1_2_introduction

11 September 2023 08:14 PM

What is Library and framework?
React is a library made by facebook, similar to jQuery
Carousel is a JS library
Framework is complete in itself, it has everything needed to create an app

A framework is a set of pre-written code that provides a structure for developing software applications. A library, on the other hand, is a collection of pre-written code that can be used to perform specific tasks

Both the framework vs library is precoded support programs to develop complex software applications. However, **libraries target a specific functionality, while a framework tries to provide everything required to develop a complete application**.

It takes minimum effort for a library to put in inside our code

What is emmet?
Emmet is a set of plug-ins for text editors that allows for high-speed coding and editing in HTML, XML, XSLT, and other structured code formats via content assist.

How to create a H1 using JS and put it inside a div with id root?
Browser has a JS engine which interprets the code written below and react accodingly

```
  <title>Namaste React</title>
</head>
<body>
  <div id="root"></div>
</body>
<script>

  const heading = document.createElement("h1");

  heading.innerHTML = "Namaste Everyone from JavaScript!";

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

  root.appendChild(heading);
```

What is React CDN?
Content delivery/distribution network
A content delivery network (CDN) is **a network of interconnected servers that speeds up webpage loading for data-heavy applications**. CDN can stand for content delivery network or content distribution network.

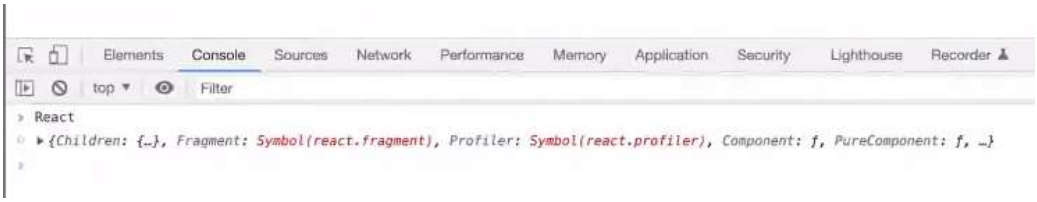
What is cross origin in Script tag?

```
<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>
```

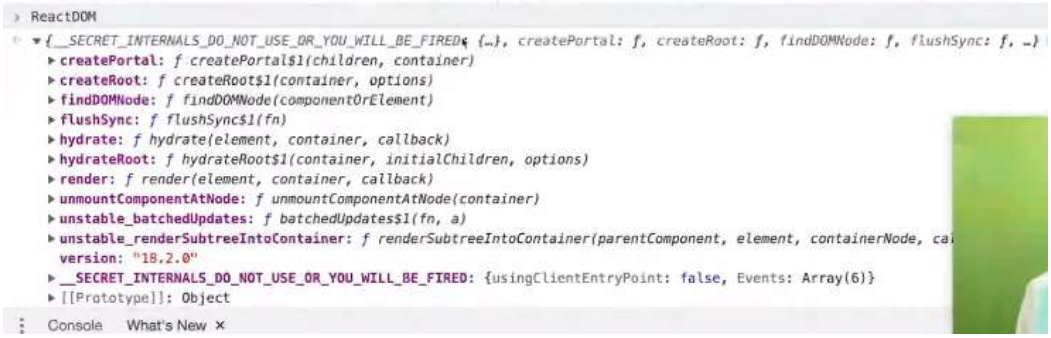
The crossorigin attribute **sets the mode of the request to an HTTP CORS Request**. Web pages often make requests to load resources on other servers. Here is where CORS comes in. A cross-origin request is a request for a resource (e.g. style sheets, iframes, images, fonts, or scripts) from another domain

Shortest Program of React?
The below is shortest program of react, we have just injected react CDN into our document

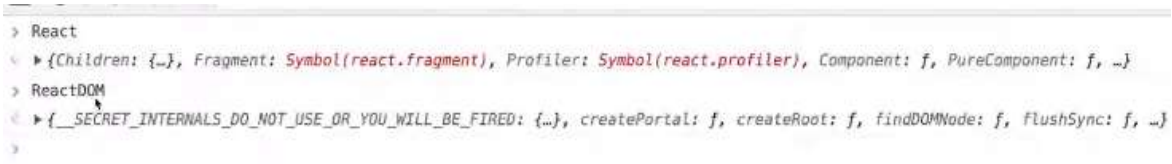
Now if we write React in console we get this



We can also use React.createContext etc etc in console now.
React is a global object and it can be used anywhere using React.something anywhere because we have added react CDN now
We also access to React DOM now



Use of those 2 CDN links were



Why there are 2 files for React and ReactDOM?
React is not limited to browsers only there is React native also for mobiles
ReactDOM means web version of React which gives us access to DOM

Let us use React Now, do same H1 thing using React

We create h1 now like this:

