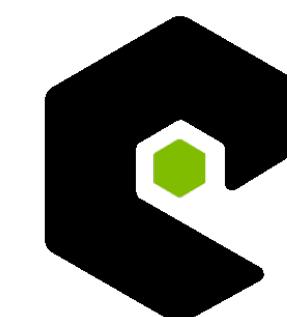
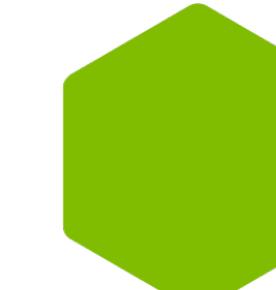
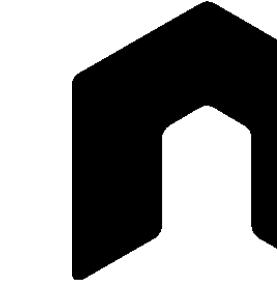
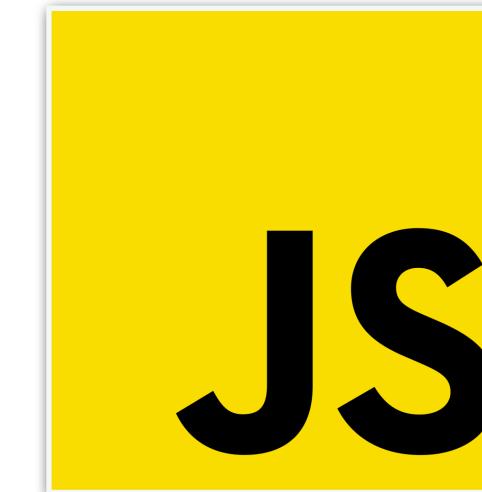
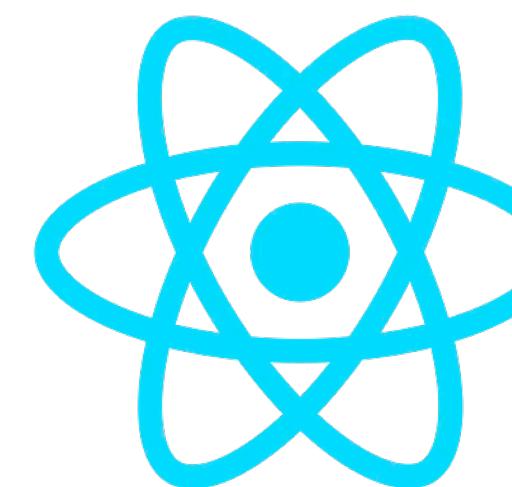

