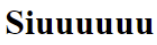
Episode - 1

▼ Creating with simple HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Namaste React</title>
</head>
<body>
    <div id="root">
        <h1>Siuuuuuu</h1>
    </div>
</body>
</html>
```



▼ Creating the same thing with JS

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Namaste React</title>
</head>
<body>
    <div id="root">
    </div>
</body>
<script>
    const root = document.getElementById("root");
    const heading1 = document.createElement("h1");
    heading1.innerHTML = "Siuuuuu";
    root.appendChild(heading1);
</script>
</html>
```

Siuuuuu

▼ Creating the same with basic React

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Namaste React</title>
</head>
<body>
    <div id="root">
    </div>
</body>
<script crossorigin src="https://unpkg.com/react@18/umd/react.development.js"></script>
<script crossorigin src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>

<script>
    const root = ReactDOM.createRoot(document.getElementById("root"));
    const heading = React.createElement("h1",{},"Muchas Gracias Eificion Esta Para Vosotros Siuuuuuuu");
    root.render(heading);
</script>
</html>
```

Muchas Gracias Eificion Esta Para Vosotros Siuuuuuuu

- root and heading constants are objects.

- When you do React.createElement(”h1”,{},””);
    -

```
1st parameter -> "h1"
This that this is an h1 HTML tag
2nd parameter-> here it is empty but it will have properties of the element such
as id,className,style it can also have event handlers such as onClick
React.createElement("h1", { style: { color: "red", fontSize: "20px" } }, "Hello")
React.createElement("button", { onClick: handleClick }, "Click me")
React.createElement("div", { className: "container" }, "Content")

In third parameter we pass child of the element
```

When you do root.render(); it converts the element passed, into HTML.

In HTML if you want to include JS code then you can either write JS code in the same file or you can write JS code in an external file and import it into the HTML file.

```
<script crossorigin src="https://unpkg.com/react@18/umd/react.development.js"></script>
```

▼ Cross-Origin

When a script file is fetched from a different domain (cross-origin request), the browser checks the CORS headers returned by the server to determine whether it is allowed to access the script file. The `crossorigin` attribute helps in specifying the behavior of the browser in case of a cross-origin request.

`https://unpkg.com/react@18/umd/react.development.js`

Lets break down the above URL.

- unpkg is a popular CDN for JS [packages and libraries.
- react@18 specifies the version of react that is being imported.
- This link is to import react itself.

React overrides the element that are present in in-line HTML or that are in scripts

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Namaste React</title>
</head>
<body>
    <div id="root">
          <h1> Siuuuu</h1>
    </div>
</body>
<script crossorigin src="https://unpkg.com/react@18/umd/react.development.js"></script>
<script crossorigin src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>

<script>
    const root = ReactDOM.createRoot(document.getElementById("root"));
    const heading = React.createElement("h1",{},"Muchas Gracias Eificion Esta Para Vosotros Siuuuuuuu");
    root.render(heading);
```

```
</script>
</html>
```

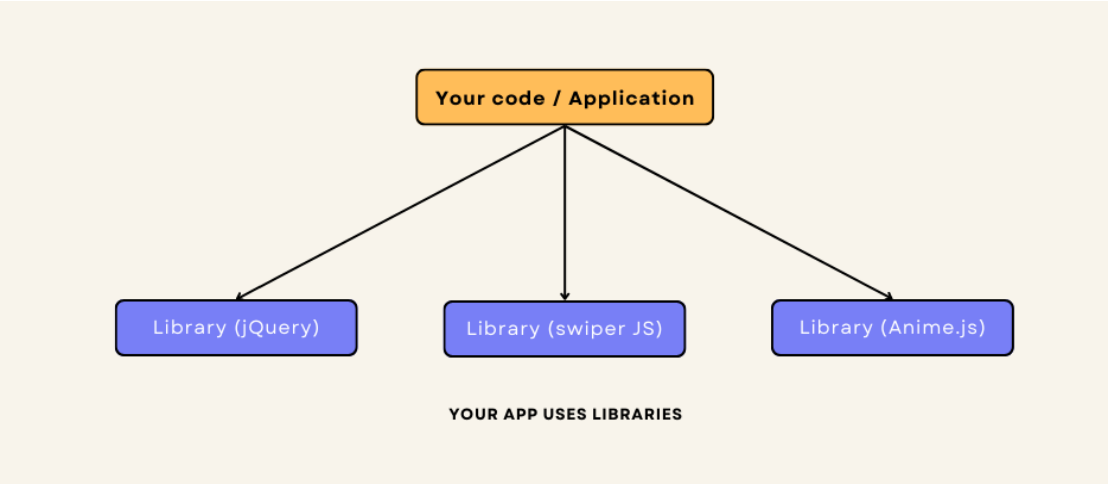
h1 tag in HTML will get override by react element when it gets rendered.