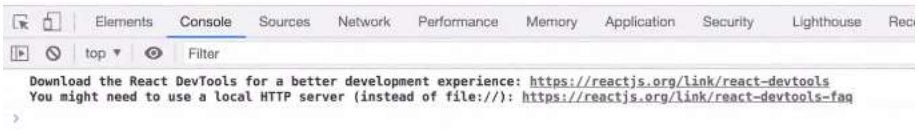
```
<script>

  const heading = React.createElement("h1",{}, "Namaste Everyone!");
```

Now we want to render this heading inside div of id = "root"
We use const root = ReactDOM.createRoot(document.getElementByid('root')) to tell react that this is my root
Now we use root.render(heading)

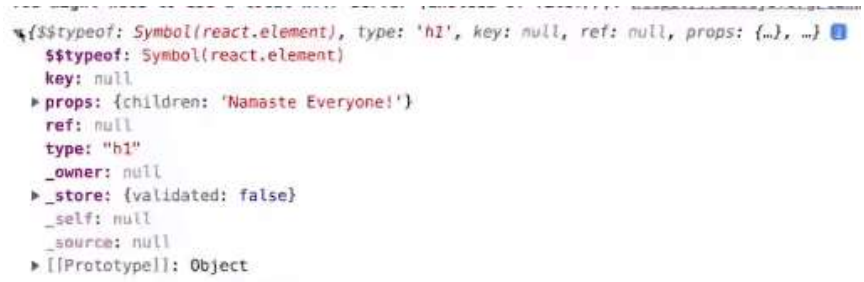
```
<script>
  const heading = React.createElement("h1", {}, "Namaste Everyone!");

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

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

Output is

Namaste Everyone!



If we do console.log(heading)
We see an object of type h1
render() injects element in to DOM



Can we have multiple roots?
Generally in our react app, we have only 1 root and 1 render method

What if we have something on top of div with Id = root?
Let say we make div of id header
And div of Id = root
And div of Id = footer
What will be output?

Everything will run as it is just that React will be rendered inside root only. There can be header or footer also. So we can use react anywhere in our project like search bar, footer etc etc

Header

Namaste Everyone!

Footer

What is { } in React.createElement?

```
const heading = React.createElement("h1", {}, "Namaste Everyone!");
```

Let say we have one more heading with id = title inside div with id = root

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

      <h1 id="title">Hello world</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 heading = React.createElement("h1", {
      id
  }, "Namaste Everyone!");
```

So to make changes in id = title we can pass these parameters inside { }

```
const heading = React.createElement(
  "h1",
  {
    id: "title",
  },
  "Namaste Everyone!"
);
```

Our DOM now looks like

```
<!DOCTYPE html>
<html lang="en">
▶<head>…</head>
▼<body>
  ▼<div id="root">
      <h1 id="title">Namaste Everyone!</h1> == $0
    </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>…</script>
  </body>
</html>
```

Now if we fill root with many h1 headings
Now if we render heading "Namaste Everyone" inside root what will happen?

React will overwrite everyhitng and only Namaste Everyone will be there inside root

```
<body>
  <div id="root">
    <h1 id="title">Hello world</h1>
    <h1 id="title">Hello world</h1>
    <h1 id="title">Hello world</h1>
    <h1 id="title">Hello world</h1>
    <h1 id="title">Hello world</h1>
    <h1 id="title">Hello world</h1>
    <h1 id="title">Hello world</h1>
  </div>
</body>
```

Output

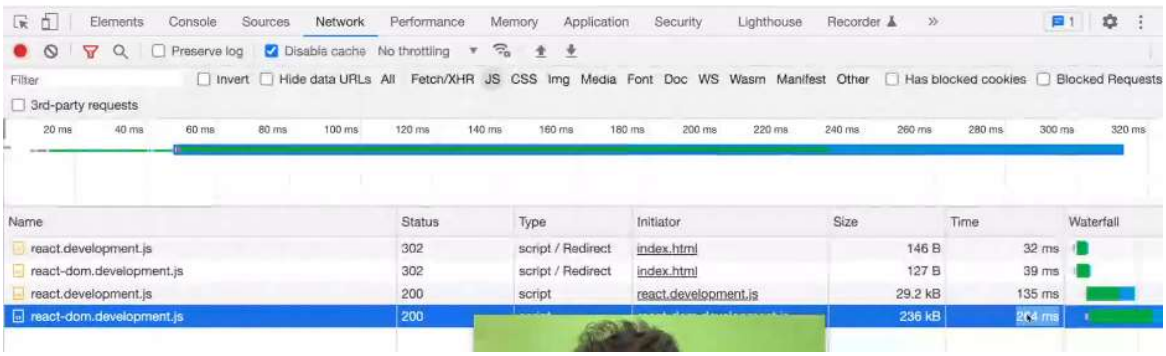
Namaste Everyone!