JAVASCRIPT

REACT FRONT-END PART 1

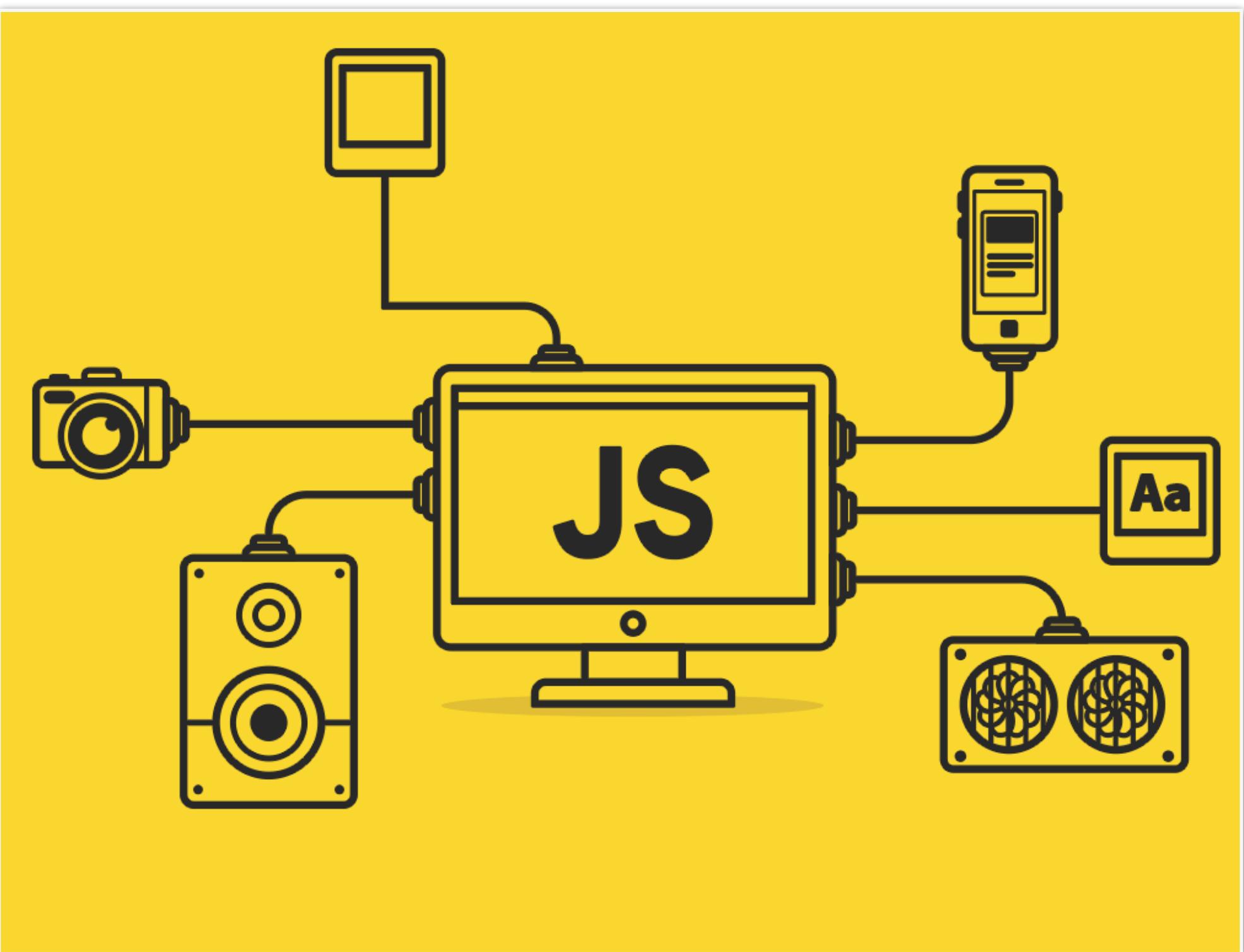
Anirach Mingkhwan

Anirach.m@fitm.kmutnb.ac.th

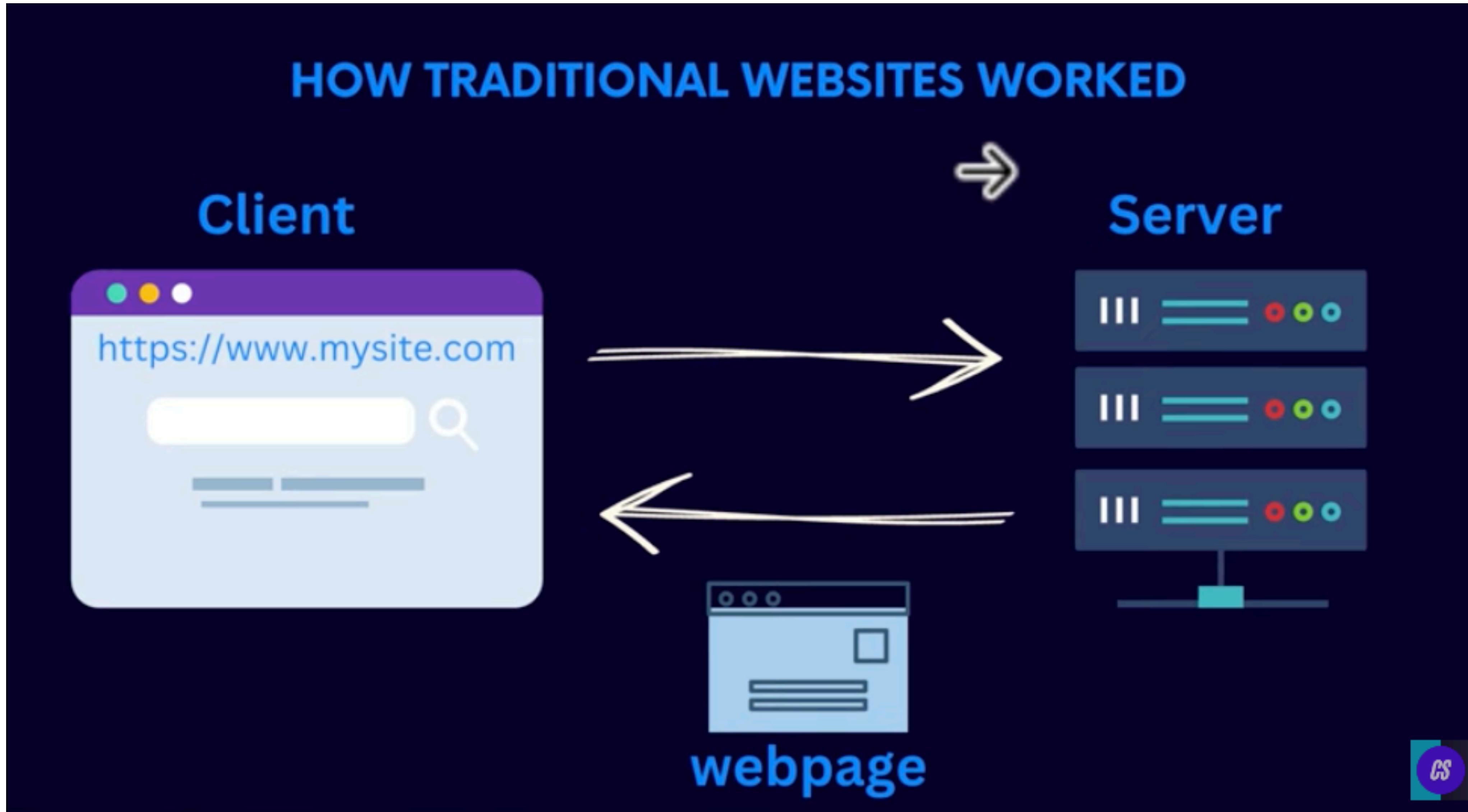


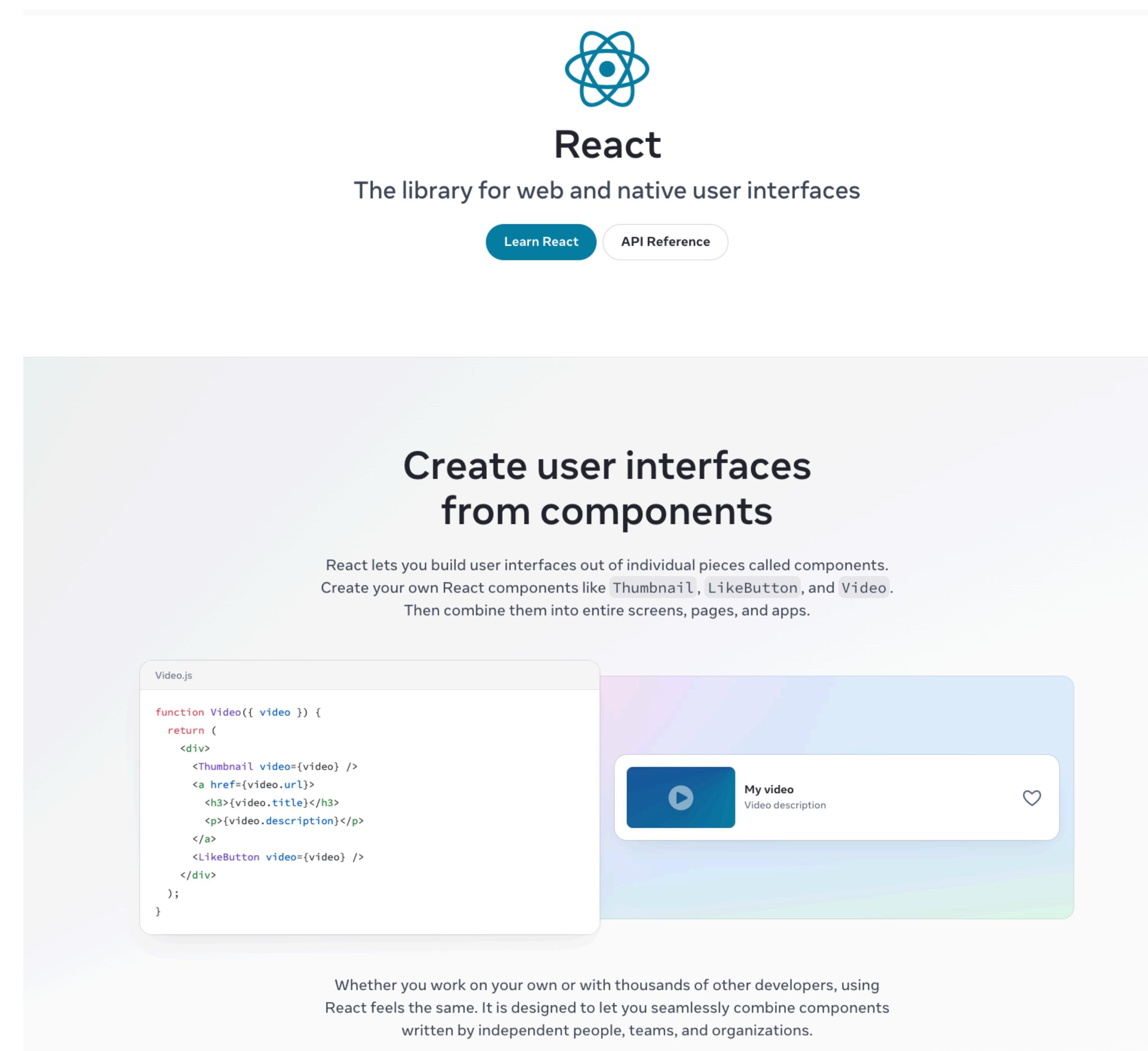
OUTLINE

- React Introduction
- Fundamentals of React
- Create React Project
- React props/state
- Thinking React



REACT INTRO





The image shows the official React website homepage. At the top center is the React logo, which is a stylized blue atom or orbital symbol. Below it is the word "React" in a bold, black, sans-serif font. Underneath "React" is the tagline "The library for web and native user interfaces". There are two buttons at the bottom of this header section: a teal button labeled "Learn React" and a white button labeled "API Reference".

