# Chapter1 – Inception

What is React? Why is `React` known as `React`?

React is a JavaScript library that is called React because it quickly responds to changes without reloading the whole page.

What is a Library?

A library is a collection of prewritten code snippets that can be used and reused to perform specific tasks. You can integrate a JavaScript library into your application code, which helps speed up development and reduces the chances of errors. Examples include React, jQuery, and Underscore.

What is a Framework?

A framework provides a foundational structure for building a website or application. An example of this is Angular.

Similarities between Library and Framework?

Both are created to speed up development by providing pre-written, reusable code.

Both help in solving regular or common problems that developers face during application development.

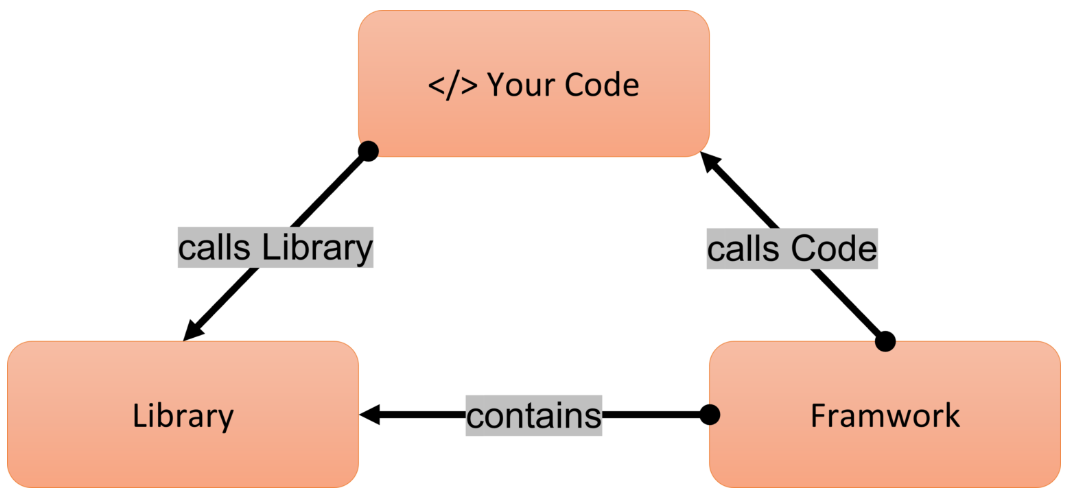
Both can be used to optimize performance by offering tested and efficient solutions.

Difference between Library and Framework?

**Control:** In the case of a library, the developer controls when to call the code. However, with a framework, the flow is managed by the framework itself, which calls the developer's code as needed.

**Flexibility:** A library offers more flexibility, allowing you to use it wherever you need. In contrast, a framework follows a structured approach with predefined rules.

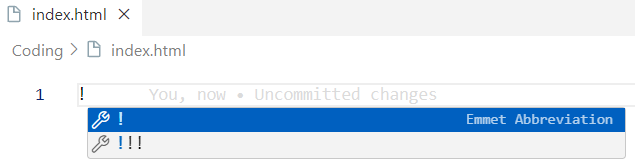
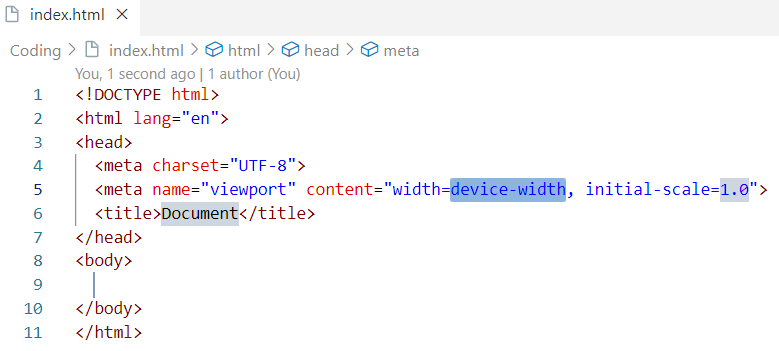
**Usage:** A library consists of a collection of specific functions designed to assist with tasks, whereas a framework offers a comprehensive solution for building applications.



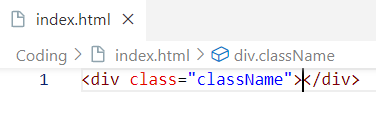
What is `Emmet`?

