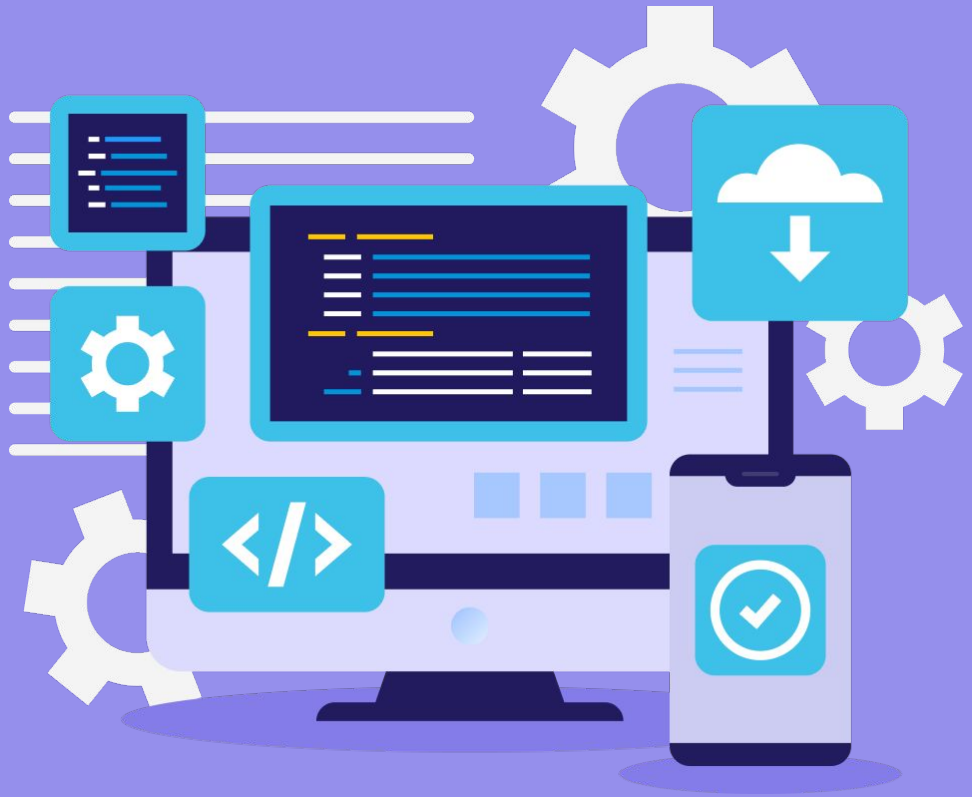
App Building: Static Ecommerce Website

Relevel
by Unacademy



Topics Covered

Introduction
What is a user interface? What are we building today?

Conditional Rendering
How to display information based on conditions?

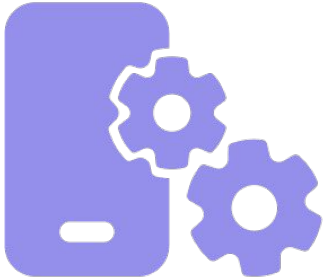
Folder Structure
What are the different ways of organizing files?

Creating our UI
We will be developing our frontend based on everything we have learnt so far.

Practice Questions
Assignments

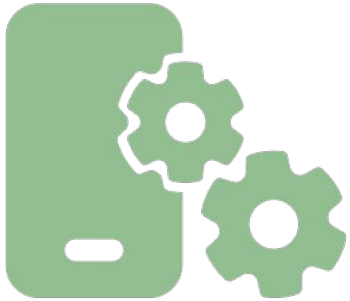
User Interface

- A user interface is a place where humans can see information & interact with it.
- Some common examples are web browsers, smartphones, desktop applications etc.
- User interfaces display what the user is supposed to see while conceding the complex business logic.
- It serves as a gateway between human beings and computers.