Create user interfaces from components

React lets you build user interfaces out of individual pieces called components. Create your 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>
  );
}
```

My video
Video description

Whether you work on your own or with thousands of other developers, using React feels the same. It is designed to let you seamlessly combine components written by independent people, teams, and organizations.

<https://react.dev/>

HOW REACT APPLICATIONS WORK ?

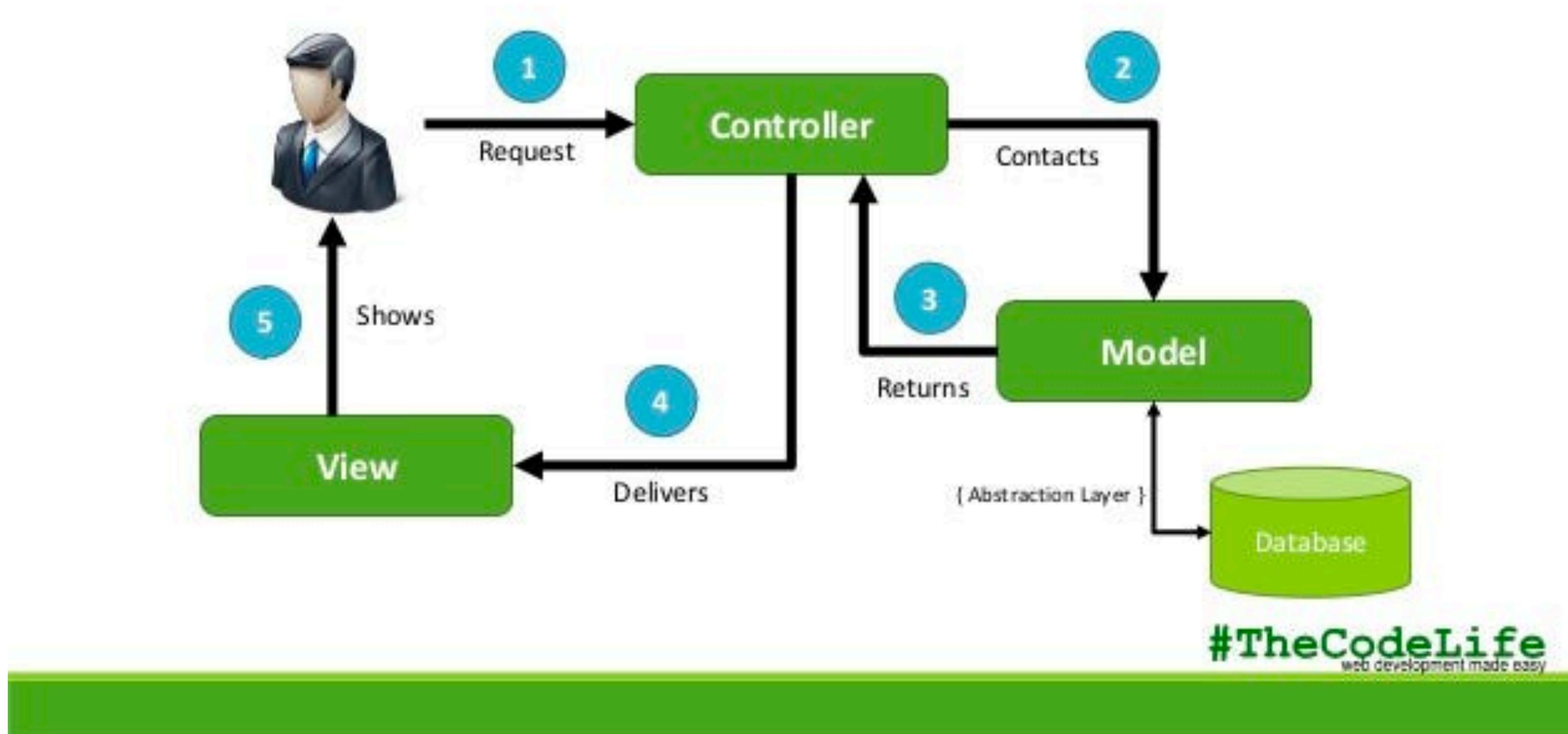
Client Side Rendering



Single Page Application
(SPA)

-
- React is an open-source JavaScript library for creating user interfaces.
 - It is a library, not a framework.
 - Used for handling the "view" layer of the application.
 - Allow us to create large web applications, that change data without reloading the webpage.
-

How it works

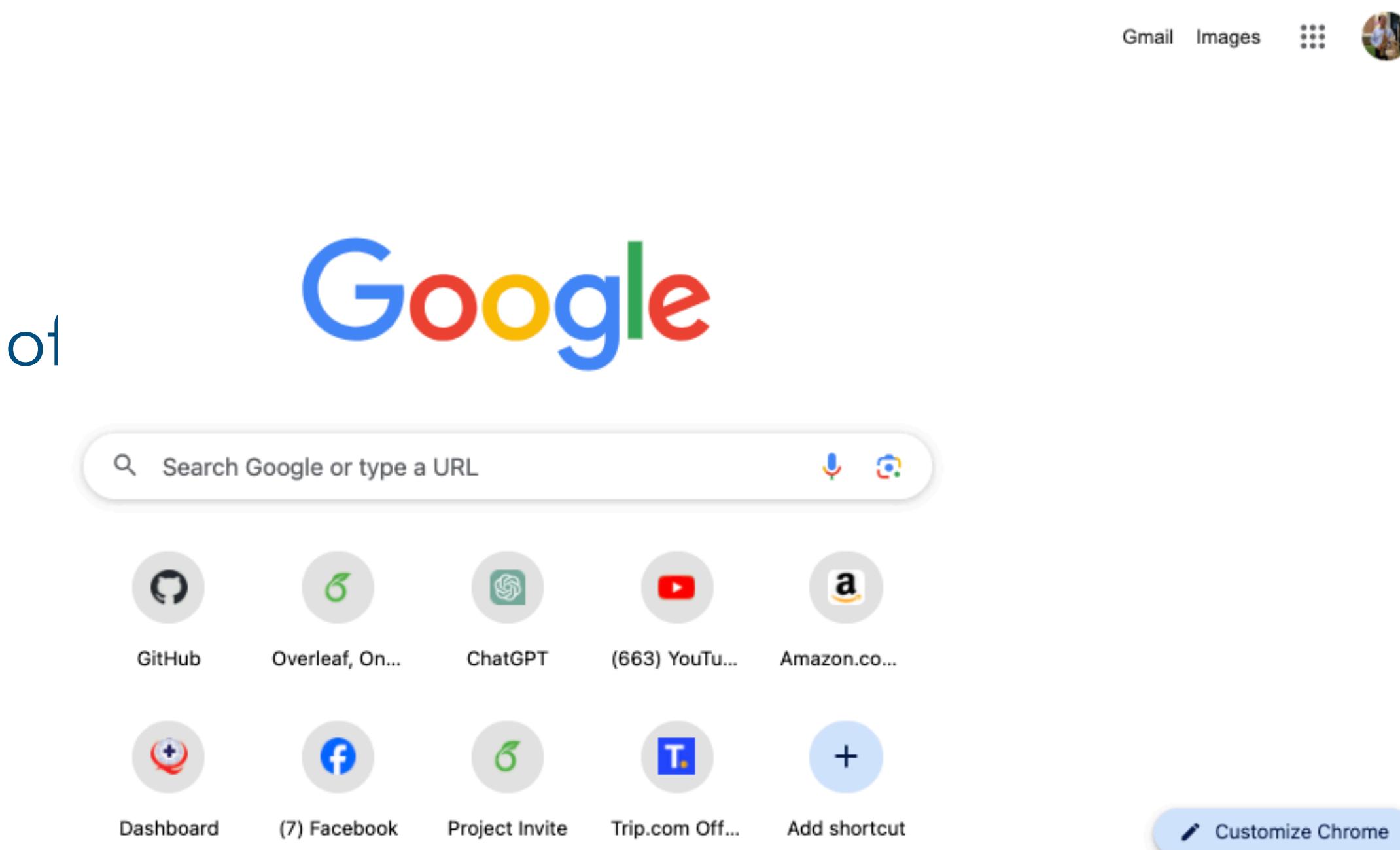


Why you should learn React?

-
- Developed by Facebook.
 - JSX.
 - In-demand skill.
 - Could be combined with any tech stack.
 - Mobile application development with React Native.
-

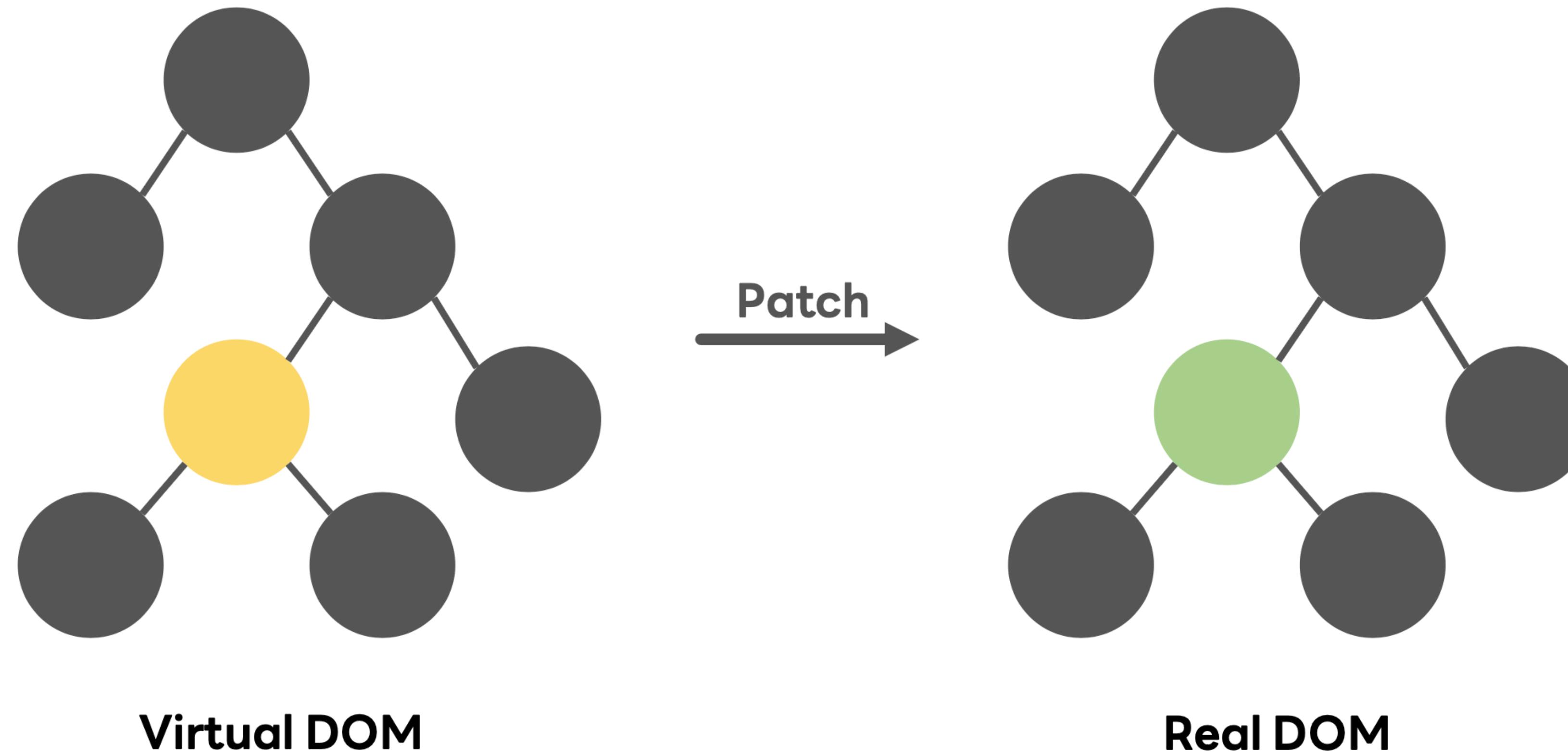
