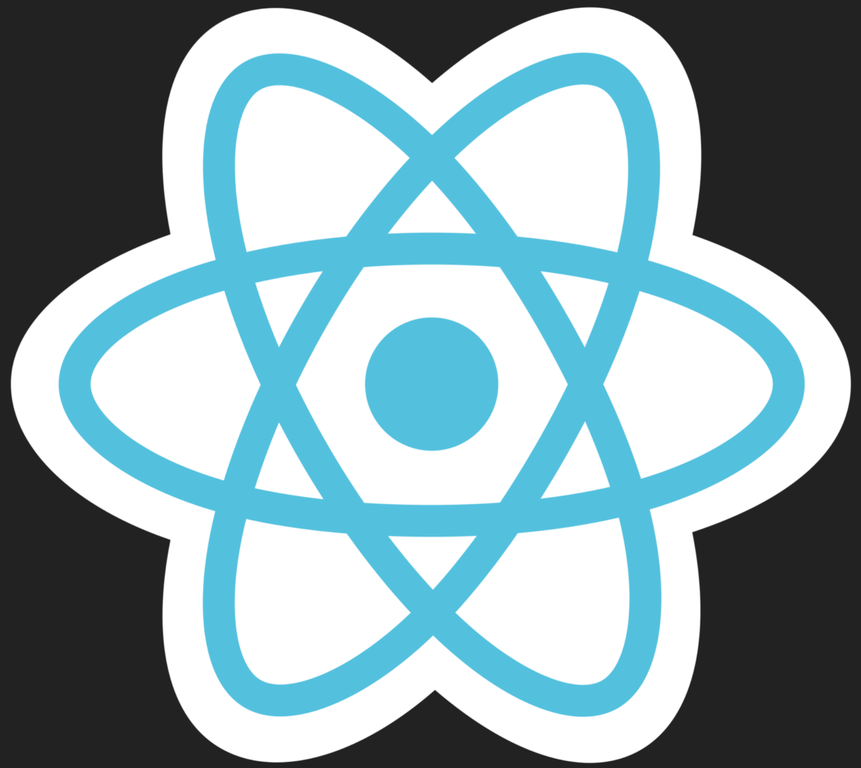


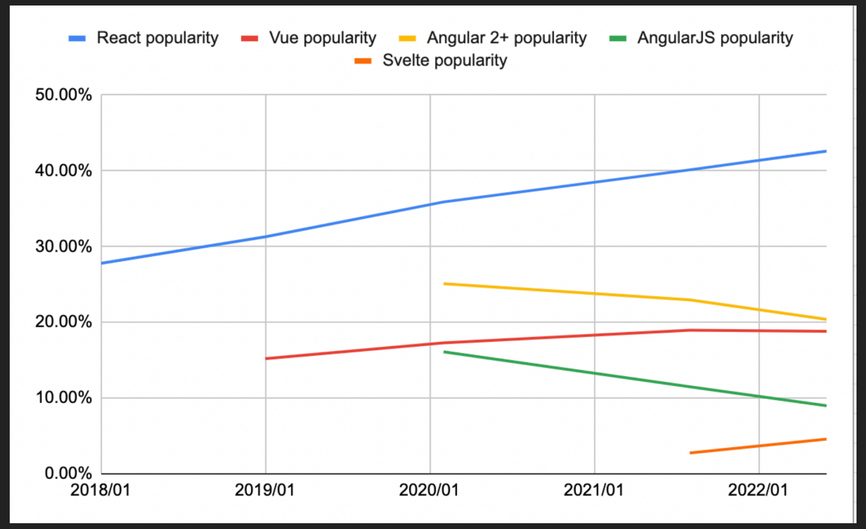
REACT

DESIGN PATTERNS

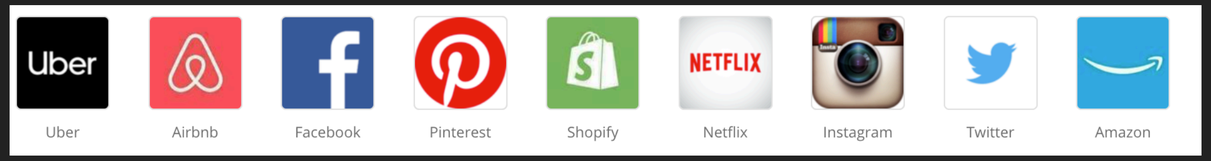
BRIEF INTRO TO REACT

WHAT IS REACT?

