1 - Inception

1) Emmet: It's a essential toolkit for web-developers. It allow our text-editor to type shortcuts that are then expanded into full pieces of code.

2) Libary and framework

	Libary	Framework
Control Flow	It's essential a collection of f(). that can be called by program to perform a specific tasks	It provide a structure for how an application is built and how it's component interacts.
Invesion of Control	The programmer remains in control of the flow of the program and decides when to call the libary f().	The framework itself is in control and the programmer

HTML

```
<body>
    <div id="root">
        <h1>Hello World!</h1>
    </div>
</body>
```

JavaScript

- Using <script> tag.

```
<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>
```

Hello World from JavaScript

this are the super power that browser already have in it. It has JS Engine, it execute in it.

React:

CDN: Content Delivery Network, is to deliver content through a network of server in a secure and efficent way.

```
    <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>
```

→ it's plain JS code.

Cross origin: To share the resource from one domain to another domain.

Why there is two link?

React → It's a core of React for building UI.

React.DOM → It's react libary that allow React to interact with the DOM.

E.g. React.createElement(), React.Children

```
    <!-- Creating h1 tag using React -->
    <script>
      // createElement takes three parameters: 1st is the tag name, 2nd is the obj, 3rd is the content
      const heading = React.createElement("h1", {}, "Hello world from React");
      //   Now you need to put this into the root #10
      //   In react you need to tell this is the root element
      const root = ReactDOM.createRoot(document.getElementById("root"));

      root.render(heading);
    </script>
```

↳ this is

→ created a h1 tag

→ created a root inside the header, it's a place where all the

react code will run.

- Everything to render, will render inside that.

Hello world from React

```
▼<div id="root"> == $0
  <h1>Hello world from React</h1>
</div>
```

- Create a App.JS and keep all the JS code there.

React.createElement("h1", {}, "Hello World")

↳ this object is the place where we give attributes to the tag.

like if h1 need a id.

```
1  const heading = React.createElement(
2    "h1",
3    { id: "heading" },
4    "Hello world from React"
5  );
```

```
1  const heading = React.createElement(
2    "h1",
3    { id: "heading", xyz: "abc" },
4    "Hello world from React"
5  );
```

```
▼<div id="root">
  <h1 id="heading" xyz="abc">Hello world from React</h1> == $0
 </div>
```

index.html U | style.css U | App.js U

style.css > #heading

```
1  #heading{
2      color: red;
3  }
```

Hello world from React

React.createElement() → basically known as JS Object.

To create nested element:

```
1  /**
2   * <div id="parent">
3   *   <div id="chid">
4   *    <h1></h1>
5   *   </div>
6   * </div>
7   */
```

```
const parent = React.createElement(
  "div",
  { id: "parent" },
  React.createElement(
    "div",
    { id: "child" },
    React.createElement("h1", {}, "I am H1 tag")
  )
);

console.log(parent);

const root = ReactDOM.createRoot(document.getElementById("root"));

root.render(parent);
```

I am H1 tag

This is how React work BTS

React Element (Object) => HTML (Brower Understand)

Now suppose to create siblings:

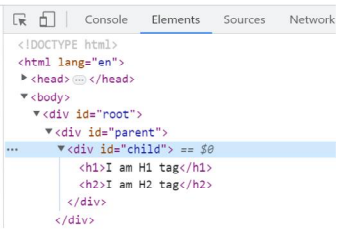
```
/**
 * <div id="parent">
 *   <div id="chid">
 *     <h1>I am H1 tag</h1>
 *     <h2>I am H1 tag</h2>
 *   </div>
 * </div>
 */
```

the 3rd argument is children. You can give it as an array of element

```
const parent = React.createElement(
  "div",
  { id: "parent" },
  React.createElement("div", { id: "child" }, [
    React.createElement("h1", {}, "I am H1 tag"),
    React.createElement("h2", {}, "I am H2 tag"),
  ])
);
```

I am H1 tag

I am H2 tag



Now it's bit hard to read the code.

```
 * <div id="parent">
 *   <div id="child">
 *     <h1>I am H1 tag</h1>
 *     <h2>I am H1 tag</h2>
 *   </div>
 *   <div id="child2">
 *     <h1>I am H1 tag</h1>
 *     <h2>I am H1 tag</h2>
 *   </div>
 * </div>
const parent = React.createElement("div", { id: "parent" }, [
  React.createElement("div", { id: "child" }, [
    React.createElement("h1", {}, "I am H1 tag"),
    React.createElement("h2", {}, "I am H2 tag"),
  ]),
  React.createElement("div", { id: "child2" }, [
```

I am H1 tag

I am H2 tag

I am H1 tag

I am H2 tag

```
    React.createElement("h1", {}, "I am H1 tag"),
    React.createElement("h2", {}, "I am H2 tag"),
  ]),
  React.createElement("div", { id: "child2" }, [
    React.createElement("h1", {}, "I am H1 tag"),
    React.createElement("h2", {}, "I am H2 tag"),
  ]),
]);
```

I am H2 tag

- Now there is something known as JSX.

What is root.render() doing?

→ root.render(parent)

↳ This will be put in the

```
<div id = "root">
  ....
</div>
```

if suppose already something there in it

```
<div id = "root">
  <h1> Something </h1>
</div>
```

↳ This will be replaced by the parent.

```
<h1>Hello top of the Root</h1>
<div id="root"></div>
<h1>Hello below of the Root</h1>
```

→ React is working only here.

Hello top of the Root

I am H1 tag

I am H2 tag

I am H1 tag

I am H2 tag

Hello below of the Root

React is a libary, it can be applied in small portion of page (like header, footer)

```
11      <div id="header">
12        <h1>Hello this is header</h1>
13      </div>
27  const root = ReactDOM.createRoot(document.getElementById("header"));
28
```

What is difference b/w React. development and React. production?

— Development is the stage of an application before it's made public

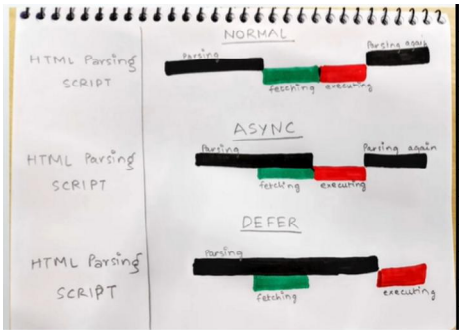
— Production when the application is made public.

Development build is several times (3-5x) slower than production build.

What is async and defer?

Async — Boolean attributes, the script is downloaded in parallel to parsing the page, and executed as soon as it's available and don't wait for anything.

Defer - Boolean attributes, the script is downloaded in parallel to parsing the page, and executed after the page has finished parsing. The defer tell the browser not to wait for the script. Instead the browser will continue to process the HTML, build DOM. Defer is preferred as order of execution matters.


NORMAL
HTML Parsing
SCRIPT
Parsing
Parsing again
fetching
executing
ASYNC
HTML Parsing
SCRIPT
Parsing
Parsing again
fetching
executing
DEFER
HTML Parsing
SCRIPT
Parsing
fetching
executing