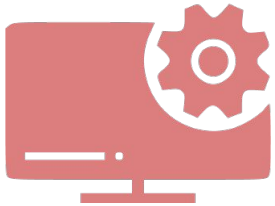


User Interface

- In this lesson we will be creating frontend or UI .
- We will be making use of HTML, CSS & JS knowledge to create frontend.
- We will also be organizing all the files in proper folder structure.

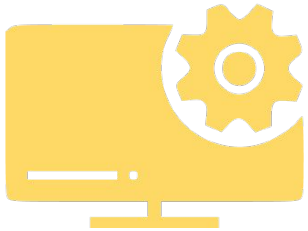


Conditional Rendering



- It is a concept where elements are displayed on the UI after meeting certain conditions.
- It is used to add dynamic behavior to the application and ensure validations are properly done.
- E.g. Facebook allows only 13+ aged people to open an account, so if the user's age is 13+, the user account screen will be displayed; else, a warning will be displayed.

Conditional Rendering



To demonstrate conditional rendering, we will create an HTML file and add a JS function there.

- The HTML file will have a button with onclick attribute.
- The JavaScript function will have conditionally render two different buttons.

HTML File

- We will create an index.html file.
- We will add an h1 tag containing ‘Welcome to Unacademy’.
- We will add a p tag containing ‘Click on the button to witness conditional rendering’.
- We will add a button ‘Click Me’.

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
<h1>Welcome to Unacademy</h1>
<p>Click on the button to witness
conditional rendering</p>
<button>Click Me</button>
</body>
</html>
```

JavaScript Function

- Now we will add a script tag.
- We will write a function conditional that will receive a parameter called num.
- We will declare two variables button & button2 and use JavaScript's document.createElement("BUTTON") method to dynamically create two buttons.
- We will then set "First thing" & "Second thing" as the labels of the first and second buttons respectively using the innerHTML attribute.
- We will set a condition if the value of num is less than 10 we will add the first button to our web page using document.body.appendChild(button).
- If the condition is not met then we will add the second button.

```
<script>
function conditional(num){
let button =
document.createElement("BUTTON");
let button2 =
document.createElement("BUTTON");
button.innerHTML="First thing";
button2.innerHTML="Second thing";
if(num<10){
document.body.appendChild(button)
}
else{
document.body.appendChild(button2)
}
}
</script>
```

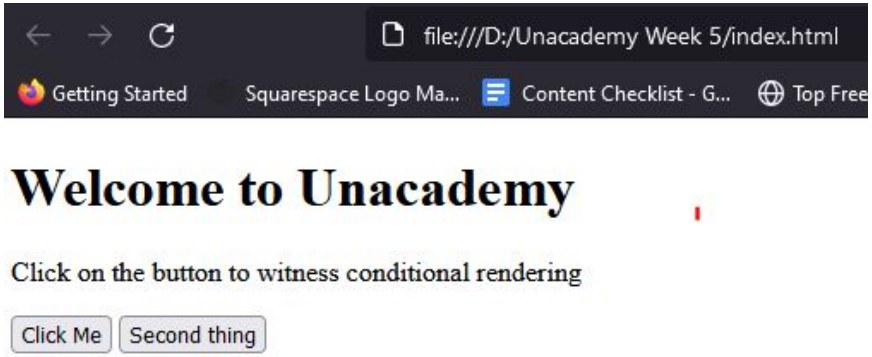
Modifying HTML

- We will write an onclick attribute for our button and provide conditional(12) as the onclick value.
- So everytime the button Click Me is clicked by the user, the onclick attribute with recognize it and call the function conditional and pass 12 as the argument.

