Namaste React Notes

Lecture 1- Inception

Hello World Program by using HTML

```html
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Namaste React</title>
</head>

<body>
  <div id="root">
    <h1>Hello World using HTML</h1>
  </div>
</body>

</html>
```

Hello World Program by using Javascript

```html
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Namaste React</title>
</head>

<body>
  <div id="root">
  </div>
  <script>
    const heading = document.createElement("h1");
    heading.innerHTML = "Hello World from JavaScript"
    const root = document.getElementById("root")
    root.appendChild(heading)
  </script>
</body>

</html>
```

Injecting React into Html file using CDN(Content Delivery Network)

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Namaste React</title>
</head>

<body>
  <div id="root">
  </div>
  <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>
</body>

</html>
```

Hello World Program using React

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Namaste React</title>
</head>

<body>
  <div id="root">
  </div>
  <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 heading = React.createElement("h1", {}, "Hello World from React")
    const root = ReactDOM.createRoot(document.getElementById("root"))
    root.render(heading)
  </script>
</body>

</html>
```

Separating the JavaScript Code, CSS and HTML into separate files

index.html

```html
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="index.css">
  <title>Namaste React</title>
</head>

<body>
  <div id="root">
  </div>
  <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 src="App.js"></script>
</body>

</html>
```

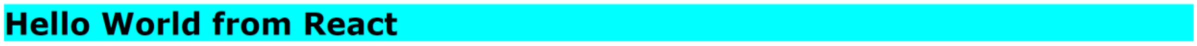
App.js

```javascript
const heading = React.createElement(
  "h1",
  { id: "heading", "data-testid": "heading", testid: "heading" },
  "Hello World from React"
);
const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(heading);
```

index.css

```css
#root {
  background-color: aqua;
}
```

Output:

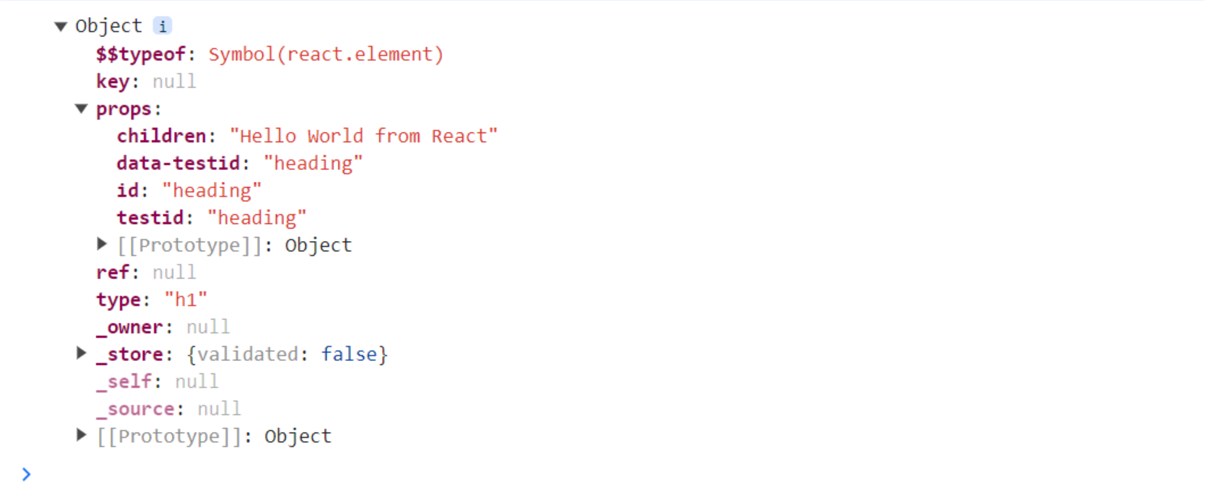


Hello World from React

If we console.log(heading) in App.js

```
const heading = React.createElement(
  "h1",
  { id: "heading", "data-testid": "heading", testid: "heading" },
  "Hello World from React"
);
console.log(heading)
const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(heading);
```