We generally write "Not Rendered Correctly" inside root div
To make sure that if there is some error we see that message and can solve error that why root not rendered correctly

```
<body>
  <div id="root">Not Rendered</div>
</body>
```

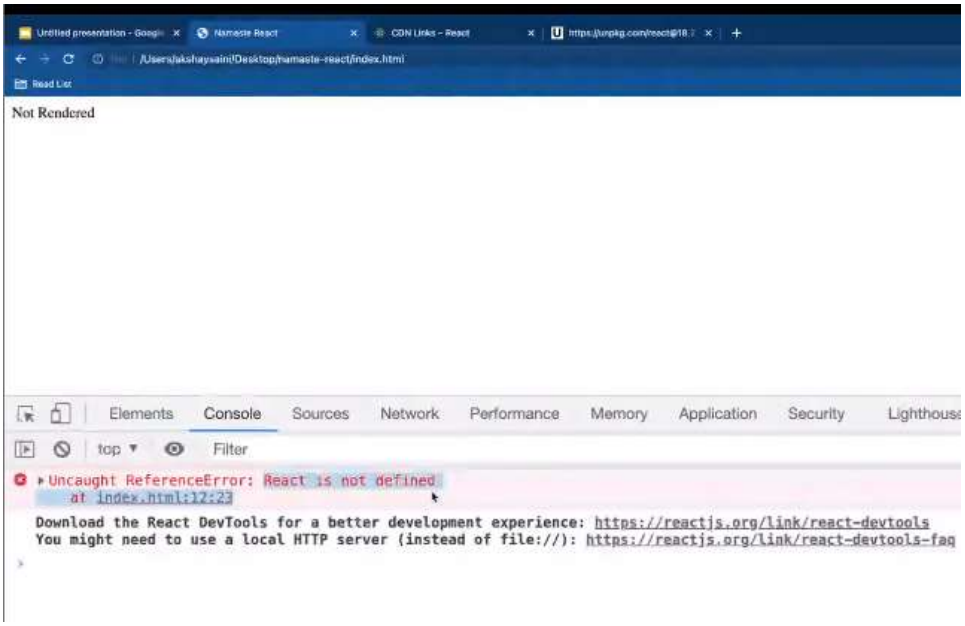
It takes some time for react to render in our browser so we always see "Not Rendered Correctly" for sometime on doing refresh.
It takes sometime for scripts to load

Namaste Everyone!



What if we put script tag inside body below div id = root?
It will not work.

```
<body>
  <div id="root">Not Rendered</div>
  <script>
    const heading = React.createElement(
      "h1",
      {
        id: "title",
      },
      "Namaste Everyone!"
    );

    console.log(heading);

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

    //passing a react element inside the root
    root.render(heading);
  </script>
  <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>
```



What is difference between async/defer?

Async allows your script to run as soon as it's loaded, without blocking other elements on the page. Defer means your script will only execute after the page has finished loading. In most cases, async is the better option — but there are exceptions

Async in script tag is a way to load scripts asynchronously. That means, if a script is async, it will be loaded independently of other scripts on the page, and will not block the page from loading.
If you have a page with several external scripts, loading them all asynchronously can speed up the page load time, because the browser can download and execute them in parallel.

To use async, simply add the async attribute to your script tag:

```
<script async src="script.js"></script>
```

By using the defer attribute in HTML, the browser will load the script only after parsing (loading) the page. This can be helpful if you have a script that is dependent on other scripts, or if you want to improve the loading time of your page by loading scripts after the initial page load.

To use defer, simply add the defer attribute to your script tag:

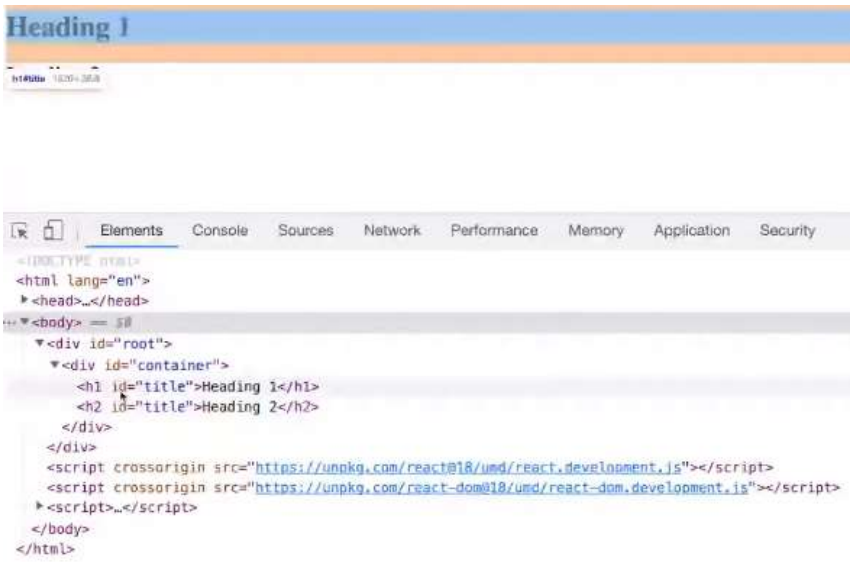
```
<script defer src="script.js"></script>
```

If we want to build the below structure in our HTML using React. How to do it?

