Web Programming notes-15 Oct

Aria- Accessibility with Rich Internet Applications

- A feature in html used for serving people with disabilities.
- no difference in HTML outputs.

HTML syntax for Navigation Bar

```
<nav>
        <a></a>
</nav>
```

There are two ways we can make our applications accessible to differently-abled people.

- Whenever possible, use **semantic tags**.
- Use of **ARIA attributes**.

Refer: www.sitepoint.com/how-to-use-aria-effectively-with-html5/ for further details.

Semantic tags are those tags that have a special meaning to the tag name.

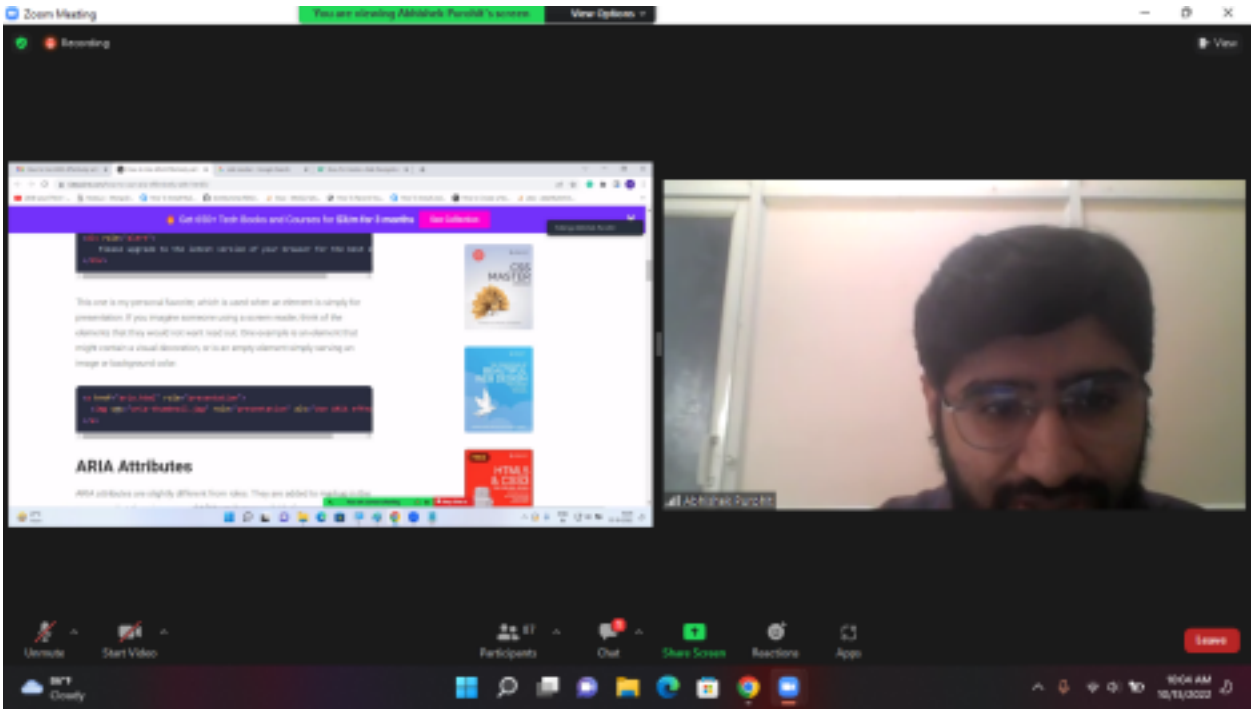
We can also use <footer> tags.
Ex:
<footer>
</footer>

ARIA Attributes:

- role: For any purpose , we are not getting any semantic tag, then just apply the function role to regular html tags/alignments.