- Open source Javascript library developed by Facebook, built to create single page web applications with ease using small building blocks called "Components"
- It is becoming more and more popular by time and is adapted by a very huge community



React popularity according to StackOverflow statistics -stackoverflow.com



11269 companies reportedly use **React** in their tech stacks, including **Uber, Airbnb,** and **Facebook**. -stackshare.io

WHAT IS REACT?

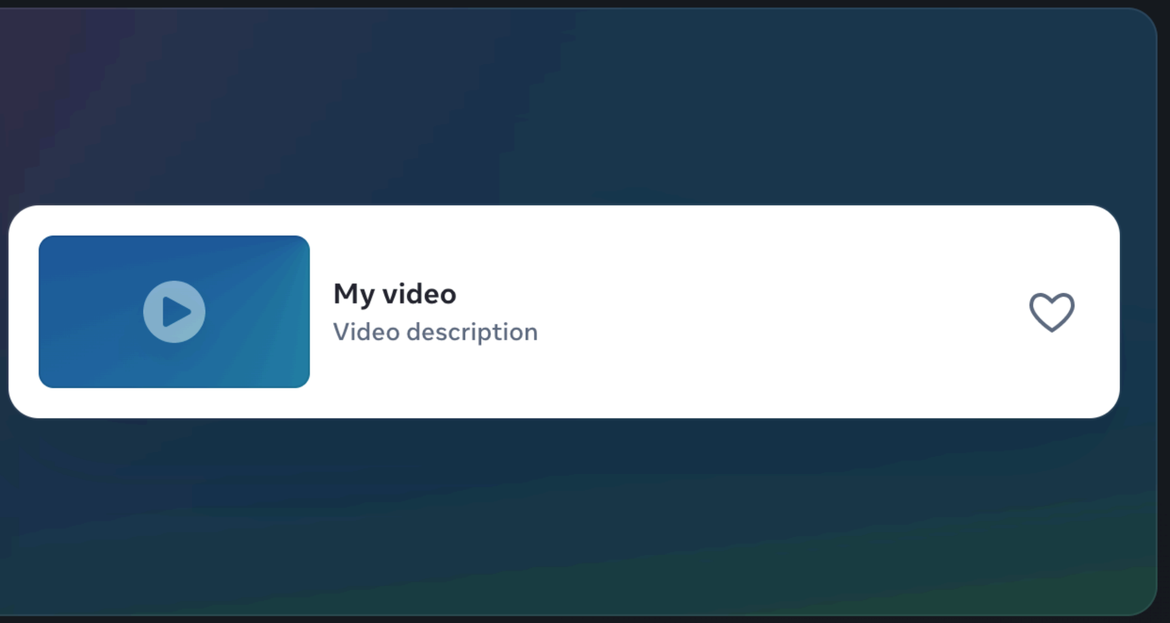
- Common misconception: mistaking React as a framework not a library
  - It could be used with other frameworks like: Angular, Vue, Ember, etc.
- React's Virtual DOM based mechanism
  - Reloads element (components) instead of building the whole DOM on small changes

BRIEF INTRO TO REACT

React lets you build user interfaces out of individual pieces called components.
Create our own React components like Thumbnail, LikeButton, and Video.
Then combine them into entire screens, pages, and apps.

Video.js

```
function Video({ video }) {
  return (
    <div>
      <Thumbnail video={video} />
      <a href={video.url}>
        <h3>{video.title}</h3>
        <p>{video.description}</p>
      </a>
      <LikeButton video={video} />
    </div>
  );
}
```



DESIGN PATTERNS

HIGHER ORDER COMP. (HOC)

PROBLEM -> UTILIZING SAME LOGIC IN VARIOUS LOCATIONS SUCH AS:

- Components that use third party subscription data
- Enhance various card views with the same design elements, like card wrappers, shadows, borders etc.
- Components that requires logged in user
- Infinite scroll containers with different data
- Etc.