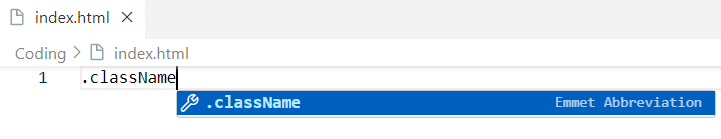
It's a tool for IDEs and text editors that helps developers generate complete boilerplate code snippets using shortcuts.

For example, when we type ! in an html file and press enter it generates foundation structure of an html document.

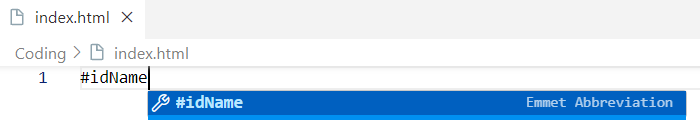
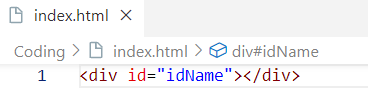


When you type .<className> in an HTML file and press Enter, it quickly creates a <div> element with the specified class name.



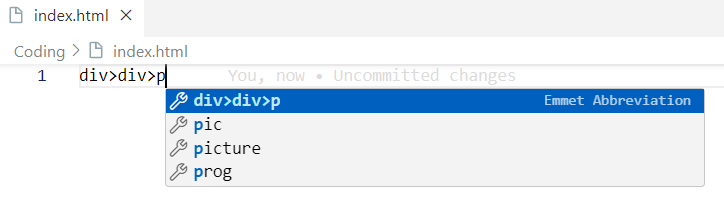
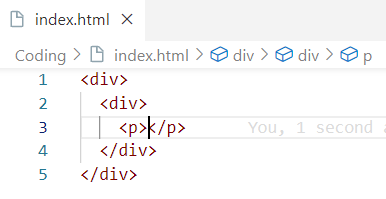


when we type #<idName> in an html file and press enter, it quickly creates a <div> element with the specified id name.



when we want to generate a nested html elements structure we need below syntax.

Parent > child 1 > child’s child > …



Before learning react, let’s create a basic hello world program using html.





Now let’s create a basic hello world program using JavaScript.



**>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>**

Now let’s create a basic hello world program using react.

Our browser doesn't know what React is, so we need to add it to our project to use its features. There are two ways to add React:

Way 1: Use a CDN (Content Delivery Network).  
Way 2: Install the React library with NPM (Node Package Manager) and then import it into your project.

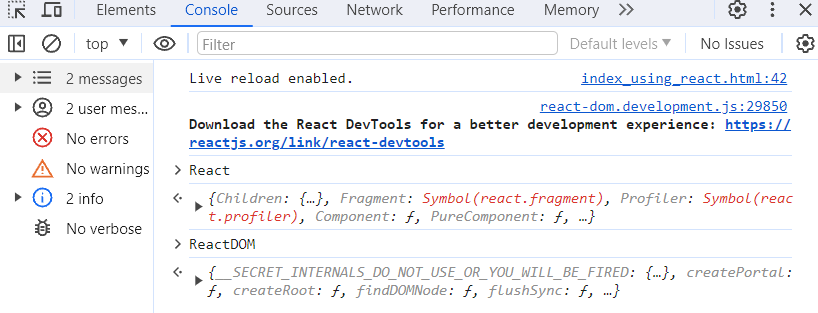
In this chapter we are going discuss about adding react into our project using CDN.

What is `CDN`? Why do we use it?

CDN stands for Content Delivery Network. It is a group of servers located in different places that work together to quickly deliver Internet content. The main purpose of a CDN is to provide content securely and efficiently through this network of servers. A CDN also aims for 100% uptime, meaning it is always available.



By using these two CDN links, we made React available to the browser. This is the shortest React program. Why? Because if we check in the browser, we can see that React and ReactDOM are now available as objects.



These objects come from the CDNs we just imported.

The first CDN link <https://unpkg.com/react@18/umd/react.development.js> is the main file for React.

The second CDN link <https://unpkg.com/react-dom@18/umd/react-dom.development.js> is needed for working with the DOM.

What is difference between react and react-dom/client packages?

The React package includes functions like React.createElement(), React.Component, React.Children, and other tools to help with creating elements and component classes. We can use these tools to build React components or elements.

The react-dom/client package contains ReactDOM.render(), which takes a React element as an argument. This function tells the browser where to display the React element in the DOM.

What is CORS?

Understanding CORS (Cross Origin RequestS) with a Simple Example -

**Imagine a Playground. Your Playground (Your Web Page):** This is where you play, like your own backyard. **Neighbour’s Playground (External Server):** This is your neighbour’s playground with cool swings.

