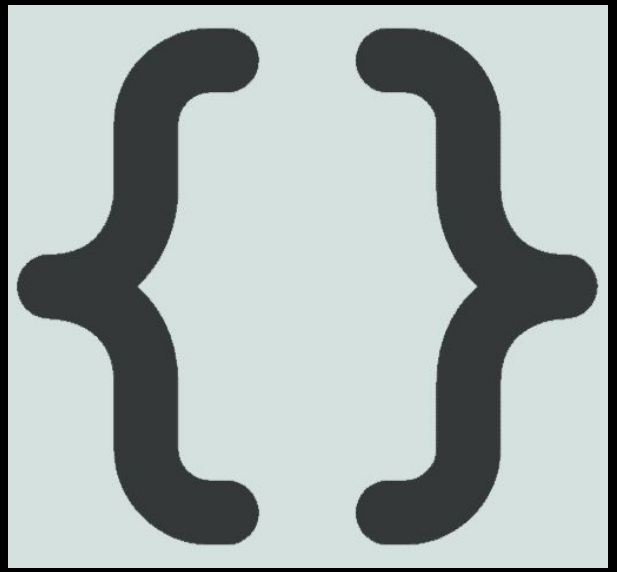
Important Points

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

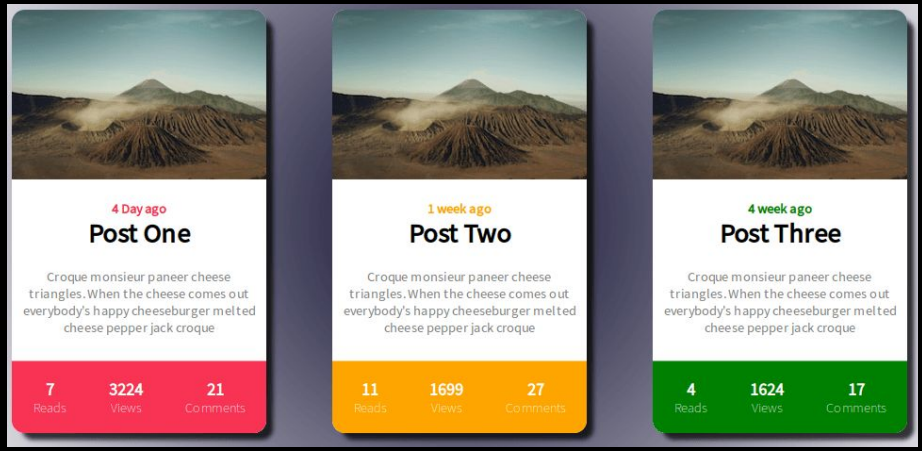
```
function Hello() {
  let myName = 'Prashant';
  let number = 456;
  let fullName = () => {
    return 'Prashant Jain';
  }

  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;
```

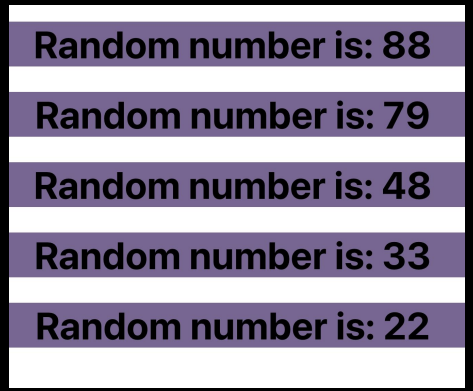
Reusable Components

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>
  );
}
```