Output:



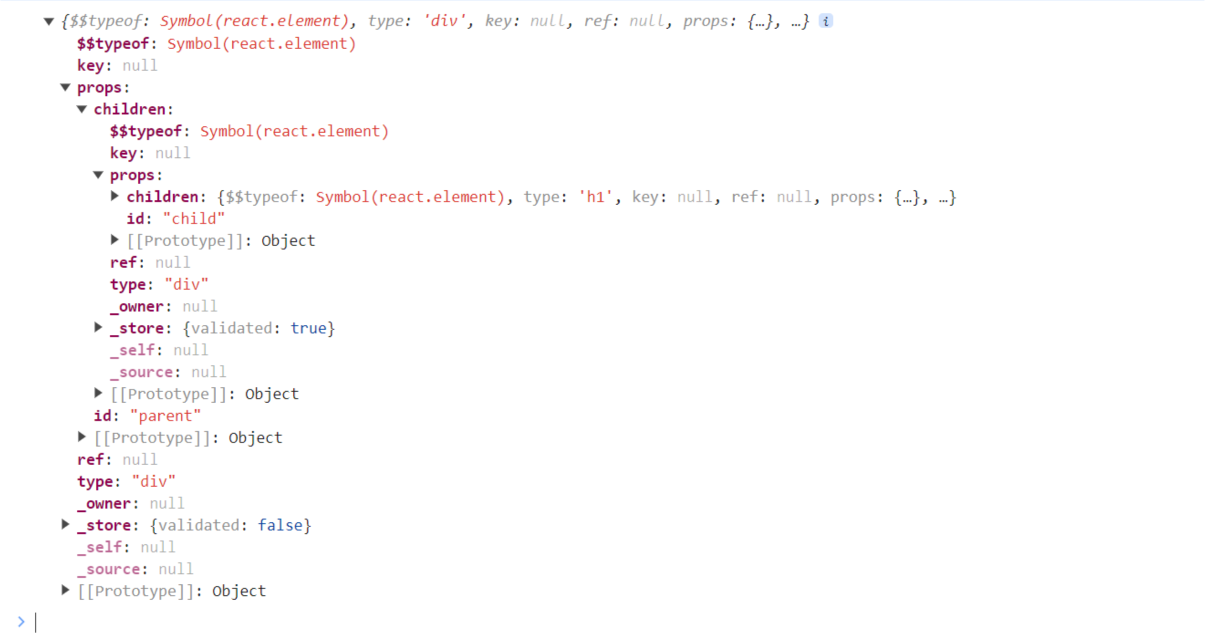
```
▼ Object ℹ
    $$typeof: Symbol(react.element)
    key: null
  ▼ props:
      children: "Hello World from React"
      data-testid: "heading"
      id: "heading"
      testid: "heading"
    ▶ [[Prototype]]: Object
    ref: null
    type: "h1"
    _owner: null
  ▶ _store: {validated: false}
    _self: null
    _source: null
  ▶ [[Prototype]]: Object
›
```

Creating Nested Elements in React

Trying to create

```
  <div id="parent">
    <div id="child">
      <h1 id="inner-child">Hello World!</h1>
    </div>
  </div>
```

```
const parent = React.createElement(
  "div",
  { id: "parent" },
  React.createElement(
    "div",
    { id: "child" },
    React.createElement("h1", { id: "inner-child" }, "Hello World!")
  )
);
const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(parent);
```



```
▼ {$$typeof: Symbol(react.element), type: 'div', key: null, ref: null, props: {…}, …} ℹ
    $$typeof: Symbol(react.element)
    key: null
  ▼ props:
    ▼ children:
        $$typeof: Symbol(react.element)
        key: null
      ▼ props:
        ▶ children: {$$typeof: Symbol(react.element), type: 'h1', key: null, ref: null, props: {…}, …}
          id: "child"
        ▶ [[Prototype]]: Object
        ref: null
        type: "div"
        _owner: null
      ▶ _store: {validated: true}
        _self: null
        _source: null
      ▶ [[Prototype]]: Object
      id: "parent"
    ▶ [[Prototype]]: Object
    ref: null
    type: "div"
    _owner: null
  ▶ _store: {validated: false}
    _self: null
    _source: null
  ▶ [[Prototype]]: Object
> |
```