**Step 1: Asking to Play.** You ask your neighbour if you can use their swings. This is like a **cross-origin request** because you want something from a different place.

**Step 2: Neighbour’s Rules (CORS).** Your neighbour has rules about who can use the swings. This is like **CORS,** which decides who can access resources from another domain.

**Step 3: Getting Permission-**If your neighbour says yes, you can play on the swings. If they don’t know you or you don’t follow their rules, you can’t use them.

Key Points -

**Cross-Origin Request:** Asking to use something from another place.

**CORS:** The rules your neighbour has about who can play.

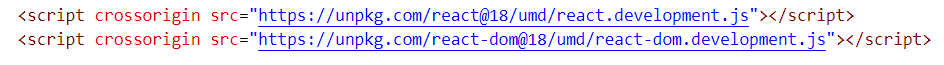
CORS Explained with the Playground Analogy

* **Cross-Origin Requests:** Wanting to use your neighbour’s swings is a cross-origin request.
* **CORS (Neighbour’s Rules):** Your neighbour checks who can use the swings. CORS works like that.
* **Permission Granted:** If your neighbour says it’s okay, you can play (the server allows the request). If not, you can’t play (the request is blocked).

Definitions

* **Cross-Origin Request:** A request made by a web page to access something from a different domain, like asking your neighbour to use their swings.
* **CORS (Cross-Origin Resource Sharing):** A security rule that allows or blocks cross-origin requests, just like your neighbours’ rules about who can play on their swings.

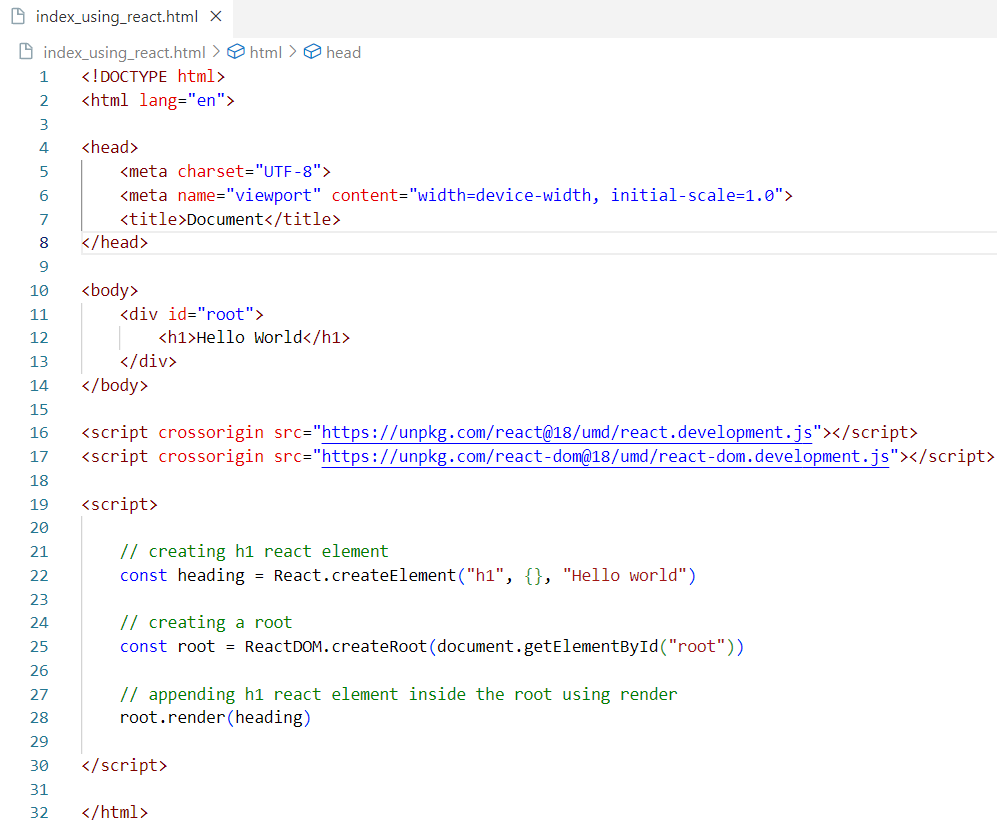
What is crossorigin in script tag?