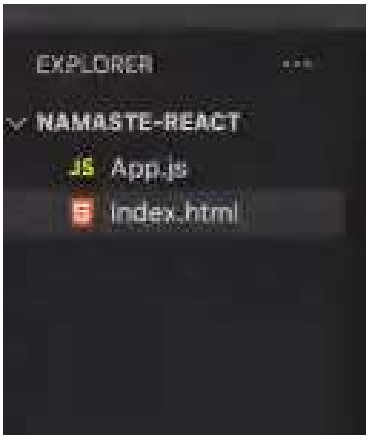
We do it like:
We will pass heading 1 and heading 2 as an Array
We put heading 1 and heading 2 inside container
Now we render the container inside root

```
<script>
  const heading = React.createElement(
    "h1",
    {
      id: "title",
    },
    "Heading 1"
  );

  const heading2 = React.createElement(
    "h2",
    {
      id: "title",
    },
    "Heading 2"
  );

  const container = React.createElement(
    "div",
    {
      id: "container",
    },
    [heading, heading2]
  );

  console.log(heading);

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

  //passing a react element inside the root

  //async defer
  root.render(container);
```

Our DOM looks like:



If we want to build a big index.html
The above method is not development-friendly
React came with a idea to build HTML using JS
We need not to go to HTML anymore, we can do anything from React using APIs like createElement() etc etc.
To make a complex file, its better to split our components and put all JS in app.js



Why do we import CSS inside head in HTML?
seeks to reduce the number of times the browser must re-flow the document by ensuring that the CSS styles are all parsed in the head, before any body elements are introduced.
This is based on the best practice for optimizing browser rendering.

What is rel = stylesheet in link tag while linking CSS in HTML?
The REL attribute is used **to define the relationship between the linked file and the HTML document**. REL=StyleSheet specifies a persistent or preferred style while REL="Alternate StyleSheet" defines an alternate style. A persistent style is one that is always applied when style sheets are enabled.

CDN of react.development.js is for development
CDN of react.production.js has same code but it is much more optimised and for production

Session 2

Revising JS

What is function keyword in JS?
It is present inside JS,

Data Structure used for memory in JS is Heap

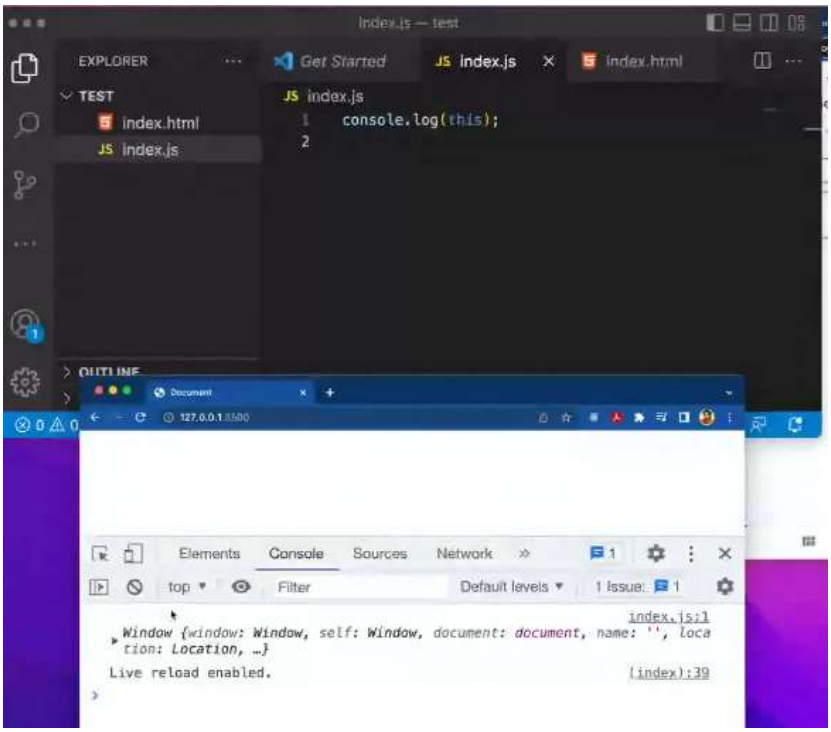
Function without name is anonymous function which can be assigned to a variable as well

```
function x() {
    const a = 10;
}
var xyz = 30;

x(); //functional excution context is created

var x = function () {
    console.log("I'm an anonymous function");
}
```