FUNDAMENTALS OF REACT

- React is component-based.
- React application is a combination of multiple components.
- Components are re-useable.



Virtual DOM & Real DOM

Slide # 12



DOM = Document Object Model

● Component

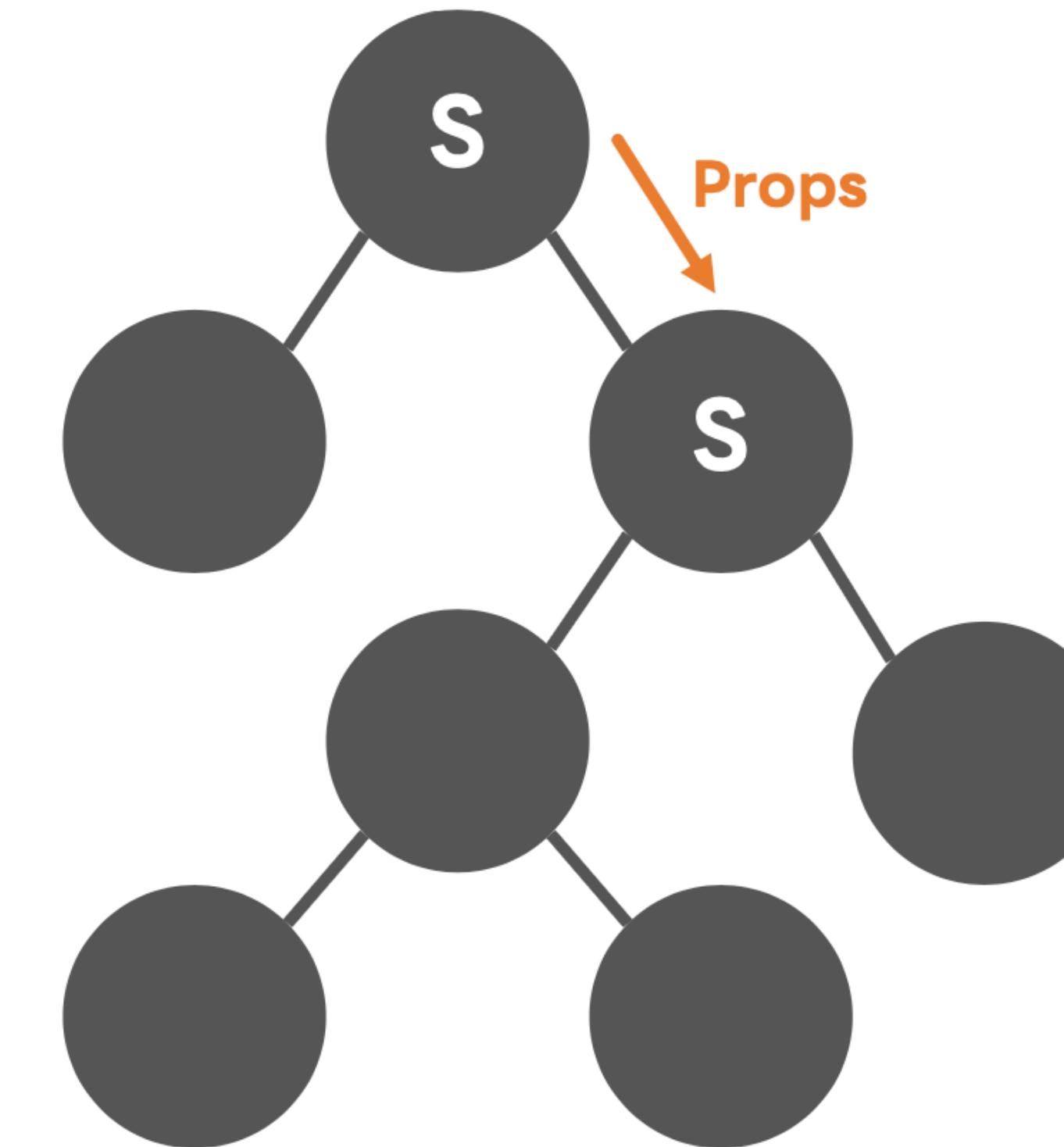
ส่วนประกอบต่างๆ ภายในเว็บไซต์
เรามองทุกสิ่งเป็น Component
ถ้าอ้างอิงตามภาพคือ ลูกของกลม

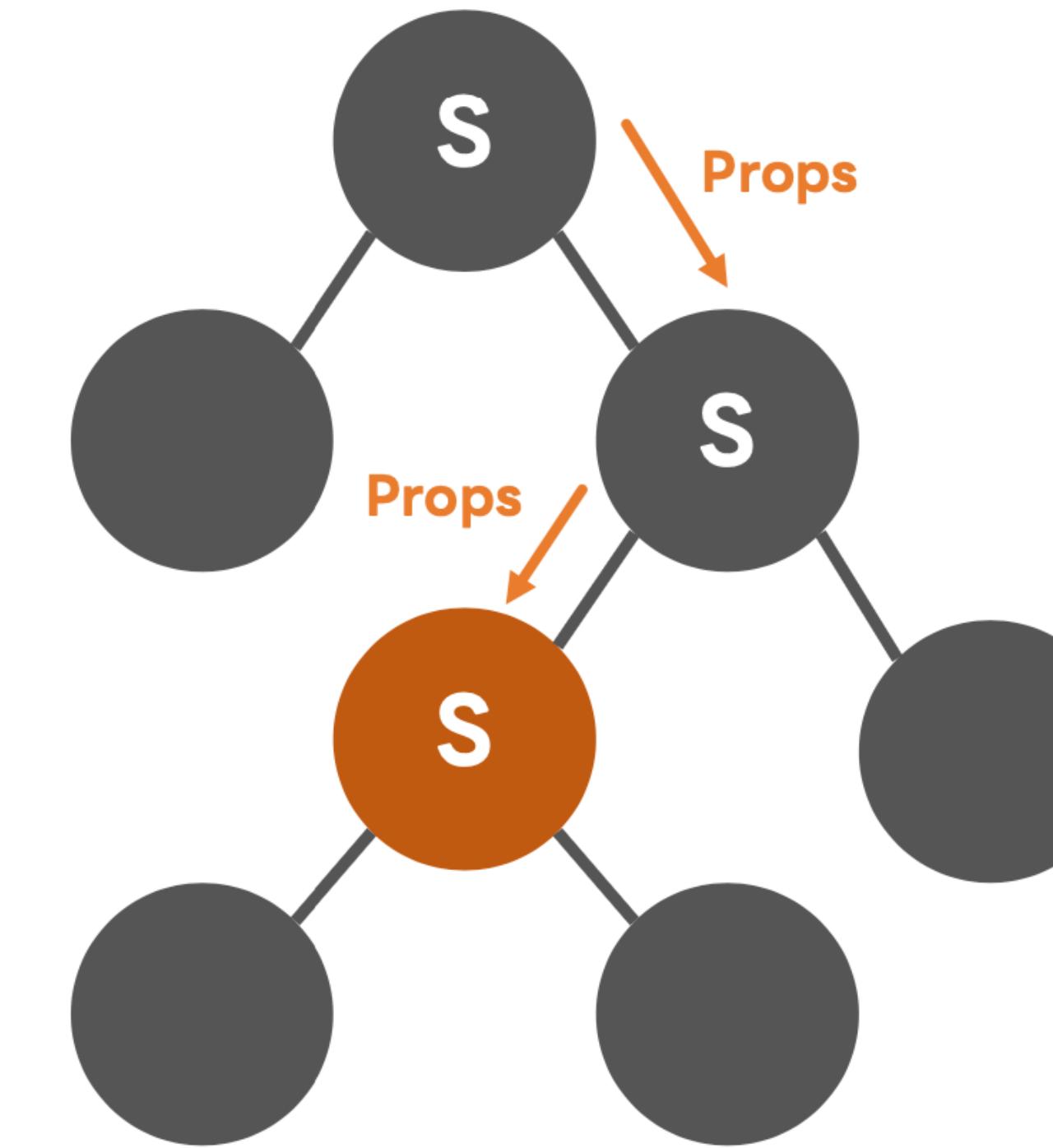
s State

ข้อมูลที่อยู่ใน Component โดย
ข้อมูลจะถูกกำหนดเน้นการตามลำดับ

Props

ข้อมูลที่ถูกส่งระหว่าง Components
ซึ่งข้อมูลที่ถูกส่งคือ State ของ
Component จากต้นทางซึ่งจะ
ถูกต่ายกอดจนกว่าจะถึงจุดจบ





Passing values between components
ส่งผ่านค่าระหว่างส่วนประกอบ

CREATE REACT PROJECT

Create GitHUB Repository

The screenshot shows a GitHub repository page for 'ReactFrontEnd'. The repository is private, created by 'Anirach' on March 1, 2024. It has 1 branch ('main') and 0 tags. The repository contains a single commit from 'Anirach' titled 'Initial commit' containing a 'README.md' file. A modal window is open over the repository details, showing cloning options. The 'Clone' section includes links for 'HTTPS', 'SSH', and 'GitHub CLI', with the 'HTTPS' link highlighted. The URL is <https://github.com/Anirach/ReactFrontEnd.git>. Other options include 'Open with GitHub Desktop' and 'Download ZIP'. To the right of the modal, the repository's 'About' section is visible, showing it's a 'React Front-End' project with 0 stars, 1 watching, and 0 forks. It also lists 'Readme', 'Activity', and links to 'Create a new release' and 'Publish your first package'.

Anirach / ReactFrontEnd

Type / to search

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

ReactFrontEnd Private

Unwatch 1 Fork 0 S

main 1 Branch 0 Tags Go to file Add file Code About

Anirach Initial commit

README.md Initial commit

README

ReactFrontEnd

React Front-End

Clone Local Codespaces

HTTPS SSH GitHub CLI

<https://github.com/Anirach/ReactFrontEnd.git>

Clone using the web URL.

Open with GitHub Desktop

Download ZIP

Readme Activity 0 stars 1 watching 0 forks

Releases No releases published Create a new release

Packages No packages published Publish your first package



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SQL CONSOLE