Someone commented on your photo

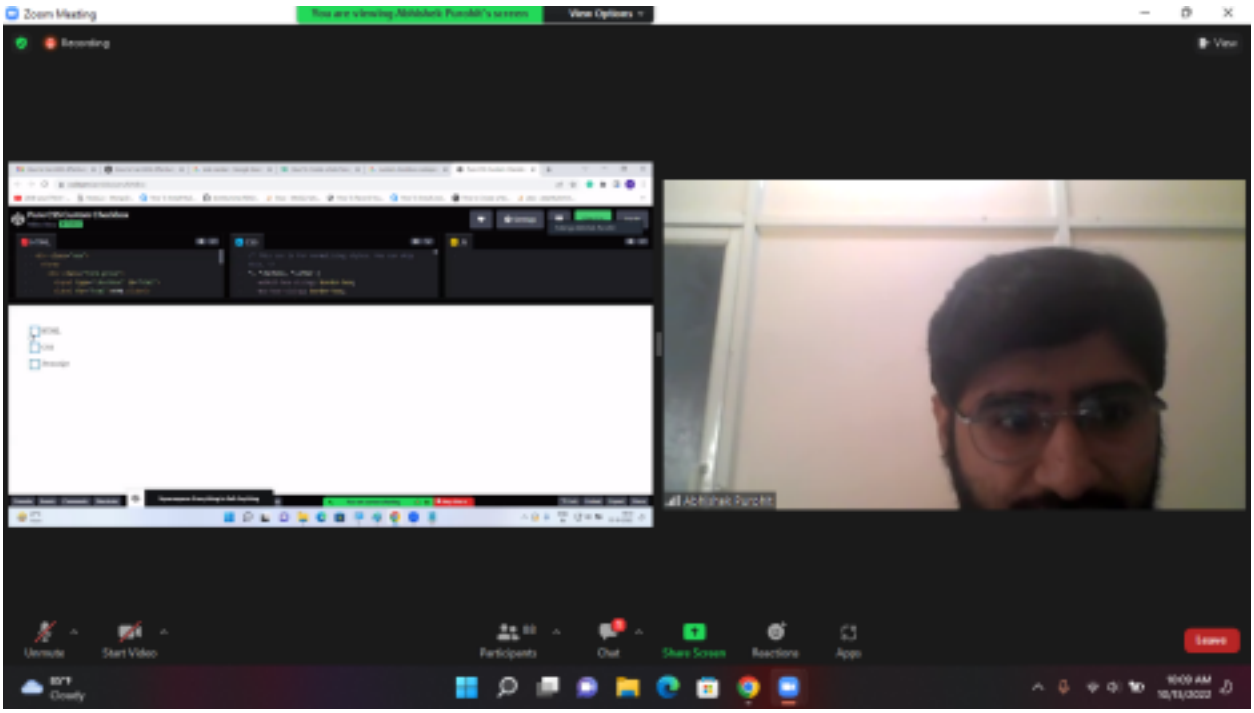
<nav role="side-navbar">
</nav>



Whenever we are displaying radio buttons, we use:
<input type=”radio”>
<input type=”checkbox”>.
But there are cases where a developer can create their own custom radio buttons and checkboxes.

Ex: For creating a custom checkbox:

A custom checkbox can be created by using ** ** tag.



Two types of aria- attributes:

- **States** - something that will change (variable).
- **Property** - something that will not change (constant).