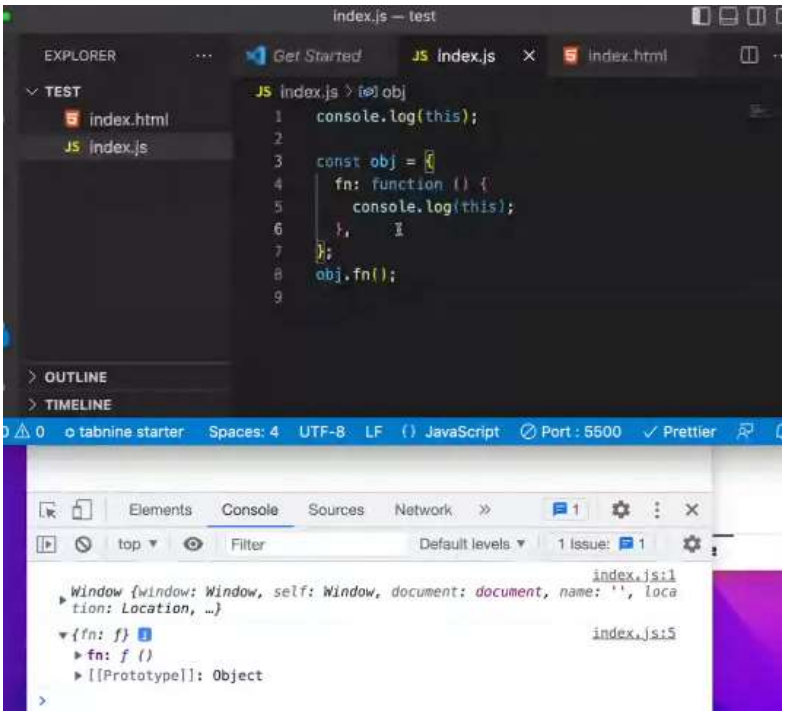
Expression is something that executes. Like console.log is a expression but a = 30 is not a expression.

Arrow function was introduced in ES6 with let, const, promises, spread operator etc etc
=> is known as fat-arrow
Only difference between normal function and arrow function is 'this' keyword
Arrow function, 'this' refers to window

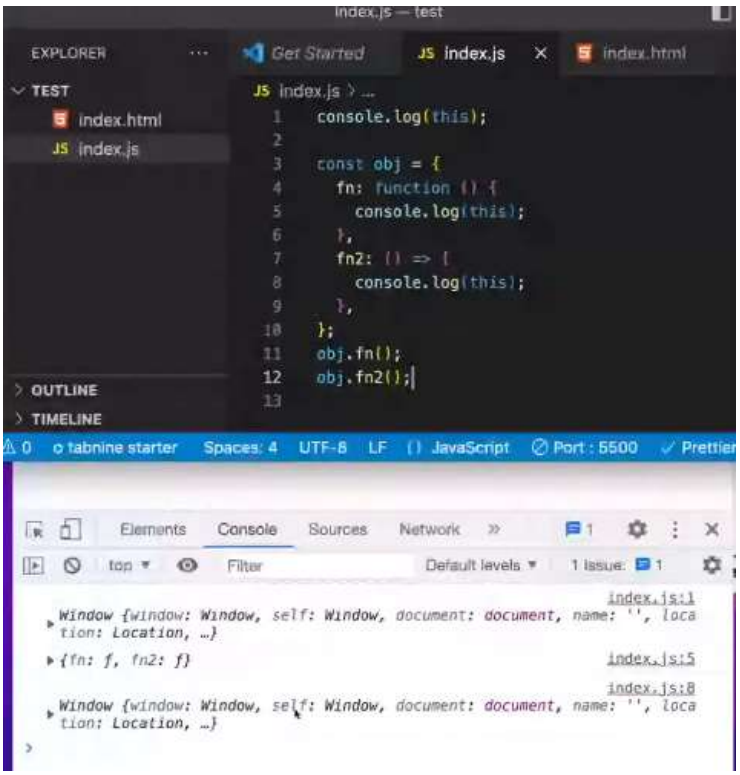
Output of console.log(this) is window object