```
<body>
<h1>Welcome to Unacademy</h1>
<p>Click on the button to witness
conditional rendering</p>
<button
onclick="conditional(12)">Click
Me</button>
</body>
</html>
```

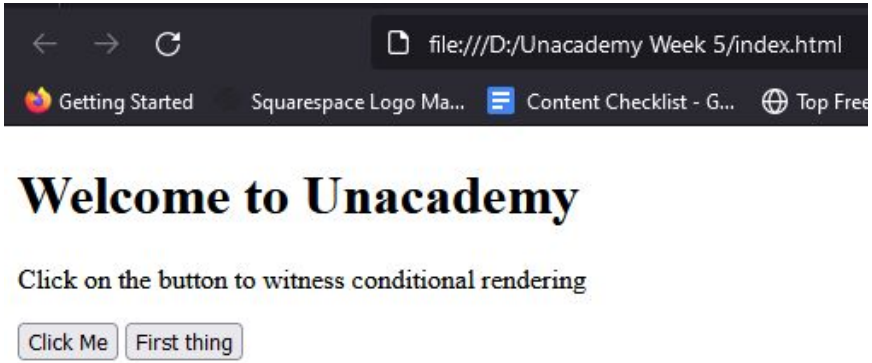
Conditional Rendering

- Now we will click on the Click Me button.
- We can see that the button with label ‘Second thing’ as appeared on the UI.
- This happened because we have passed 12 as the argument for conditional() function.



Conditional Rendering

- We will change the argument from 12 to 9.
- <button onclick="conditional(9)">Click Me</button>
- We will refresh the page.
- We will click on the ‘Click Me’ button
- We can see that the button with label ‘First thing’ as appeared on the UI.
- This happened because we have passed 9 as the argument for conditional() function.

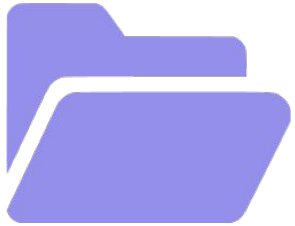


Folder Structure

- Every web project has a number of Javascript, Html & CSS files, these files compose and contribute to the user interface and business logic of the project.
- Whenever the project grows, the number of files grows along with it.
- If these files aren't organised they become cluttered and challenging to maintain.
- In software companies, in one project, multiple people work.
- Different files are created by the people and over time it becomes complex to understand their purpose.

Folder Structure

- For everybody in the project to understand the purpose of the files there needs to be a structure accepted and acknowledged by the project peers.
- Thus we follow certain methods to group these files in folders, so that it becomes easier to track and maintain them.
- There is no hard and fast rule on maintaining folder structure. Depending on the project, the folder structure keeps changing, but we have certain practises that we can follow or draw inspiration from.



Best Practices

Group the files properly:

- Keep the base folder as “src” or “root”.
- The files that don’t have any dependency on other files should be kept here.
- Keep all JS files in one folder and CSS files in one.
- Keep files pertaining to one feature in one folder.
    - E.g. For login page, create a folder called Login and keep the essential files there.

Best Practices

- Avoid Nesting:
    - It is a cumbersome and ugly practice to have too many sub-folders and files in them, it becomes challenging to search for a particular file if there are many levels of nesting.
    - Nesting causes major issues when a deeply nested file has to be imported, for this a large file reference will be required.

```
src/
        ----login/
                ----username/
                        ---index.html
                        ---index.css
                        ---index.js
                ----password/
                        ---index.html
                        ---index.css
                        ---index.js
```

Best Practices

Avoid Nesting:

- We can flatten the above by removing the unnecessary password folder and adding the three files in one folder, like this:

```
src/
        ----login/
                ---index.html
                ---index.css
                ---index.js
```

Best Practices

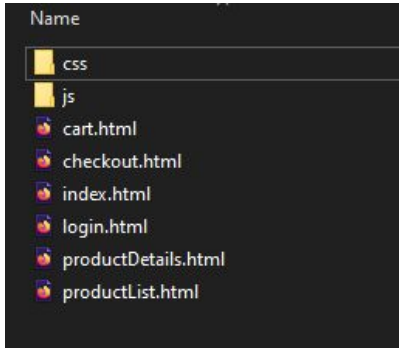
Avoid creating folders for housing just one file:

- If there is only one file, try to keep it in the root folder or club it with other folders containing multiple files, do not create a folder to contain just one file or a few files as it makes the folder structure nested and unnecessary complex.

Let’s create our Frontend

Base Folder

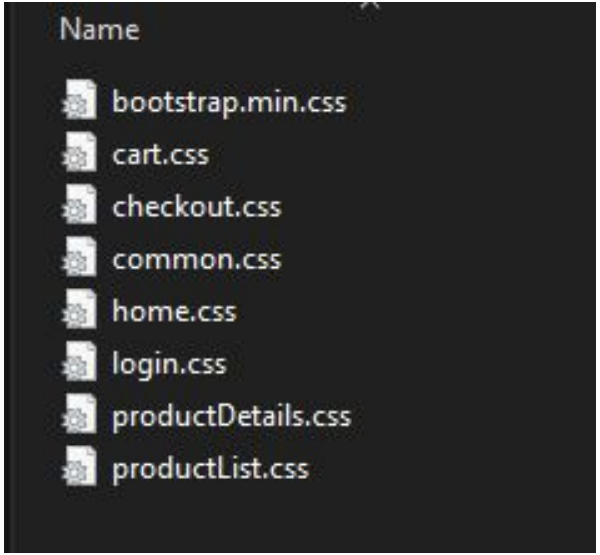
- We will create a folder called ecommerce-bootstrap.
- This will serve as a root folder..
- All the non-dependent files will be stored here.
- We will create two separate folders called css & js to contain our css & JavaScript files.
- The following HTML files will be stored here:
  cart.html, checkout.html, index.html, login.html, productDetails.html, productList.html



CSS

Inside the css folder we will store the following files:

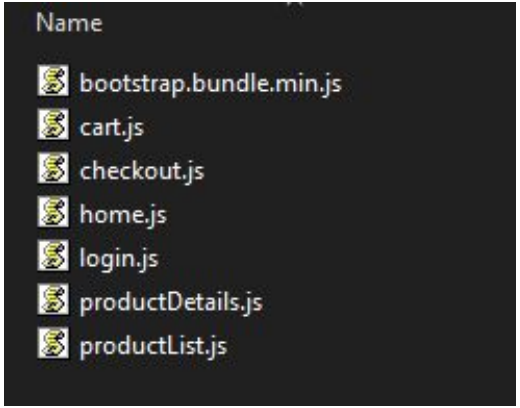
- bootstrap.min.css
- cart.css
- checkout.css
- common.css
- home.css
- login.css
- productDetails.css
- productList.css



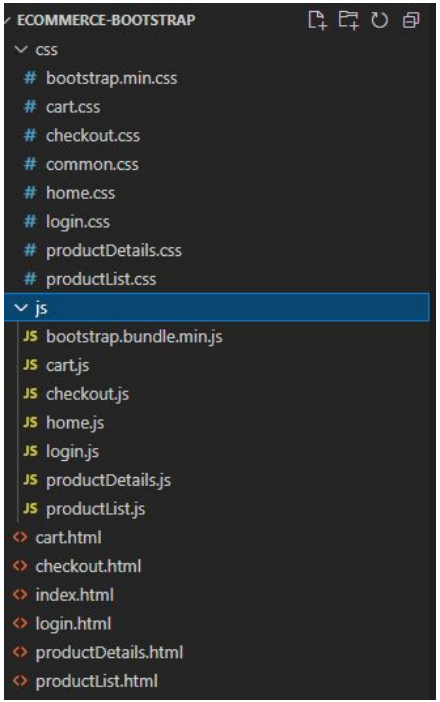
JavaScript

We will be storing the following files inside the js folder:

- bootstrap.bundle.min.js
- cart.js
- checkout.js
- home.js
- login.js
- productDetails.js
- productList.js



This is how the folder structure would appear on any IDE:



Assignment

Compile the files and load the UI of ecommerce app on browser.

Thank You