**What is React**

A screenshot of a computer game

Description automatically generated

With react we can create components and react will take care of DOM

A blue background with white text

Description automatically generated

Consider following React Application

A screenshot of a video game

Description automatically generated

A diagram of a game

Description automatically generated with medium confidence

Essentially React Application is tree of components with the app being the root.

**Create React App**

1. Create React App (CRA)
2. Vite  
   npm create vite@latest

npm create [vite@4.1.0](mailto:vite@4.1.0)

**Project Structure**

public – public assets images,pdf files etc

src- source code of our application

App.tsx – single component

index.html – container of application -root

tsconfig.json – setting to tell typescript compilet how to compile our ts code to js

**Creating React Component**

.ts – plain typescript file

.tsx – react component

Two ways to create react component

1. JS class
2. JS function (more popular- concise, easier to write)

In React applications follow PascalCasing

JSX: JavaScript XML – Compiled by babel into JS

function Message(){

    return <h1>Hello World</h1>;

}

export default Message;

import Message from "./Message";

function App()

{

  return <div><Message/></div>

}

export default App;

{} Inside this we can write any JS expression – any piece of code that returns a value

function Message(){

    const name ="Riser";

    if(name)

      return <h1>Hello {name}</h1>;

    return <h1>Hello World</h1>;

}

export default Message;

**How React Works**

A screenshot of a computer

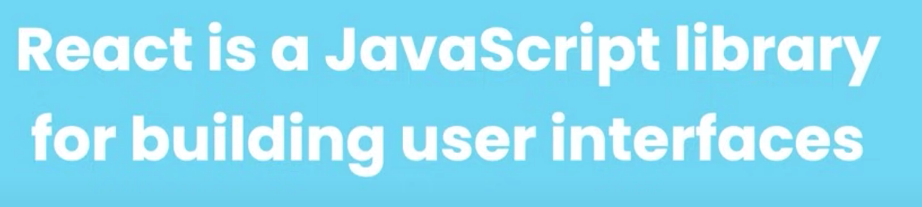
Description automatically generated

Our app tree has two component that will be converted into Virtual DOM(JS Data Structure) by react.

Vritual DOM- Lightweight in memory representation of our component tree where each node represents the component and it’s properties. It is different from actual DOM created in browser.

State or data of component changes > React updates Virtual DOM > compares previous Virtual DOM with current Virtual DOM > identify nodes that should be updated > React DOM > update the node in actual DOM.

**React Ecosystem**



A red and white logo

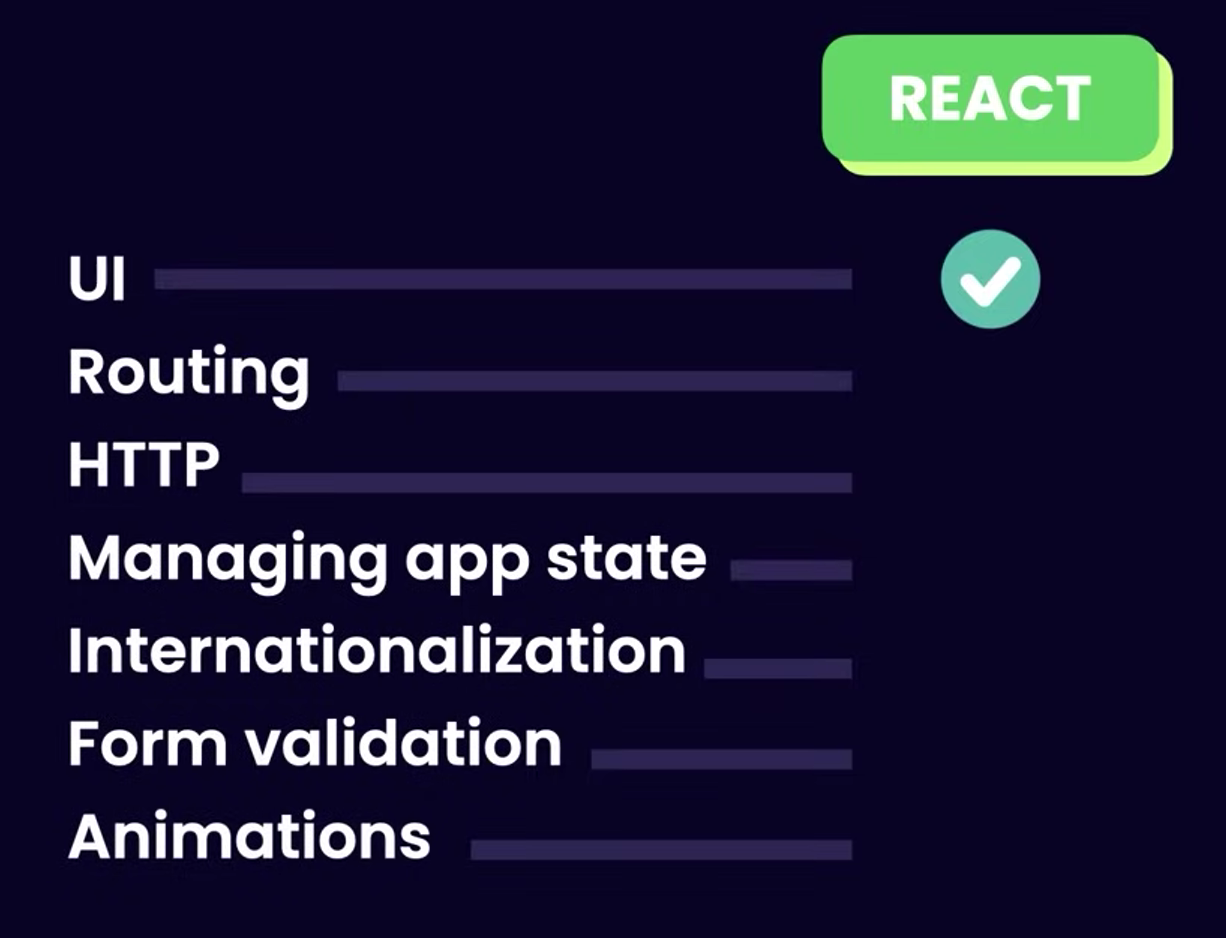
Description automatically generated

A yellow and blue background with words

Description automatically generated

Library is a tool

Framework is toolset



React is for UI only

Modern App requires other tools as well but React doesn’t bound to use specific tools with it.