In these scripts tag the crossorigin attribute indicates that you are loading JavaScript files React and ReactDOM from an external CDN unpkg.com, which is a different domain than your own website. This is a **cross-origin request**. The crossorigin attribute helps the browser manage this request securely, allowing it to check if the CDN permits such access according to CORS rules.

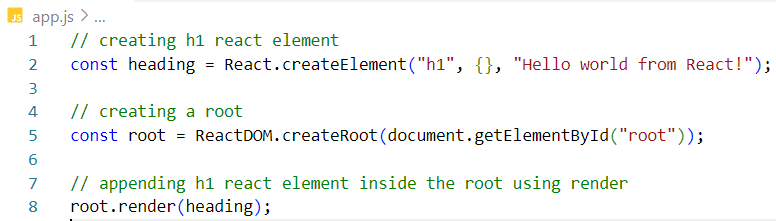
If we navigate to these CDN links, we get source code of React which is written using JavaScript by Facebook developers. So, react in the end of the day is JavaScript.

Now let’s continue writing hello world program using react.



Now let’s keep this script code in an external JavaScript file and put a reference to this file in html code.





The application behaves the same way it was behaving before.

**>>>>>>>>>>>>>>>>>>>>>**

React.createElement

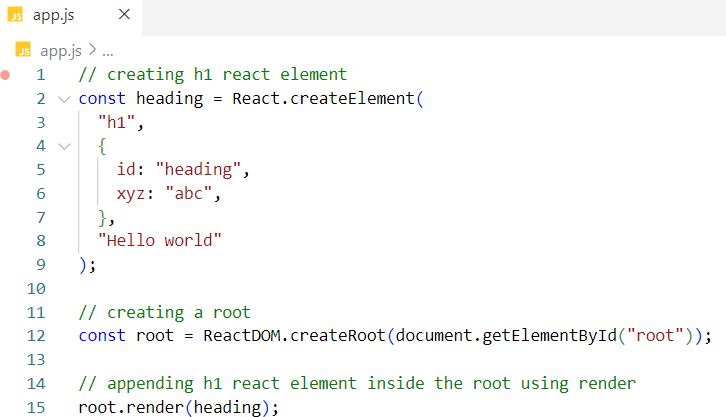
React.createElement is a key function in React that helps you create a React element. It takes three arguments:

1. tagName: This is the name of the HTML tag you want to create, like h1, div, etc.
2. props: These are the attributes you want to add to the HTML element. This can also be set to null if you don’t have any attributes.
3. children: This represents the content inside the HTML element. It can be a string or an array of other React elements.

Syntax - React.createElement(tagName, {props}, [children])

When you use this function, it returns a JavaScript object, which is called a React element.

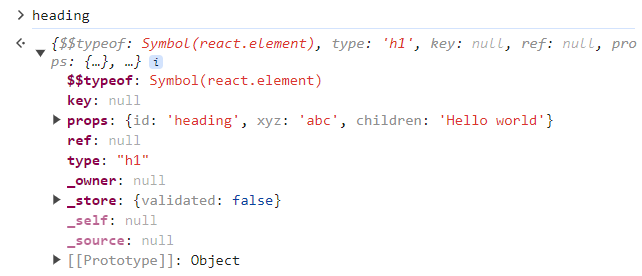
In the example above, let's add some attributes to the <h1> React element. We can specify these attributes within the props object.



We can see that these attributes have been added to the <h1> element in the DOM.



In the JavaScript code snippet above, heading is a plain JavaScript object that represents a React element



# >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

How to create nested elements using react?

How to represent two levels of nested elements using HTML?

<div id="parent">

<div id="child">

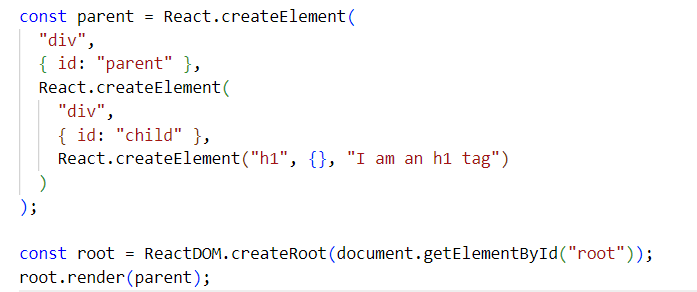
<h1>I am an h1 tag</h1>

</div>

</div>

In this example, we are creating a parent <div> element that contains a child <div>, which in turn contains an <h1> tag.

How to represent two levels of nested elements using REACT?





How to represent sibling elements using HTML?