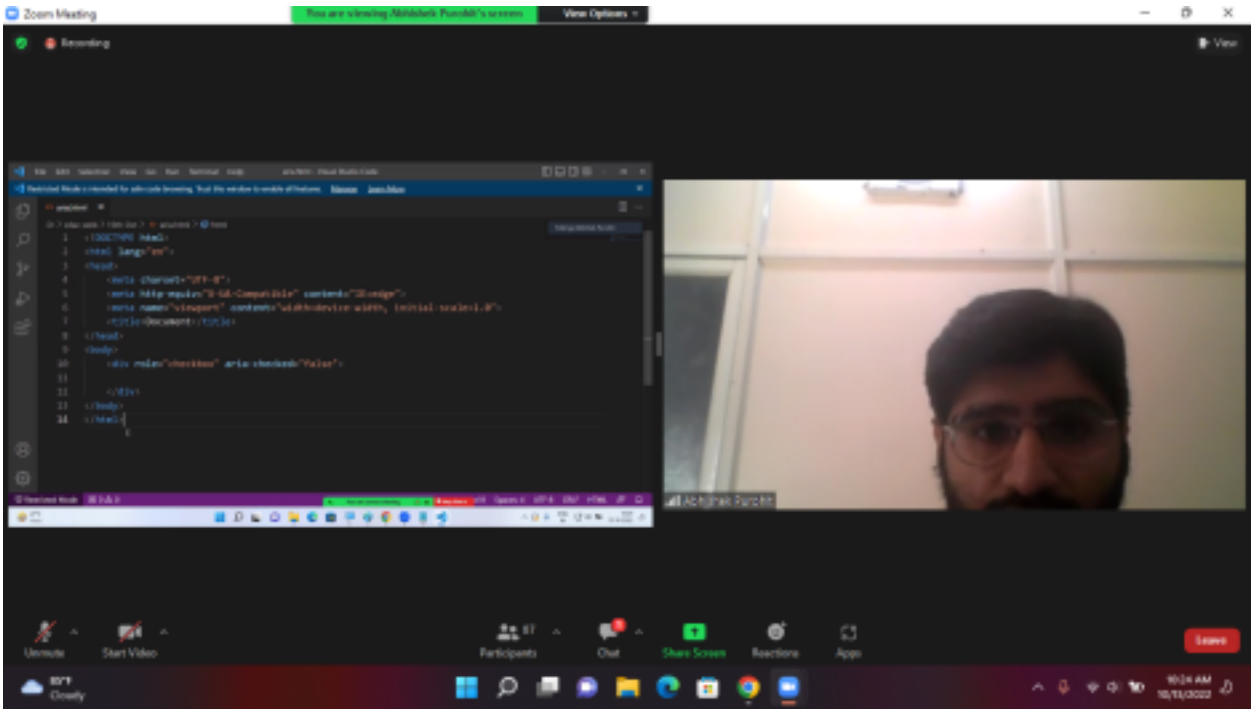
*Ex:**Checkboxes for Hobbies:***

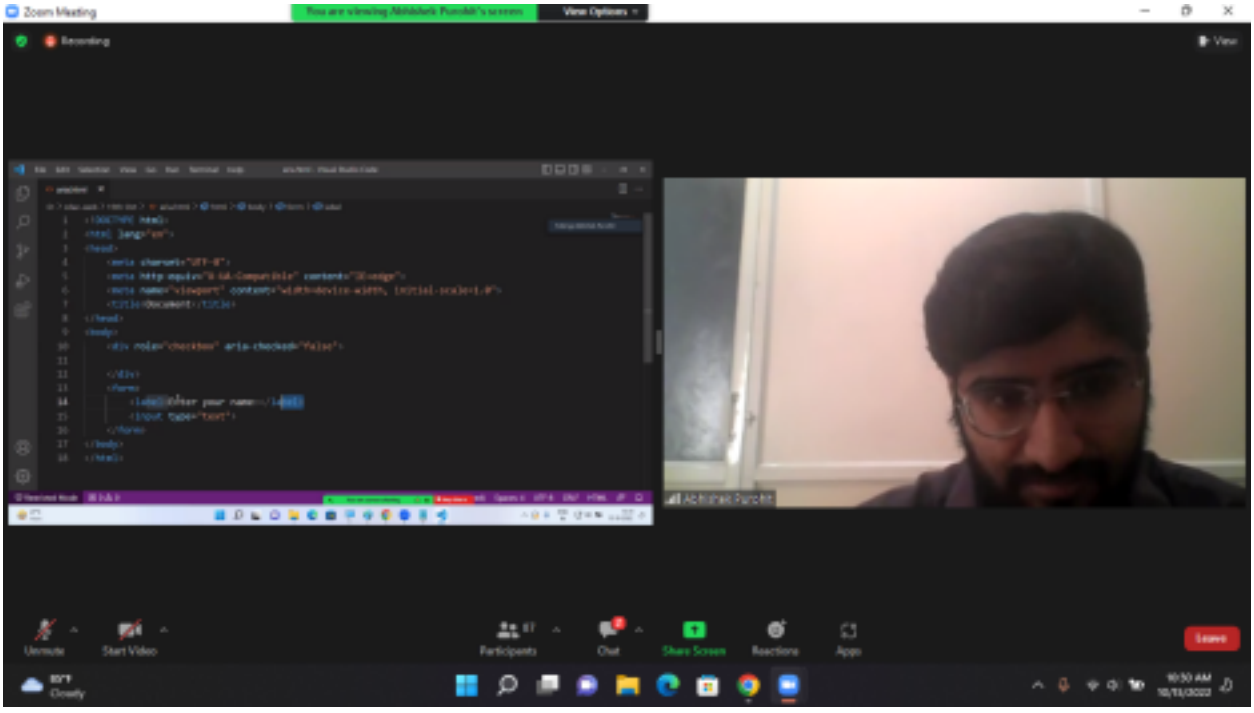
- Property : hobby
- State : checked /unchecked.

aria-checked - gives **semantic meaning to user-defined functions.tags**.