```
> npm create vite
✓ Project name: ... REACT-FrontEnd
✓ Package name: ... react-frontend
? Select a framework: > – Use arrow-keys. Return to submit.
  Vanilla
  Vue
> React
  Preact
  Lit
  Svelte
  Solid
  Qwik
  Others
```

The screenshot shows the VS Code interface with the 'TERMINAL' tab selected. The terminal window displays the following command-line interaction:

```
> npm create vite
✓ Project name: ... REACT-FrontEnd
✓ Package name: ... react-frontend
✓ Select a framework: > React
? Select a variant: > - Use arrow-keys. Return to submit.
    TypeScript
    TypeScript + SWC
> JavaScript
    JavaScript + SWC
```

The 'JavaScript' option is highlighted with a blue underline.

The screenshot shows the VS Code interface with the 'TERMINAL' tab selected. The terminal window displays the command `npm create vite` followed by a series of green checkmarks indicating configuration choices: Project name (REACT-FrontEnd), Package name (react-frontend), Framework (React), and Variant (JavaScript). Below this, a message says 'Scaffolding project in /Users/anirachmingkhwan/Code/ReactFrontEnd/REACT-FrontEnd...'. The next line, 'Done. Now run:', is followed by three terminal commands: `cd REACT-FrontEnd`, `npm install`, and `npm run dev`. At the bottom, a summary of the process is shown: `took 2m 49s` and `system`.

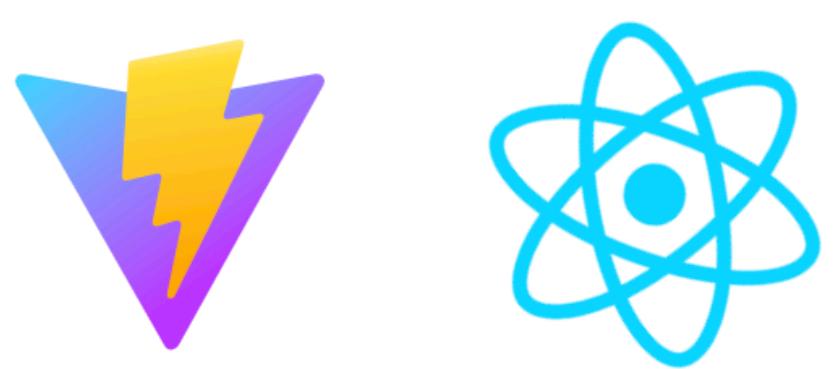
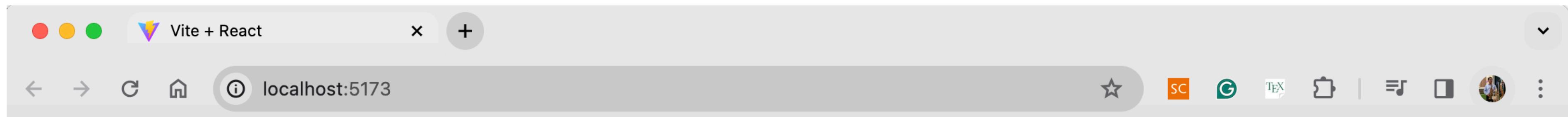
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SQL CONSOLE ... > zsh

> npm create vite
✓ Project name: ... REACT-FrontEnd
✓ Package name: ... react-frontend
✓ Select a framework: > React
✓ Select a variant: > JavaScript

Scaffolding project in /Users/anirachmingkhwan/Code/ReactFrontEnd/REACT-FrontEnd...
Done. Now run:

cd REACT-FrontEnd
npm install
npm run dev

> ~ /Code/ReactFrontEnd > on main ?1 ..... took 2m 49s < system
```



Vite + React

count is 0

Edit src/App.jsx and save to test HMR

Click on the Vite and React logos to learn more

STRIP TO BASIC

The screenshot shows a code editor interface with the following details:

- EXPLORER:** On the left, the project structure is displayed under the folder "REACTFRONTEND". It includes "node_modules", "public", "src" (with "assets" containing "Hello.jsx", "App.jsx", and "main.jsx"), ".eslintrc.cjs", ".gitignore", "index.html" (which is currently selected), "package-lock.json", "package.json", "README.md", and "vite.config.js". There are 2 untracked files indicated by a blue circle with a '2'.
- EDITOR:** The main area shows the content of "index.html". The code is as follows:

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>React Project</title>
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.jsx"></script>
  </body>
</html>
```

The screenshot shows a VS Code interface with the following details:

- EXPLORER:** Shows the project structure under "REACTFRONTEND".
 - node_modules
 - public
 - src
 - assets
 - Hello.jsx
 - App.jsx
 - main.jsx
- SEARCH:** Search bar at the top right containing "ReactFrontEnd".
- EDITOR:** The main editor area displays the content of "main.jsx".

```
src > main.jsx
1 import React from 'react'
2 import ReactDOM from 'react-dom/client'
3 import App from './App.jsx'
4
5 ReactDOM.createRoot(document.getElementById('root')).render(
6   <React.StrictMode>
7     <App />
8   </React.StrictMode>,
9 )
10
```
- STATUS BAR:** Shows tabs for main.jsx, index.html, package.json, and COMMIT_EDITMSG.

The screenshot shows a code editor interface with the title bar "ReactFrontEnd". The left sidebar is labeled "EXPLORER" and displays the project structure:

- REACTFRONTEND
 - node_modules
 - public
 - src
 - assets
 - Hello.jsx
 - App.jsx
 - main.jsx
 - .eslintrc.cjs
 - .gitignore
 - index.html
 - package-lock.json
 - package.json
 - README.md
 - vite.config.js

Two specific items are highlighted with blue circles and numbers:

- Item 1: "Hello.jsx" in the "assets" folder.
- Item 2: "App.jsx" in the "src" folder.

The main editor area shows the content of "App.jsx":

```
src > App.jsx > ...
1 import Hello from './assets/Hello';
2
3 function App() {
4   return (
5     <div className='App'>
6       <Hello />
7     </div>
8   );
9 }
10
11 export default App;
```

The screenshot shows a code editor interface with the title bar "ReactFrontEnd". The left sidebar is labeled "EXPLORER" and contains a tree view of a project structure:

- REACTFRONTEND
 - node_modules
 - public
 - src
 - assets
 - Hello.jsx
 - App.jsx
 - main.jsx
 - .eslintrc.cjs
 - .gitignore
 - index.html
 - package-lock.json
 - package.json
 - README.md
 - vite.config.js

Two numbered callouts are present:

- Callout 1 points to the "Hello.jsx" file in the Explorer sidebar.
- Callout 2 points to the "src/assets" folder in the Explorer sidebar.

The main editor area displays the content of the "Hello.jsx" file:

```
src > assets > Hello.jsx > ...
1 function Hello() {
2   return (
3     <div>
4       <h1>Hello World</h1>
5     </div>
6   );
7 }
8
9 export default Hello;
```

REACT - PROPS

```
src > App.jsx > ...
1 import Contact from './assets/Contact';
2 import Hello from './assets/Hello';
3
4
5 function App() {
6     const helloData = [
7         { name: 'Anirach', message: 'Hi there' },
8         { name: 'Tom', message: 'Hello..' }
9     ];
10
11    return (
12        <div >
13            {helloData.map((data, index) => (
14                <Hello key={index} name={data.name} message={data.message} />
15            ))}
16            <Contact email="Anirach@gmail.com" phone ="0817320731" />
17        </div>
18    );
19
20
21
22 export default App;
23
```

The image shows a code editor interface with two main panes. On the left is the Explorer pane, which displays the file structure of a React frontend project named 'REACTFRONTEND'. The structure includes 'node_modules', 'public', 'src' (which contains 'assets' and files like 'Contact.jsx', 'Hello.jsx', 'App.jsx', 'main.jsx'), and configuration files like '.eslintrc.cjs', '.gitignore', 'index.html', 'package-lock.json', 'package.json', 'README.md', and 'vite.config.js'. A blue circular icon with the number '1' is visible next to the 'src' folder. The right pane is the Editor pane, showing the content of 'Contact.jsx'. The code defines a functional component 'Contact' that returns a JSX structure with an H2 and a message. It also includes prop types for 'email' and 'phone'.