<div id="parent">

<div id="child">

<h1>I am an h1 tag</h1>

<h2>I am an h2 tag</h2>

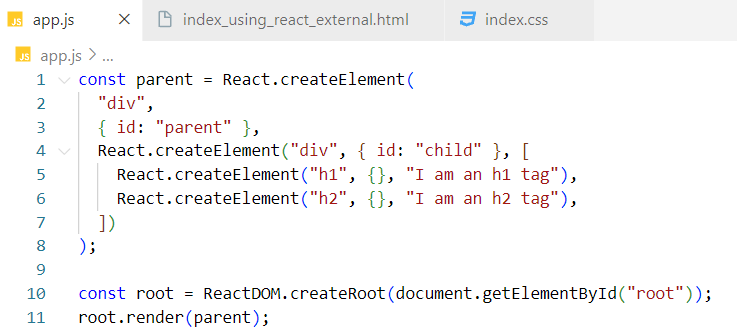
</div>

</div>

In this example, the <h1> and <h2> tags are siblings because they are both inside the same parent <div> element.

How to represent sibling elements using REACT?

In React, when there are two sibling elements, they are wrapped inside an array. However, if there is only one child, react does not wrap it in an array, as shown above.



How to represent nested sibling elements using HTML?

<div id="parent">

<div id="child1">

<h1>I am an h1 tag</h1>

<h2>I am an h2 tag</h2>

</div>

<div id="child2">

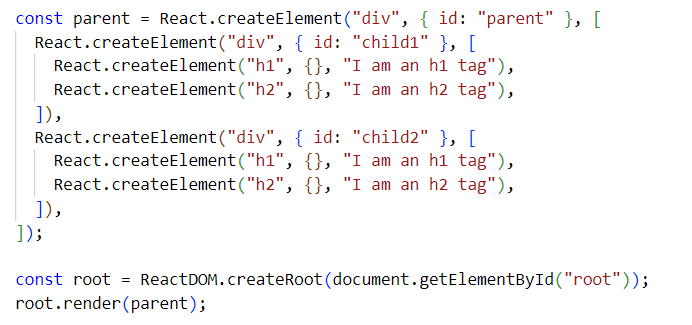
<h1>I am an h1 tag</h1>

<h2>I am an h2 tag</h2>

</div>

</div>

How to represent nested sibling elements using React?



**# >>>>>>>>>>>**

**# >>>>>>>>>>>**

**# >>>>>>>>>>>**

**# >>>>>>>>>>>**

**# >>>>>>>>>>>**

**# >>>>>>>>>>>**

**# >>>>>>>>>>>**

**# >>>>>>>>>>>**

**# >>>>>>>>>>>**

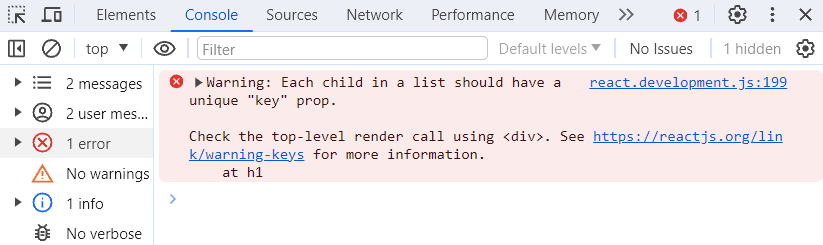
**# >>>>>>>>>>>**

**# >>>>>>>>>>>**

**# >>>>>>>>>>>**

The code works fine, but we get a warning in the console when multiple children are introduced.

This warning is valid because we have not provided a key for each of the children. React tracks these elements using their keys. We will discuss keys in more detail in later chapters



When dealing with deeply nested elements, writing code in this manner can become difficult to read, maintain, and test. To address this issue, JSX was introduced. We will cover JSX in more detail in later chapters.