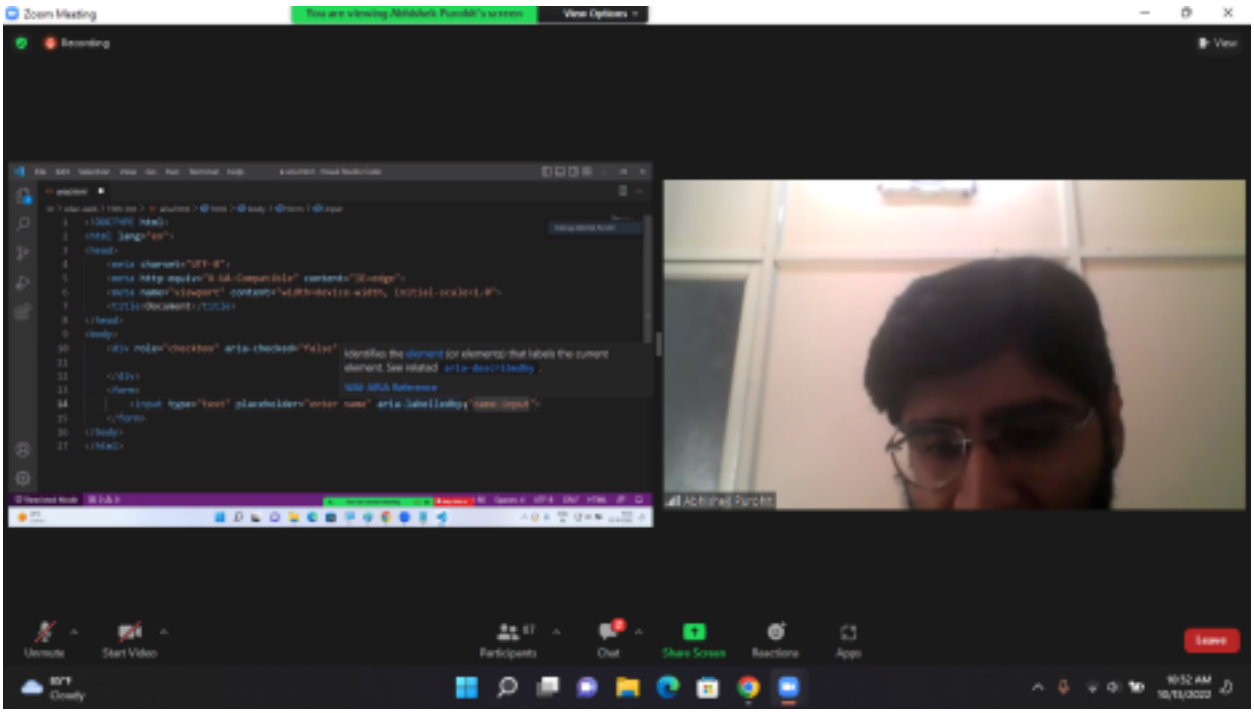
Syntax for aria-checked

** **





<input type=”text” placeholder=”Enter name” aria-labelledby=”name-input”>



<input type=”text” placeholder=”Enter name” aria-labelledby=”name-reader.”>

aria-checked is a **state** while **aria- labelledby** is a **property.**

Ex: ** **
** **
**<div div role=”checkbox” aria-checked=”true”> **
** **
** **

Client -Server Architecture.

1. We cannot store the entire data into the client. So the server comes into the picture to store the data permanently.

   In the sign up mechanism, the **role of the client** is to **display the form as well as send data to the server.**
   The **role of the server** is to **read the data received from the client.**

   There is some data exchange as well as communication between the client and server.
   For effective communication between the client and server, they have to follow a set of rules. These rules are called **protocols** .

1. **HTTP Protocols**:

HTTP => Hyper Text Transfer Protocol.

HTTPS=> Secure version of HTTP. It is recommended that while hosting an application on any website or while banking online, we must check the protocol. It is best if we use sites that follow https:// protocol.

HTTP can be used for local applications.

HTTP Request - Response mechanism:

Sending a request to server :

Requirement of **IP address**- IP address is the **address used to recognize the target server on the internet.**

To simplify the process of storing the IP address on your machine, they brought in the concept of **domain name**.

Ex for domain names: www.facebook.com

We need the IP address/url of that server.

Every request must have a type/method:

Whenever we are sending request to the server, along with the url, a request method must be present (For what purpose we are sending the request to the server). **Request methods are mandatory.**