In normal function 'this' refer to parent object

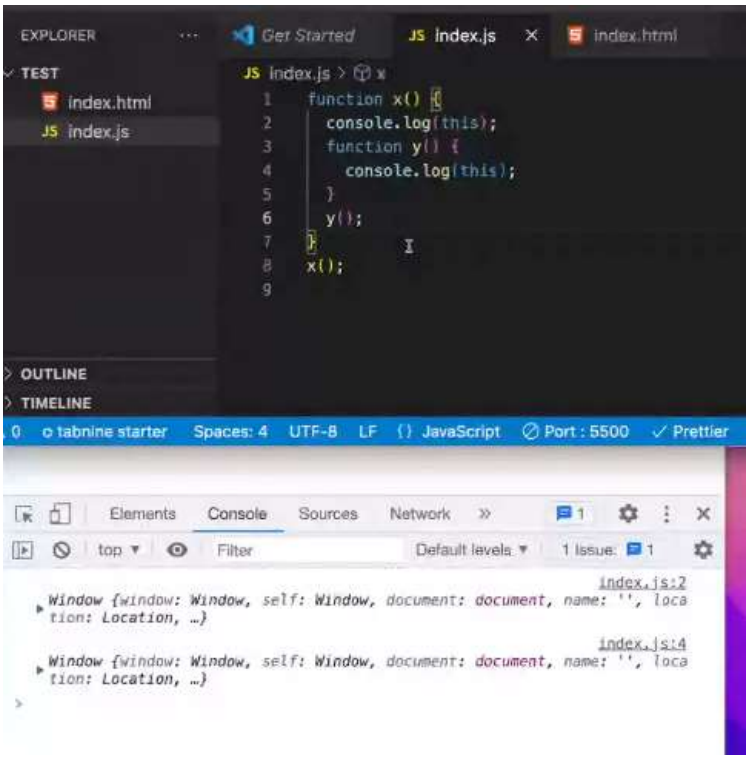


But for arrow function it refers to window object

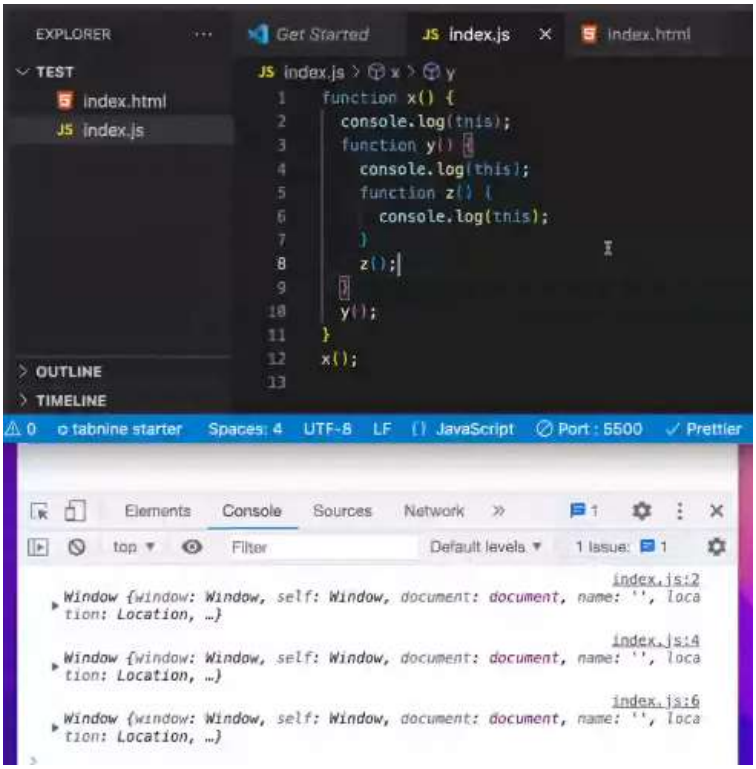


Window is the global object given to us by the browser

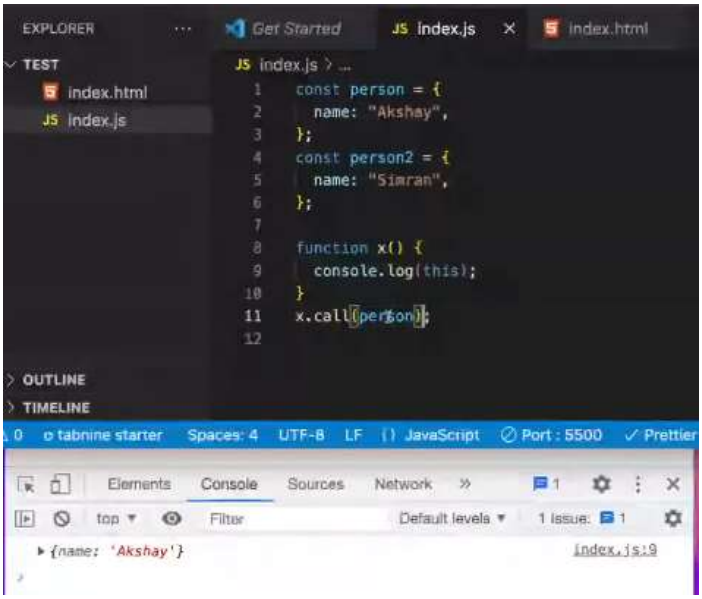
What if we have normal function and another normal function inside it.