It is visible when you reload the page for quite few seconds the page first loads the value in h1 tag from html and then renders the react elements.

▼ What is React? Library or a Framework

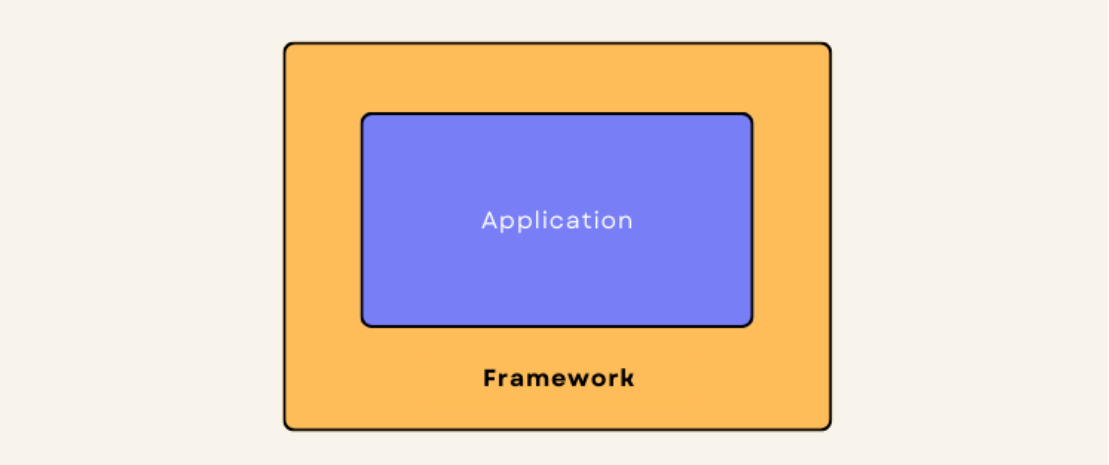
Library

- A library is a collection of pre-written codes that can be used to simplify tasks.
- Technically, it is a **set of pre-defined functions and classes** that developers can use to simplify their work and speed up the development process.
- You can integrate libraries into existing projects to add additional functionalities. It means **your code or application uses the library**.



Framework

- A framework is like a **“foundation”** on which developers build applications.
- It means a framework provides us an eco-system or structure along with reusable pieces of code to deal with complex development challenges easily.
- If we are creating an application using next.js or Laravel, then we work within their infrastructure i.e. we cannot put a file anywhere as we want or change anything as we wish. They provide us ways to manage routing, do authentication, manipulate databases, provide folders to put our public files (HTML, CSS, js, static assets like images), etc.



Key difference

- The main difference between a framework and a library lies in a term called **”inversion of control”**.
- When you use a library, you are in charge of the flow of the application. You are choosing when and where to call the library.
- But when you use a framework, the framework is in charge of the flow. It provides some places for you to put your code, and it will call your code when required.
- It does not provide the complete **“eco-system”** to develop an application not even provide a design system.
- It **lacks a lot of built-in features** and **heavily depends on third-party packages**, like for routing you need to use react-router, to deal with forms you need to select one package among Redux-Forms, Formiq, or Final-Form, etc., for making API calls you need to decide whether to use AXIOS or fetch API and thus, it makes React confusing and time-consuming.
- A framework provides you a proper **eco-system** to do something which means you cannot write or place anything anywhere, everything is built according to a design pattern.
- **For Example,** Next.js is a framework of React JS which provides a lot of built-in features like routing, server-side rendering, image optimization, and much more which you can do in React JS too, but you need to write a lot of extra codes.
- A **framework specifies a pattern** like where you will put routing files, component files, pages files, etc. Next.js provides a proper place to put your files accordingly and has a design pattern **that makes it a framework**.