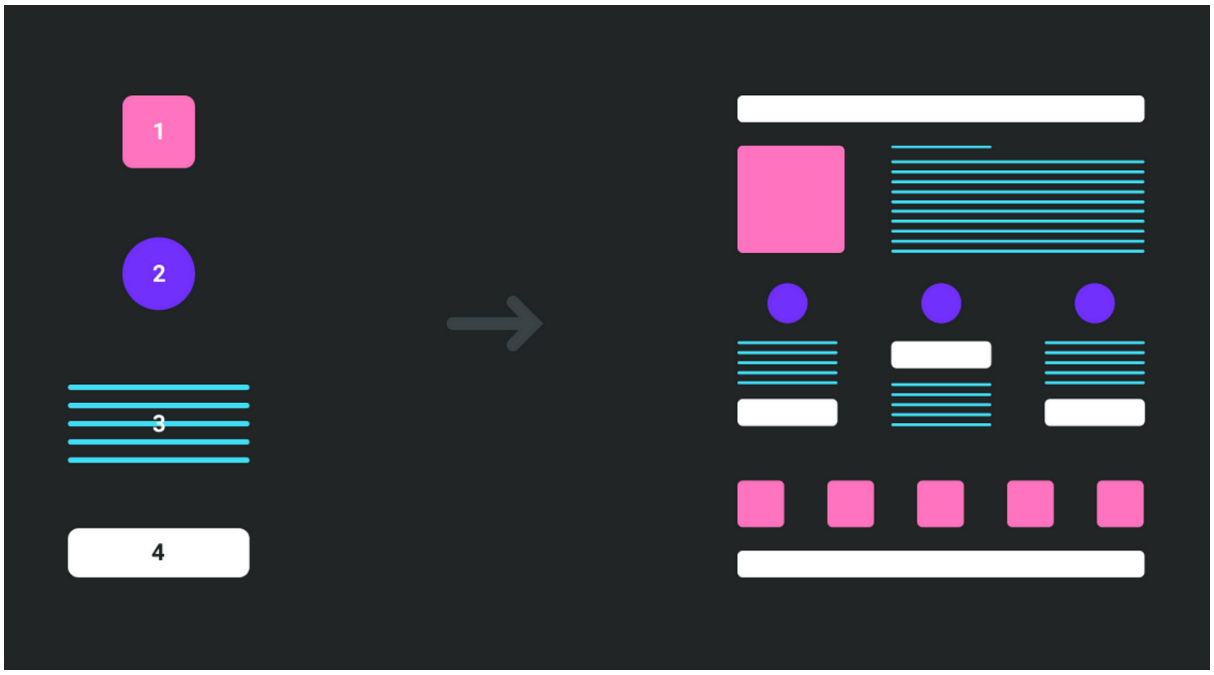
React Components

Table of Contents:

What are Components?

- React components are independent, reusable building blocks in a React application that define what gets displayed on the UI.
- They accept inputs called props and return React elements describing the UI.
- Components let you split the UI into independent, reusable pieces, and think about each piece in isolation.

💡 Conceptually, components are like JavaScript functions. They accept arbitrary inputs (called “props”) and return React elements describing what should appear on the screen.



Two Types of Components

1. **Functional Components**

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}
```

- This function is a valid React component because it accepts a single “props” (which stands for properties) object argument with data and returns a React element.
- We call such components “function components” because they are JavaScript functions.
- **Simpler Syntax:** Ideal for small and reusable components.
- **Performance:** Generally faster since they don’t require a ‘this’ keyword.

2. **Class Components**

   You can also use an ES6 class to define a component:

```
class Welcome extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
}
```

The above two components are equivalent from React's point of view.

Composing Components

Components can refer to other components in their output. This lets us use the same component abstraction for any level of detail.

For example, we can create an `App` component that renders `Welcome` many times:

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}

function App() {
  return (
    <div>
            <Welcome name="Sara" />
            <Welcome name="Cahal" />
      <Welcome name="Edite" />
    </div>);
}
```