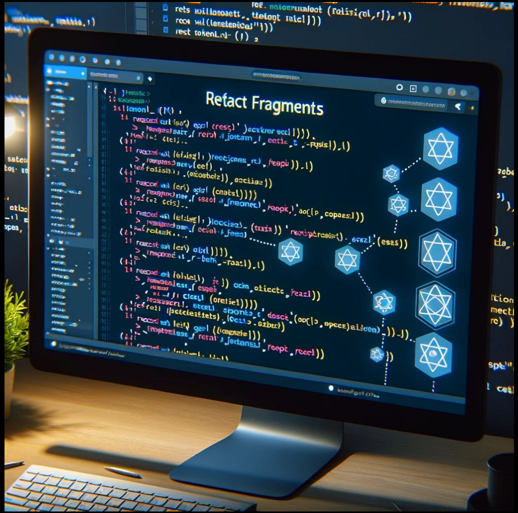
Fragments

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: `<>...</>`



React

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

Conditional Rendering

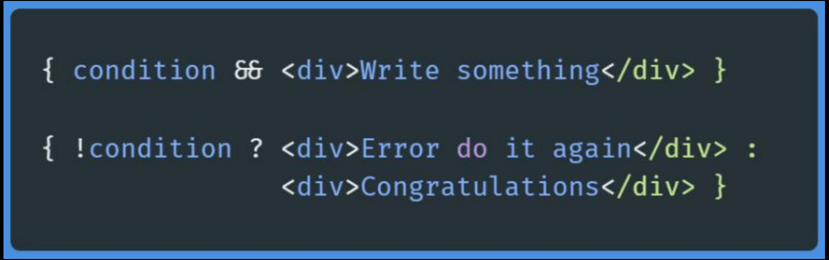
- Displaying content based on certain conditions.
- Allows for dynamic user interfaces.

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.