DESIGN & IMPLEMENTATION

- Takes a component as an argument and returns another component "with" additional data and/or functionality

```
// HOC creator function
export const withHigherOrderComponent = (DecoratedComponent) => {
  const HOC = () => {
    return (
      <div
        style={{
          padding: "10px",
          backgroundColor: "blue",
        }}
      >
        <DecoratedComponent />
      </div>
    );
  };
  return HOC;
};

// usage
const PageWithPadding = withHigherOrderComponent(Page);
```

CHECK HOC IMPLEMENTATION ON GITHUB

COMPOUND COMP. PATTERN

PROBLEMS -> DEPENDENCIES BETWEEN COMPONENTS

- Child components are standalone components, while they have no use outside their parent components
- Child components tends to use their parent component states and hooks which usually leads to prop drilling
- This pattern encapsulates the logic in the parent and ensures that children are strictly coupled to their parents

THE ANTI PATTERN

```
// Child component
const Option = ({ children, onClick, active }) => {
  return (
    <div
      style={{
        ...optionStyle,
        backgroundColor: active && "#61dafb",
      }}
      onClick={onClick}
    >
      <p>{children}</p>
    </div>
  );
};

// Parent component
const Select = ({ options }) => {
  const [selectedOption, setSelectedOption] = useState(null);

  return options.map(({ key, value }) => (
    <Option
      active={selectedOption === key}
      onClick={() => setSelectedOption(key)}
    >
      {value}
    </Option>
  ));
};
```

```
function App() {
  return (
    <Select
      options={[
        { key: "OLIVER", value: "Oliver" },
        { key: "EVE", value: "Eve" },
      ]}
    />
  );
};
```

CHECK COMPOUND COMPONENT IMPLEMENTATION ON GITHUB

FACTORY PATTERN

PROBLEMS

- React page components tends to grow and gets more complex by time and starts to be unmaintainable
- Large components usually violates the "Single Responsibility Principle"
- Children components Render Performance issues

HOME PAGE EXAMPLE

- Basic home page with dashboard and a side panel

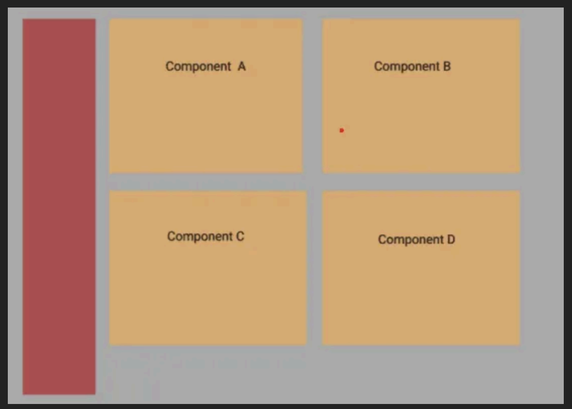
```
export const Home = () => {
  return (
    <>
      {/* Header */}
      {/* Side Panel*/}
      {/* Other components... */}
      {/* Dashboard */}
      {USER_DATA.items.map((card) => {
        switch (card.type) {
          case "A":
            return <A />;
          case "B":
            return <B />;
          case "C":
            return <C />;
          case "D":
            return <D />;
          default:
            return null;
        }
      })}
    {/* Footer */}
    </>
  );
};
```

```
function A() {
  return <div>Type A Component</div>;
}

function B() {
  return <div>Type B Component</div>;
}

function C() {
  return <div>Type C Component</div>;
}

function D() {
  return <div>Type D Component</div>;
}
```



THE ANTI PATTERN

- By time the dashboard will have more card types: A,B,C,D,E,F,G,H, ...
- Any re-render caused by the side panel would re render all other components in the home page
- Thus this page would be laggy because of all the re-renders
- Maintainability is now an issue 😟

CHECK FACTORY PATTERN IMPLEMENTATION ON GITHUB

CONTAINER & PRESENTATIONAL COMP. PATTERN

PROBLEMS

- Duplicate code caused by lack of modularity
- Components are drowned with Logic and UI
- Non-opinionated nature of React

CHECK CONTAINER AND PRESENTAIONAL COMPNENTS IMPLEMENTATION ON GITHUB

USEFUL LINKS

- This session's Project on Github

- Building blocks of React
- React docs
- React common anti patterns
- 10 React anti patterns you should avoid