**# >>>>>>>>>>>>>>>>>>>>>>>>>>>**

**# >>>>>>>>>>>>>>>>>>>>>>>>>>>**

**# >>>>>>>>>>>>>>>>>>>>>>>>>>>**

**# >>>>>>>>>>>>>>>>>>>>>>>>>>>**

**# >>>>>>>>>>>>>>>>>>>>>>>>>>>**

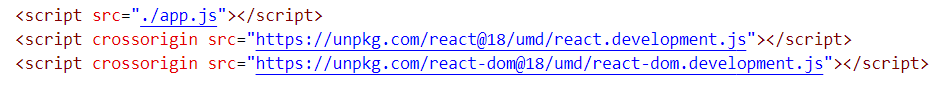
**# >>>>>>>>>>>>>>>>>>>>>>>>>>>**

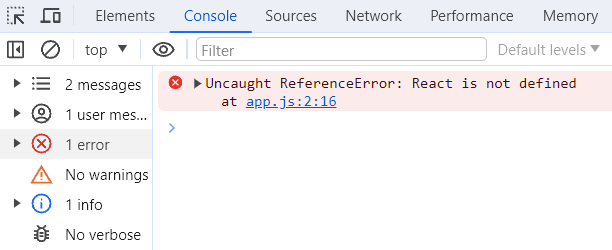
**# >>>>>>>>>>>>>>>>>>>>>>>>>>>**

Important Observations

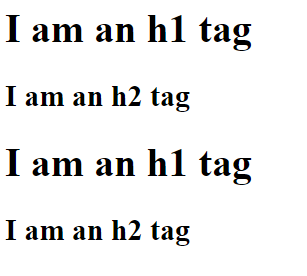
**1. The order of the script tag matters.**

**We moved our script tag two levels up, placing it before the CDN fetches the React code into our browser. As a result, we encounter an error stating that React is not defined. This occurs because we are trying to access React before its source code has been loaded into the browser.**





2. When rendering a React element into the DOM, react overrides the child elements that are present inside the root element



The text 'Not rendered' will be replaced by the React element that we are pushing into the root. Therefore, when we load the page, 'Not rendered' will be displayed for a brief moment until the React source code is fetched from the CDNs and made available to the browser. Once the React code is available, it will be replaced by the React element, which consists of two child div elements under one parent div.

**3. We can use React in specific parts of our application.**

**In the next example, we are using React for the root component, but we are not using it for the header and footer components. This is why we refer to React as a library. it allows us to integrate React into just a portion of our app, giving developers control over the overall application structure.**



ReactDOM.createRoot

**Definition:** createRoot is a ReactDOM API that creates a root in the DOM where the JavaScript engine runs the React application.

**Returns:** It returns an object containing two methods render and unmount

**Syntax:** ReactDOM.createRoot(document.getElementById(<rootID>))

render

**Definition:** The render method takes a React element or JSX as an argument and renders it inside the specified root element in the DOM.

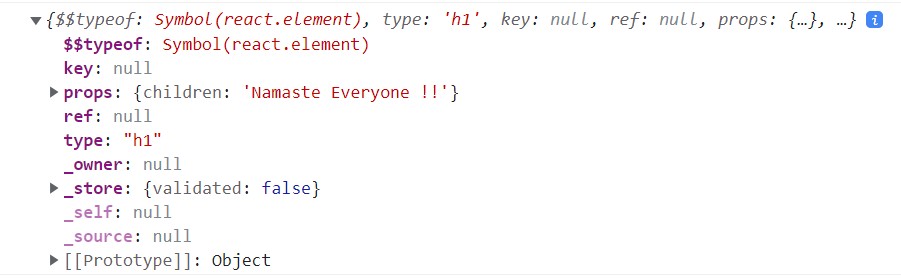
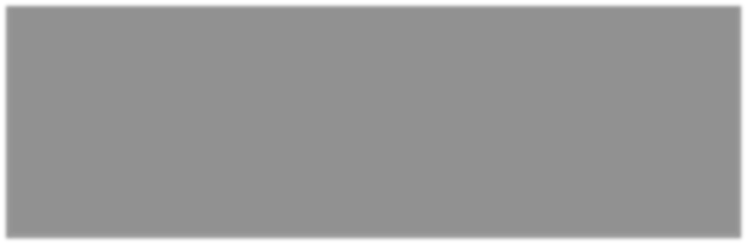
**Returns:** This method returns undefined.

**Syntax:**<root>.render(<JSX/React element>)

Logging React Elements

When logging a React element to the console, you will see a plain JavaScript object representing the element. The console output may vary depending on the component structure and how the console displays objects.

-



What is difference between `Development` and `production` stage of an application?

**Development Stage**

The application is not public and is primarily for internal use and testing.

Code changes are frequent, allowing for quick experimentation and iteration.

The development build includes debugging and error reporting features, making it slower compared to production builds.

**Production Stage**

Refers to the version of the application that is made public and available to end users.

The code is optimized for performance and security.

The production build is faster and more efficient as it omits debugging tools and non-essential features.

It is typically minified and bundled to reduce load times and improve user experience.

The **development build** is slower due to debugging features, while the **production build** is optimized for speed and efficiency, suitable for public deployment.