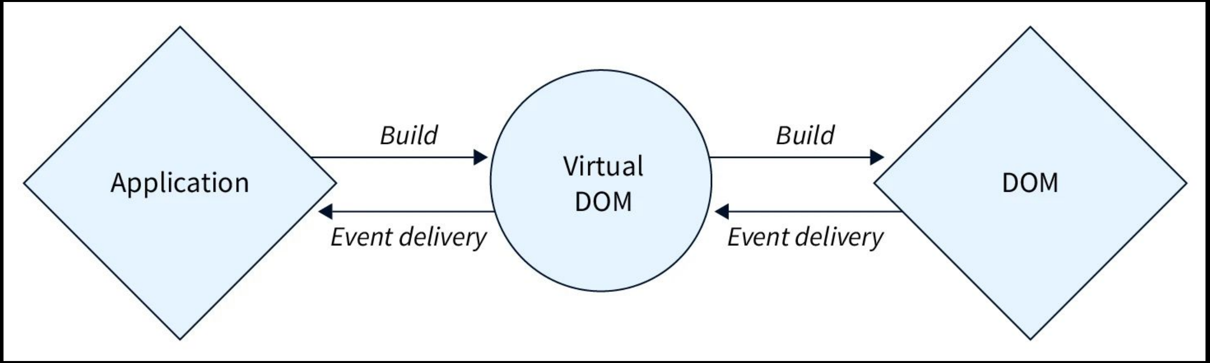
Benefits

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



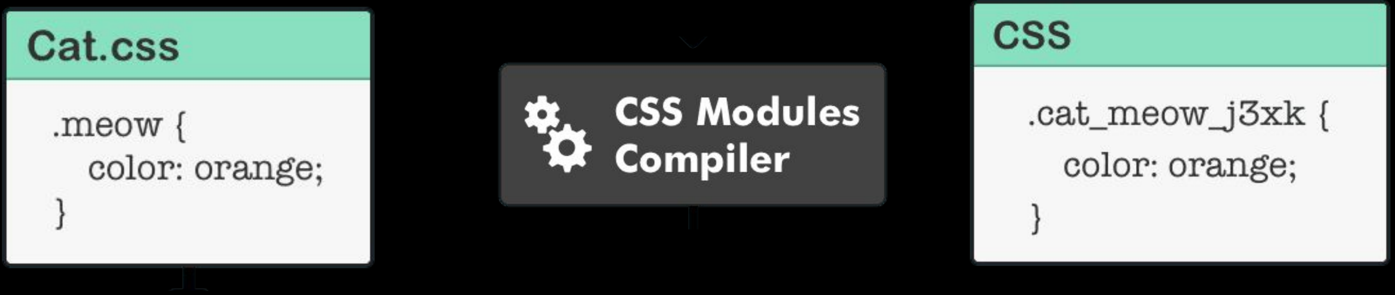
```
{ condition && <div>Write something</div> }

{ !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.

CSS Modules



```
.meow {
    color: orange;
}
```

```
.cat_meow_j3xk {
    color: orange;
}
```

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>
  );
}
```

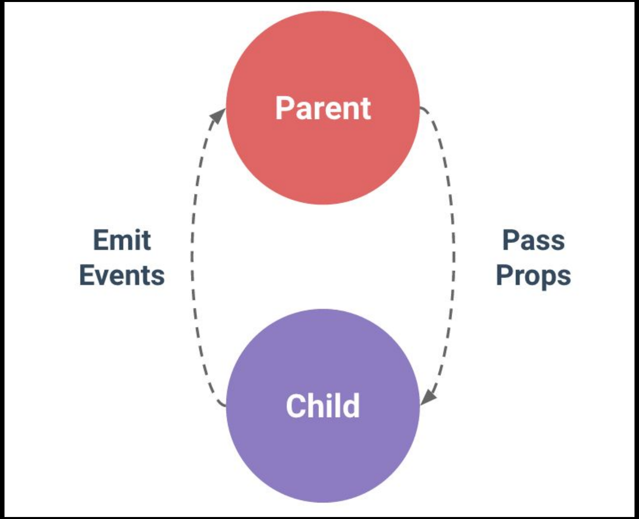
```jsx
<Container>
  <h1>Welcome to My App</h1>
  <p>This content is passed as children to the
  Container component.</p>
</Container>
```

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.

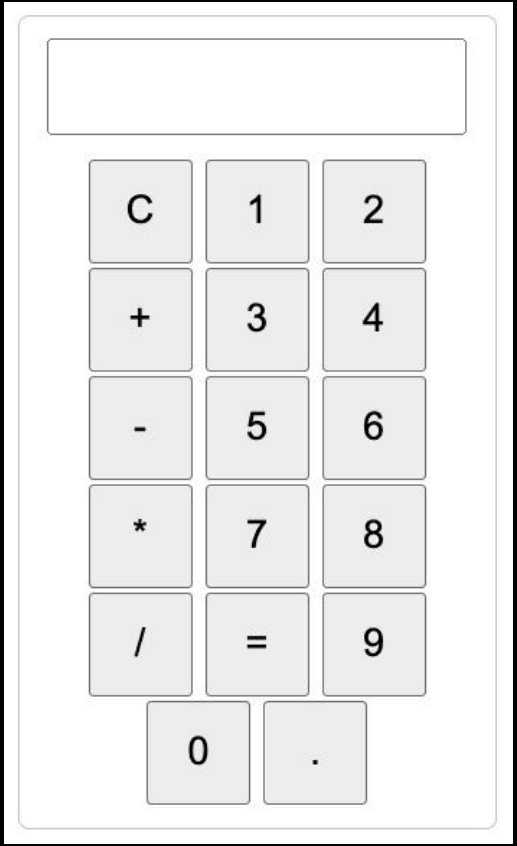
Passing Functions via Props

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} />
```



Project Calculator


C
1
2
+
3
4
-
5
6
*
7
8
/
=
9
0
.