**Integrate Bootstrap**

npm i [bootstrap@5.2.3](mailto:bootstrap@5.2.3)

In main.tsx

import 'bootstrap/dist/css/bootstrap.css'

**Fragments**

In React component we can return single html element only to return multiple HTML elements we have to wrap those elements as fragments <> </>.

**Rendering Lists**

 return (

    <>

      <h1>Hello</h1>

      <ul className="list-group">

        <li className="list-group-item">An item</li>

        <li className="list-group-item">A second item</li>

        <li className="list-group-item">A third item</li>

        <li className="list-group-item">A fourth item</li>

        <li className="list-group-item">And a fifth one</li>

      </ul>

    </>

  );

In React we don’t have for loops so we use map to do the task

  const items = ["a", "b", "c", "d", "e"];

  return (

    <>

      <h1>Hello</h1>

      <ul className="list-group">

        {items.map((item) => (

          <li className="list-group-item">{item}</li>

        ))}

      </ul>

    </>

  );

In React each dynamically rendered element should have unique key so React can identify those and take actions when required.

          <li key={item} className="list-group-item">

            {item}

          </li>

**Conditional Rendering**

return (

    <>

      <h1>Hello</h1>

      {items.length === 0 && <p>No Items Found</p>}

      <ul className="list-group">

        {items.map((item) => (

          <li key={item} className="list-group-item">

            {item}

          </li>

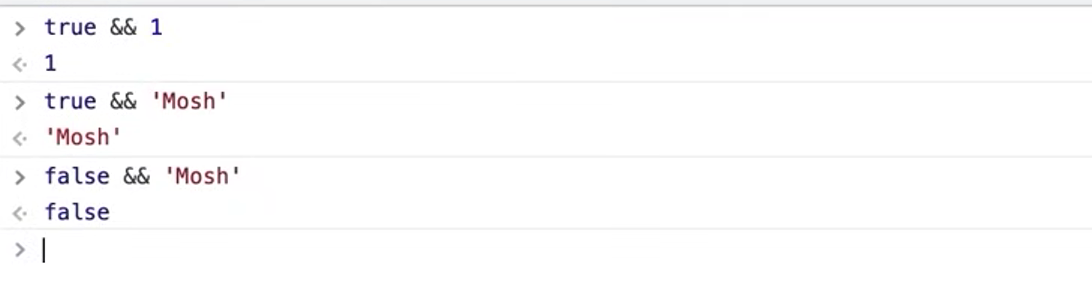
        ))}

      </ul>

    </>

  );

To understand the above code first try to understand this



So now if items is empty {} will evaluate to <p>No item found</p>

Otherwise it will evaluate to false.

**Handling Events**

  const handleClick = (event: MouseEvent, item: string, index: number) => {

    console.log(event);

    console.log(item);

    console.log(index);

  };

Here we are passing the reference of function handleClick to onClick event of li item. So that function will be called at runtime.

        {items.map((item, index) => (

          <li

            key={item}

            className="list-group-item"

            onClick={(event) => handleClick(event, item, index)}

          >

            {item}

          </li>

        ))}

**Managing State**

  const [selectedIndex, setSelectedIndex] = useState(-1);

  //arr[0] index

  //arr[1] updater function

  const handleClick = (event: MouseEvent, item: string, index: number) => {

    setSelectedIndex(index);

  };

    <li

            key={item}

            className={

              index === selectedIndex

                ? "list-group-item active"

                : "list-group-item"

            }

            onClick={(event) => handleClick(event, item, index)}

          >

**Passing Data via props**

interface Props {

  items: string[];

  heading: string;

}

function ListGroup({ items, heading }: Props)

Now items and heading can be accessed within the function

function App() {

  let items = ["a", "b", "c", "d", "e"];

  return (

    <div>

      <ListGroup items={items} heading="Alphab" />

    </div>

  );

}

**Passing Functions via Props**

function ListGroup({ items, heading, onSelectItem }: Props)

  const handleItemClick = (event: MouseEvent, item: string, index: number) => {

    handleClick(event, item, index);

    onSelectItem(item);

  };

 onClick={(event) => handleItemClick(event, item, index)}

  const onSelectItem = (item: string) => {

    console.log(item);

  };

  return (

    <div>

      <ListGroup items={items} heading="Alpha" onSelectItem={onSelectItem} />

    </div>

  );

}

**State vs Props**

A yellow and blue rectangular boxes with black text

Description automatically generated

**Passing Children**

import { ReactNode } from "react";

interface Props {

  children: ReactNode;

}

export const Alert = ({ children }: Props) => {

  return (

    <div className="alert alert-primary" role="alert">

      {children}

    </div>

  );

};

export default Alert;

      <Alert>

        <h1>Ha Mai galat</h1>

        <p>Galat meri bate</p>

      </Alert>

A white background with black text

Description automatically generated