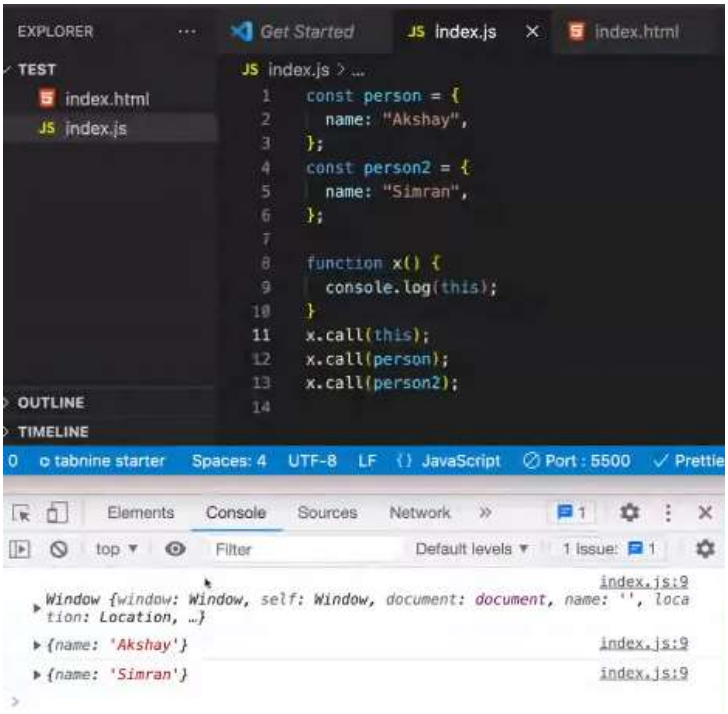


What if we have one more function inside it Z( )

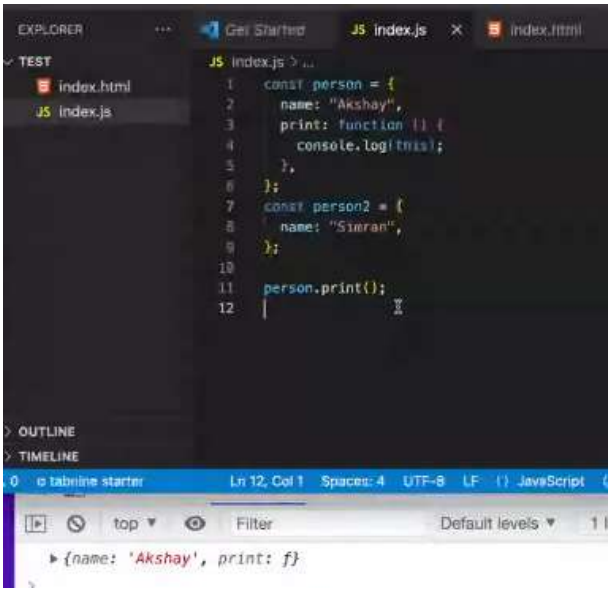


If we make a function and an object.
We call that object using the function. Let's see what happens