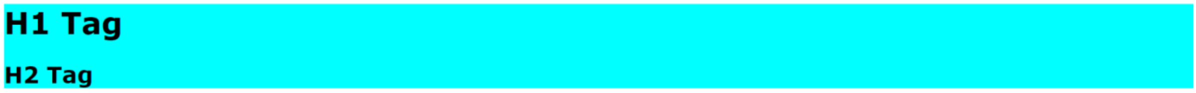
Notice the children in the above example

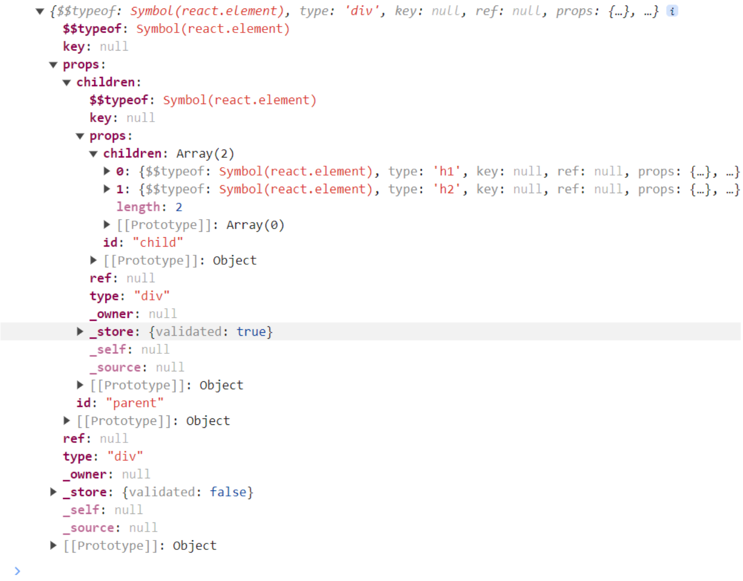
Creating Siblings in React

```
  <div id="parent">
    <div id="child">
      <h1 id="inner-child1">H1 Tag</h1>
      <h2 id="inner-child2">H2 Tag</h2>
    </div>
  </div>
```

```
const parent = React.createElement(
  "div",
  { id: "parent" },
  React.createElement(
    "div",
    { id: "child" },
    [React.createElement("h1", { id: "inner-child1", key:"1" }, "H1 Tag"),
    React.createElement("h2", { id: "inner-child2", key:"2" }, "H2 Tag")]
  )
);
console.log(parent);
const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(parent);
```

Siblings are passed inside an Array





```
▼ {$$typeof: Symbol(react.element), type: 'div', key: null, ref: null, props: {…}, …} ℹ
    $$typeof: Symbol(react.element)
    key: null
  ▼ props:
    ▼ children:
        $$typeof: Symbol(react.element)
        key: null
      ▼ props:
        ▼ children: Array(2)
          ▶ 0: {$$typeof: Symbol(react.element), type: 'h1', key: null, ref: null, props: {…}, …}
          ▶ 1: {$$typeof: Symbol(react.element), type: 'h2', key: null, ref: null, props: {…}, …}
            length: 2
          ▶ [[Prototype]]: Array(0)
          id: "child"
        ▶ [[Prototype]]: Object
        ref: null
        type: "div"
        _owner: null
      ▶ _store: {validated: true}
        _self: null
        _source: null
      ▶ [[Prototype]]: Object
      id: "parent"
    ▶ [[Prototype]]: Object
    ref: null
    type: "div"
    _owner: null
  ▶ _store: {validated: false}
    _self: null
    _source: null
  ▶ [[Prototype]]: Object
›
```

It becomes extremely complex to write React Code like this. So, there came the need for JSX (HTML Like syntax inside Javascript)