<https://github.com/public-apis/public-apis>

--------------------------------------------------------

**React Overview**  
--------------------------------------------------------

React is an open-source JavaScript library focused on creating declarative user interfaces (UIs) specifically for single-page applications, using a reusable component-based concept. It’s used for handling the view layer and can be used for web and mobile apps. React’s main goal is to be extensive, fast, declarative, flexible, and simple.

**Basic Manual React Example**:  
  
1) index.html:

| <!DOCTYPE html> <**html** lang="en"> <**head**>   <**meta** charset="UTF-8">  <**script** src="https://unpkg.com/react@18/umd/react.development.js" crossorigin></**script**>  <**script** src="https://unpkg.com/react-dom@18/umd/react-dom.development.js" crossorigin></**script**>  <**title**>React Page</**title**>  </**head**> <**body**> <**div** id="root"></**div**>   <**script** src="script.js"></**script**> </**body**> </**html**> |
| --- |

2) script.js:

| const rootElement = document.getElementById('root'); const root = ReactDOM.createRoot(rootElement);  const reactElement = React.createElement('header',  {className: 'site-header'},  React.createElement('h1', {}, 'Hello from React!!!!'),  React.createElement('h2', {}, 'React Subheader) );  root.render(reactElement); |
| --- |

React uses JSX syntax - with that syntax you can write concise HTML/XML-like structures (e.g., DOM like tree structures) in the same file as you write JavaScript code, then Babel will transform these expressions into actual JavaScript code.

**JSX Alternative to the above code:**

1) npm init -y  
  
2) npm install babel-cli@6 babel-preset-react-app@3  
  
3) npx babel --watch src --out-dir . --presets react-app/prod

4) script.js:

| const rootElement = document.getElementById('root'); const root = ReactDOM.createRoot(rootElement);  const reactElement = (  <**header** className="site-header">  <**h1**>Hello from JSX!</**h1**>  <**h2**>React Subheader</**h2**>  </**header**> );  root.render(reactElement); |
| --- |

Sandbox Environment: <https://codesandbox.io/>

**The standard way of initiating a React app:**  
  
1) npx create-react-app .  
  
2) npm start

**React Component** - takes an optional input and returns a React element which is rendered on the screen.Components let you split the UI into independent, reusable pieces, and think about each piece in isolation

1) Footer.js:

| export default function Footer() {  return <**div**>All rights reserved &copy;</**div**> } |
| --- |

2) App.js:

| function App() {  return (  <**div** className="App">  <**header** className="App-header">  <**img** src={logo} className="App-logo" alt="logo" />  <**h1**>Hello From React</**h1**>  <**Footer** />  </**header**>  </**div**>  ); } |
| --- |

!!! We may pass information to a component via props (the arguments to the component function):  
  
1) App.js:  
  
 <Header text="Hello React" />

2) Header.js:

| import logo from "./logo.svg";  export const Header = (props) => {   return (  <**header** className="App-header">  <**img** src={logo} className="App-logo" alt="logo" />  <**h1**>{props.text}</**h1**>  </**header**>  ); } |
| --- |