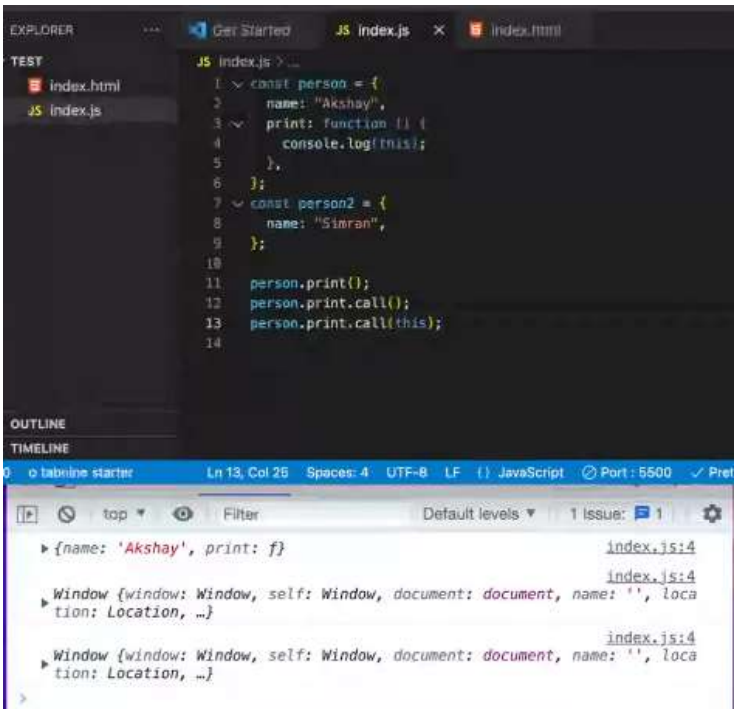




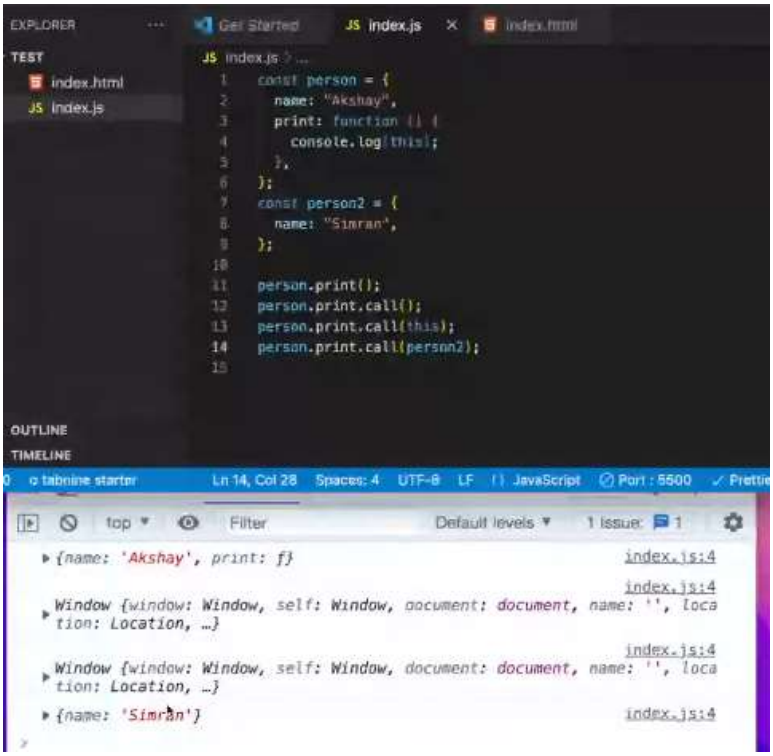
What if we put that function inside object



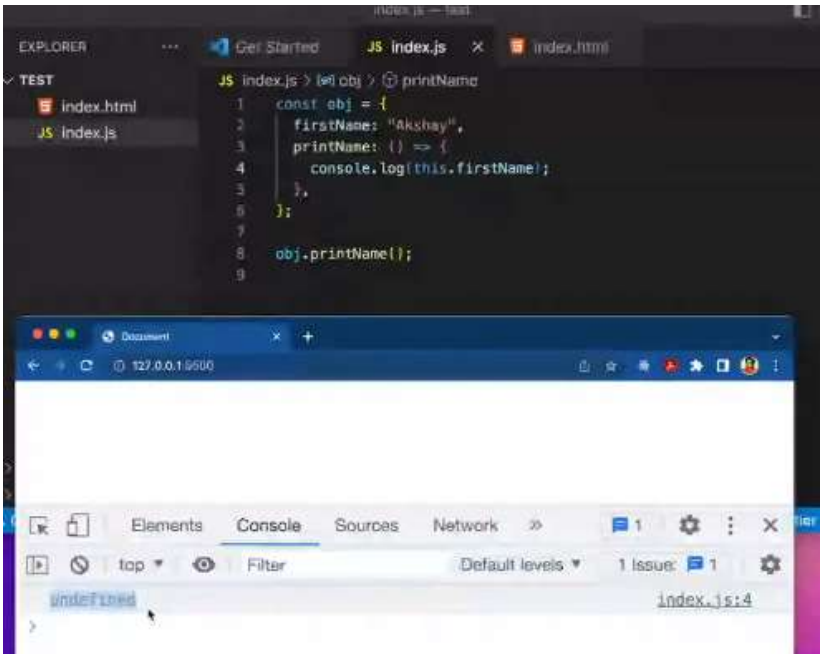
What if we use .call()
call takes window object and then this refers to window object



What if we use .call(object)



This refers to window in arrow function that is why we are getting undefined



Interview Tips

Luck is very important and we cannot control it.
Many Companies don't train their interviewers well
A person can be a good software engineer but bad interviewer

What we can control is:

1. Our preparation
   Technical preparation
   Communication skills: Lot of people fail due to communication skills ,We spend so much time for tech skills but comm skills are equally important.
   Learn to speak while you write, speak your thoughts.
   Practice to speak even when you are coding alone.
   Mock Interviews

2. Talk while you are coding so that interviewer can also see what you are doing.
   If you cannot explain, interviewer thinks you also don't know or you have crammed things.

3. In a company you don't work alone so comm skills should be good. Does not matter how good Software engineer you are, you should be able to communicate your thoughts and ideas to others.

4. Spoken english is very important.

5. Preparation on the interview day (not technical preparation)
   You should not be in panic state before/during interview
   Keep your pen and paper handy
   Keep you water, next to you.
   Keep your laptop charged, keep your charger handy.
   Keep power point near to you.
   These small things mess your interview.
   Keep your camera always open during interview.
   Keep your phone on silent.
   Have a power backup, mini UPS (it does not cost much).

6. Confidence comes from preparation (not just your technical prep) (it includes non tech prep also which are mentioned above).