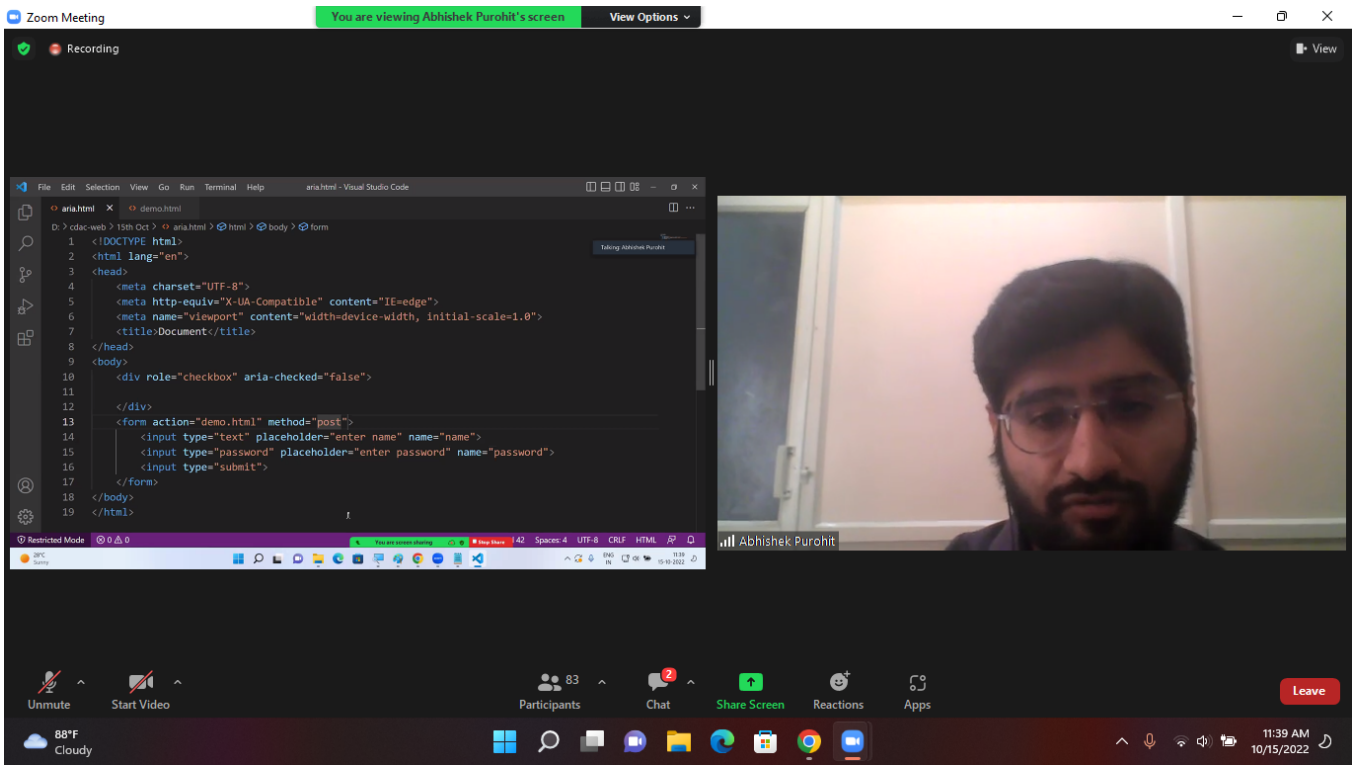
Types of Request methods:

GET :

Getting data from the server.

POST:

To send data on the server to create a new resource on the server. The post request will maintain the form data in the request body.



PUT:

To send a request which will **update the entire data on the server.**

Patch:

Used to update **a small part of the data**.

DELETE:

To send a request which will **delete the resource from the server.**

NB: Whenever we send a request from the client application, we must specify the request method.If we haven't specified any method, the GET request will be considered for action from this url..

Refer on internet: idempotent request.

SERVER:

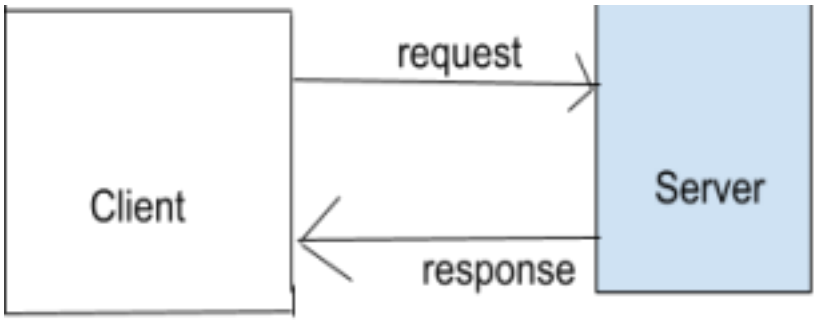
- Infinitely listens to the requests.
- Accept the requests
- Process the requests
- Must give a response.

HTTP Response may have some data. **Along with data, HTTPResponse must have a status**

code.

The status code specifies the current state of the server/ what actually happened on the server side. It is the way of sending the state of the server.

- Server has responded successfully.
- Server has an error.
- Request not found
- Server is busy.
- The Status code will always be of integer value.
- Any status code that has **4 as a prefix** , then it is the **fault of the client.** ● Any status code that has **2 as a prefix** , then it is a successful operation.



Ex: **404- Not Found**
400- Bad request.
401- unauthorized.
200- OK
201- CREATED.

Refer: HTTP status codes.

-