```
src > assets > Contact.jsx > ...
1 import PropTypes from 'prop-types';
2
3 export default function Contact(props) {
4     return (
5         <div>
6             <h2>{props.email} {props.phone}</h2>
7             <p>Send me a message!</p>
8         </div>
9     );
10}
11
12 Contact.propTypes = {
13     email: PropTypes.string.isRequired,
14     phone: PropTypes.string.isRequired
15};
16
17
18
```

The screenshot shows the VS Code interface with the following details:

- EXPLORER** pane on the left, showing the project structure:

 - REACTFRONTEND** folder:
 - node_modules
 - public
 - src** folder:
 - assets** folder:
 - Contact.jsx
 - Hello.jsx**
 - App.jsx
 - main.jsx
 - .eslintrc.cjs
 - .gitignore
 - index.html
 - package-lock.json
 - package.json
 - README.md
 - vite.config.js

- Editor** pane on the right, showing the **Hello.jsx** file content:

```
src > assets > Hello.jsx > ...
1
2 const Hello = ({ message, name }) => {
3   console.log({ message, name });
4   return (
5     <div>
6       <h1>
7         {message} {name}
8       </h1>
9     </div>
10    );
11
12 import PropTypes from 'prop-types';
13
14 Hello.propTypes = {
15   message: PropTypes.string.isRequired,
16   name: PropTypes.string.isRequired,
17 };
18
19
20 export default Hello;
21
```

```
src > ⚛ App.jsx > ...  
1 import Contact from './assets/Contact';  
2 import Counter from './assets/Counter';  
3 import Hello from './assets/Hello';  
4  
5  
6 function App() {  
7     const helloData = [  
8         { name: 'Anirach', message: 'Hi there' },  
9         { name: 'Tom', message: 'Hello..' }  
10    ];  
11  
12    return (  
13        <div className='App'>  
14            < Counter />  
15            {helloData.map((data, index) => (  
16                <Hello key={index} name={data.name} message={data.message} />  
17            ))}  
18  
19            <Contact email="Anirach@gmail.com" phone ="0817320731" />  
20        </div>  
21    );  
22}  
23  
24 export default App;  
25
```

A screenshot of a browser's developer tools, specifically the Console tab. The main content area displays three lines of text:

Hi there Anirach

Hello.. Tom

Anirach@gmail.com 0817320731

Below this, a message reads "Send me a message!"

The developer tools interface includes tabs for Elements, Console, Sources, Network, Performance, and Memory. The Console tab is active. On the left, a sidebar shows log levels: top (29 messages), user messages (26), errors (4), warnings (0), info (13), and verbose (12). A message from Vite indicates a hot update to Counter.jsx. The bottom right corner of the browser window shows a small green icon.

REACT - STATE

-
- **Props are passed to the component.**
 - **State is contained inside the component.**
 - **Props are immutable.**
 - **State can be changed.**
 - **Props are like parameters passed to a function.**
 - **State is like a variable declared in function body.**
-

The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER** pane on the left, showing the project structure:

 - REACTFRONTEND** folder:
 - node_modules
 - public
 - src
 - assets
 - Contact.jsx
 - Counter.jsx**
 - Hello.jsx
 - App.jsx
 - main.jsx
 - .eslintrc.cjs
 - .gitignore
 - index.html
 - package-lock.json
 - package.json
 - README.md
 - vite.config.js

- Editor** pane on the right, showing the content of **Counter.jsx**:

```
src > assets > Counter.jsx > ...
1 import { useState } from 'react';
2
3 export default function Counter() {
4     const [count, setCount] = useState(0);
5     function increment() {
6         setCount(count + 1);
7     }
8     function decrement() {
9         setCount(count - 1);
10    }
11
12    return (
13        <div>
14            <h1>Count value is: {count}</h1>
15            <button onClick={increment}>increment</button>
16            <button onClick={decrement}>decrement</button>
17        </div>
18    );
19}
20
```

The screenshot shows a code editor interface with the following details:

- EXPLORER View:** Shows the project structure under "REACTFRONTEND".
 - node_modules
 - public
 - src
 - assets
 - Contact.jsx
 - Counter.jsx
 - Hello.jsx
 - App.jsx
 - main.jsx
 - .eslintrc.cjs
 - .gitignore
 - index.html
 - package-lock.json
 - package.json
 - README.md
 - vite.config.js
- Code Editor View:** Displays the content of the App.jsx file.

```
src > App.jsx > App
1 import Contact from './assets/Contact';
2 import Counter from './assets/Counter';
3 import Hello from './assets/Hello';
4
5
6 function App() {
7   const helloData = [
8     { name: 'Anirach', message: 'Hi there' },
9     { name: 'Tom', message: 'Hello..' }
10 ];
11
12   return (
13     <div className='App'>
14       <Counter />
15       {helloData.map((data, index) => (
16         <Hello key={index} name={data.name} message={data.message} />
17       ))}
18       <Contact email="Anirach@gmail.com" phone ="0817320731" />
19     </div>
20   );
21 }
22
23
24 export default App;
```

The screenshot shows a development environment with a browser window and a sidebar.

Sidebar (Left):

- File icon: Counter.jsx X
- Search icon
- Link icon
- Project icon
- Git icon
- File icon
- Terminal icon
- Lightbulb icon

Browser Window (Right):

- Address bar: localhost:5174
- Dimensions: Responsive ▾ 624 x 411 100% ▾ No throttling ▾
- Content area:
 - Count value is: 0**
 - Buttons: **increment** **decrement**
 - Hi there Anirach**
 - Hello.. Tom**
 - Anirach@gmail.com 0817320731**
 - Send me a message!**

Code Editor (Bottom Left):

```
src > assets > Counter.jsx > Counter > decrement
1 import { useState } from 'react';
2
3 export default function Counter() {
4   const [count, setCount] = useState(0);
5   function increment() {
6     setCount(count + 1);
7   }
8   function decrement() {
9     setCount(count - 1);
10  }
11
12  return (
13    <div>
14      <h1>Count value is: {count}</h1>
15      <button onClick={increment}>increment</button>
16      <button onClick={decrement}>decrement</button>
17    </div>
18  );
19}
20
```

THINKING REACT

Start with the mockup

Imagine that you already have a JSON API and a mockup from a designer.

The JSON API returns some data that looks like this:

```
[  
  { category: "Fruits", price: "$1", stocked: true, name: "Apple" },  
  { category: "Fruits", price: "$1", stocked: true, name: "Dragonfruit" },  
  { category: "Fruits", price: "$2", stocked: false, name: "Passionfruit" },  
  { category: "Vegetables", price: "$2", stocked: true, name: "Spinach" },  
  { category: "Vegetables", price: "$4", stocked: false, name: "Pumpkin" },  
  { category: "Vegetables", price: "$1", stocked: true, name: "Peas" }  
]
```

The mockup look like this:

```
[  
  { category: "Fruits", price: "$1", stocked: true, name: "Apple" },  
  { category: "Fruits", price: "$1", stocked: true, name: "Dragonfruit" },  
  { category: "Fruits", price: "$2", stocked: false, name: "Passionfruit" },  
  { category: "Vegetables", price: "$2", stocked: true, name: "Spinach" },  
  { category: "Vegetables", price: "$4", stocked: false, name: "Pumpkin" },  
  { category: "Vegetables", price: "$1", stocked: true, name: "Peas" }  
]
```

Only show products in stock

| Name | Price |
|------|-------|
|------|-------|

Fruits

| | |
|--------------|-----|
| Apple | \$1 |
| Dragonfruit | \$1 |
| Passionfruit | \$2 |

Vegetables

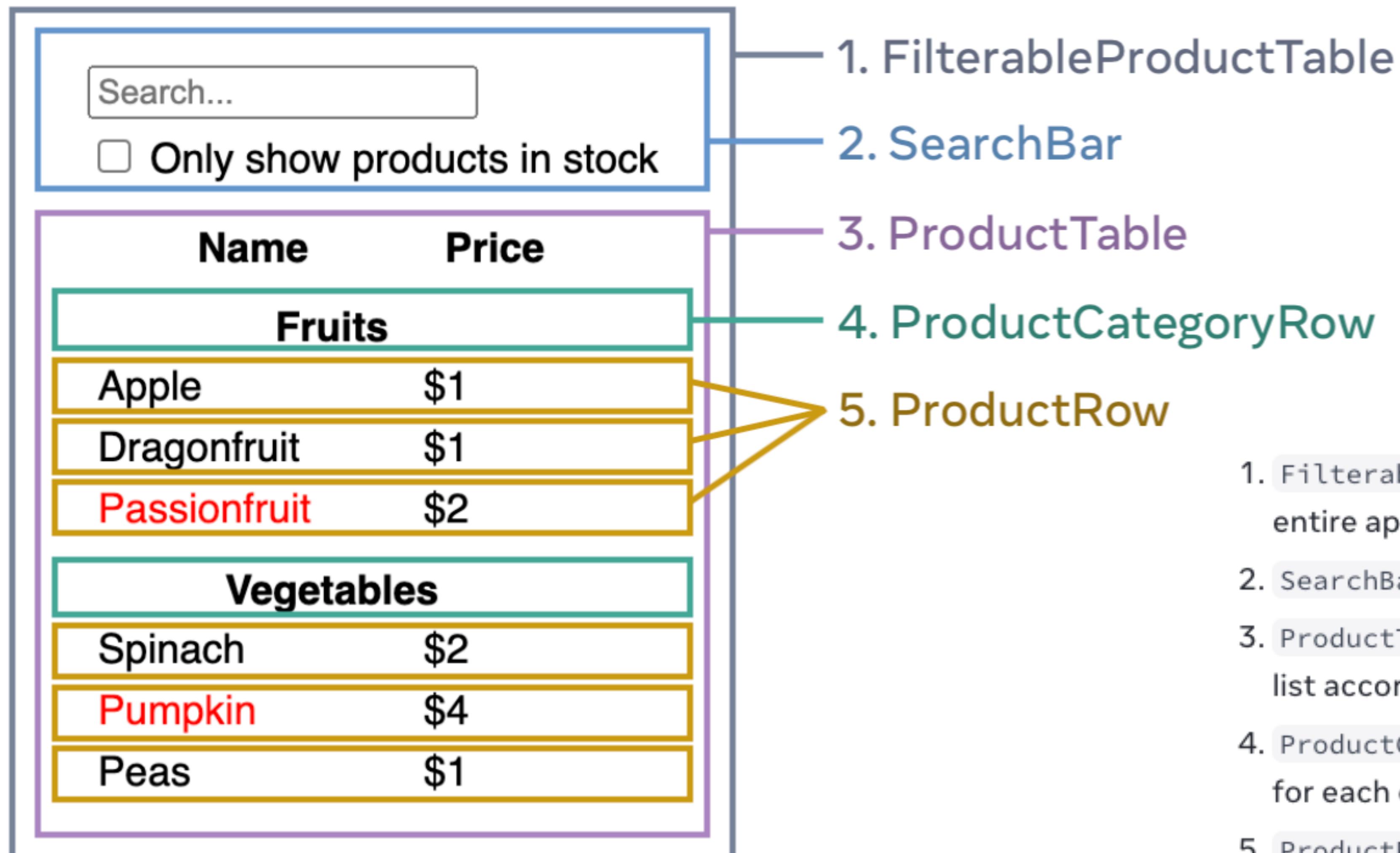
| | |
|---------|-----|
| Spinach | \$2 |
| Pumpkin | \$4 |
| Peas | \$1 |

-
- **Programming** -- use the same techniques for deciding if you should create a new function or object. One such technique is the **single responsibility principle**.
 - **CSS** -- consider what you would make class selectors for.
 - **Design** -- consider how you would organize the design's layers.
-

Step 1: Break the UI into a component hierarchy

Slide # 40

There are five components on this screen:



1. `FilterableProductTable` (grey) contains the entire app.
2. `SearchBar` (blue) receives the user input.
3. `ProductTable` (lavender) displays and filters the list according to the user input.
4. `ProductCategoryRow` (green) displays a heading for each category.
5. `ProductRow` (yellow) displays a row for each product.

Step 2: Build a static version of React

Slide # 41

App.js

Download Reset Fork

```
1 function ProductCategoryRow({ category }) {
2   return (
3     <tr>
4       <th colSpan="2">
5         {category}
6       </th>
7     </tr>
8   );
9 }
10
11 function ProductRow({ product }) {
12   const name = product.stocked ? product.name :
13     <span style={{ color: 'red' }}>
14       {product.name}
15     </span>;
16 }
```

▼ Show more

Step 2: Build a static version of React

The screenshot shows a code editor interface with a dark theme. On the left, the file tree (EXPLORER) shows a project structure with files like App.js, index.html, and styles.css. The main editor area displays the code for `App.js`, which contains a `ProductTable` component. The component iterates over products, grouping them by category and rendering `ProductCategoryRow` and `ProductRow` components. The code uses ES6 syntax with arrow functions and template literals. On the right, a preview window shows a table of products categorized into Fruits and Vegetables, with columns for Name and Price.

```
src > App.js > ProductRow
22 );
23 }
24
25 function ProductTable({ products }) {
26   const rows = [];
27   let lastCategory = null;
28
29   products.forEach((product) => {
30     if (product.category !== lastCategory) {
31       rows.push(
32         <ProductCategoryRow
33           category={product.category}
34           key={product.category} />
35       );
36     }
37     rows.push(
38       <ProductRow
39         product={product}
40         key={product.name} />
41     );
42     lastCategory = product.category;
43   });

```

| Name | Price |
|-------------------|-------|
| Fruits | |
| Apple | \$1 |
| Dragonfruit | \$1 |
| Passionfruit | \$2 |
| Vegetables | |
| Spinach | \$2 |
| Pumpkin | \$4 |
| Peas | \$1 |

Search...
 Only show products in stock

VS Code Share Sign

A

Sandbox Draft / react.dev

Preview https://kytlc.csb.app/

Ln 16, Col 1 Spaces: 2 UTF-8 LF javascriptreact ✓ Prett

THANK YOU

- React Introduction
- Fundamentals of React
- Create React Project
- React props/state
- Thinking React