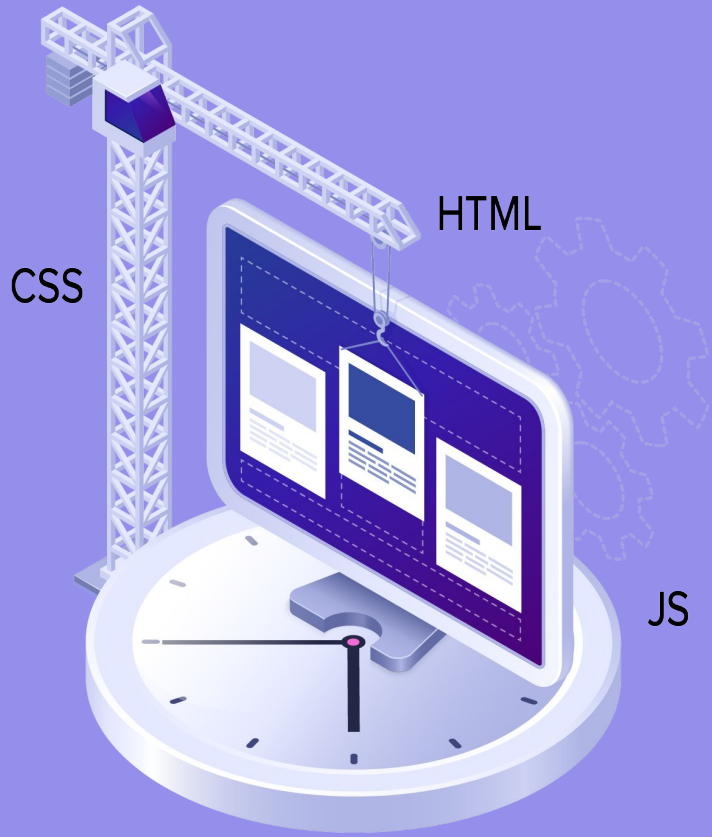
Countdown timer app

Relevel
by Unacademy



Previous class (5 mins):

Please recap the following concepts taught in the previous session


Arrays


Objects


JSON


Properties


Destructuring


forEach()

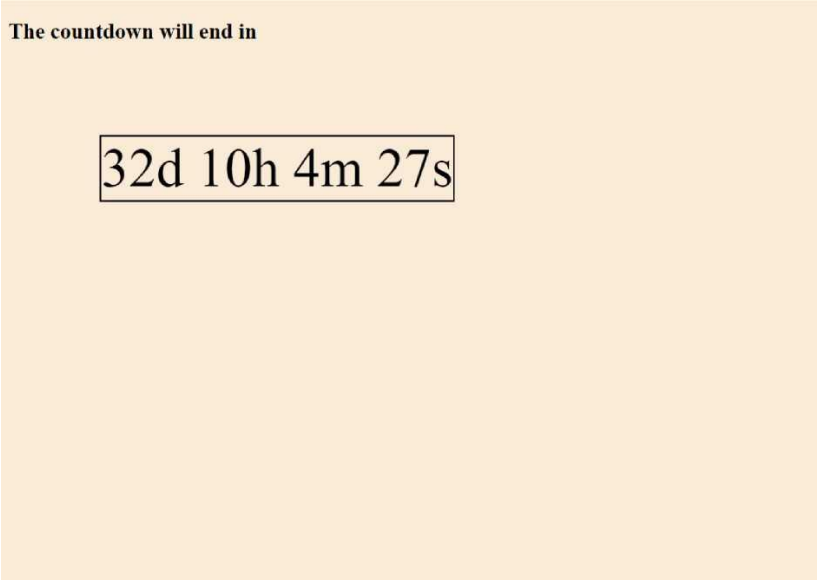

map()


filter()


Rest parameters

App Introduction (5-10 mins):

We will be building a simple countdown timer app that will set a timer from the time we are creating the timer to the time or date that is provided as shown below



List of topic content (5 mins):

Async programming

Callbacks

Promises

Timers

Async programming in javascript:(25-30 mins)

- Before you start async programming you should know the basics of javascript function.
- JavaScript is a function-oriented language. One can create a function dynamically, copy it to another variable, or pass it as an argument to another function and call from a different place later.
- Functions are the main "building blocks" of the program. They allow the code to be called many times without repetition.

Asynchronous programming means the execution of our code or function takes place simultaneously while another part of a function or another function is running in the background
This method helps us a lot like it does not stop the execution of a further code until the result of the current function is not ready like in synchronous programming

Callback (30-35 mins):

In javascript, Callback is just a method that we assign in another method as an argument.

Example:

```
function myDisplayer(some) {
document.getElementById("demo").innerHTML = some;
}

function myCalculator(num1, num2) { let sum =
num1 + num2;
return sum;
}

let result = myCalculator(5, 5);
myDisplayer(result);
```

Explanation:

Here we have created a function myDisplayer which takes one argument in and just target element in our HTML by id and assigns the HTML which is passed to it as an argument

After then we have also created another function named myCalculator Which takes three arguments num1, num2 the numbers and the third argument is another function mycallback

This is called a callback function

When we call our function myCalculator it executes and then our callback get executes
These callback functions are used in asynchronous programming

Callback hell

Callback hell is major issue caused during the callback function it occurs due to calling multiple callback function nested within itself this will affect the maintainability of our code and it will become hard to read the code

```
1
2     /-unction getContinents(cutliracfe) {
3     setTimeout(function{) {
4     Let continent = 'Asia'; callback(continent)
}, 2000)
1 }
8
9 /unction getCountries(continent, callback) {
10     setTimeout(/unction() {
Let countries = ["India"* "Pakistan",
"Bangladesh", "Sri Lanka", "Afghanistan"];
callback(countries);
13     }, 2000)
14     }
15
16     /unction getStates(country* callback) {
17     setTimeout(function() {
18     Let states - ["UP", "HP"* "AP"* "TK"];
```

```
21    }
22

_
1
2
3- function getcities(state, callback) {
4 •     setTimeout(function() {
let cities = ["Lucknow"* "Pilibhit”* "Prayagraj”* "DDU"]j callback(cities);
7     »
8     }
9
10
11     getContinents(/unction(continent) {
12     - getCountries(continent, /unction(countries) {
13 -     getStates(countries[0]* function(states) {
14-     getCities(states[0]* /unction(cities) {
```

```
15      console.log(cities)
16      })
17      })
18      })
19      »;
20
```

In the above example we have created many function for continent ,countries,states and cities
Now if we want the city from any continent followed by country and state we have called functions inside function for multiple times this will create a callback hell

If there is error at certain point in any of the function the main function will also get affected
To overcome this issues there is a method called promises in javascript

Promises (25-30 mins):

Promises are the new style of writing code which depends on other function or when we need to call certain function after executing another function we use promises Like in Fetch API,

API

Apis are basically a method for connecting our front-end or client side to server or the backend which is used to build softwares ,web sites or any other application with the feature of storing ,accessing data from servers Some REST APIs are get, post, put, delete, and many more

We will illustrate the concept of promises in an example:

```
var promise = new
Promise(function(resolvej reject) {
const x = "hello world"; const y =
"hello world" if(x === y) { resolve();
} else { reject<);
}
});
promise.
then(function () {
console.log("Success, You are a GEEK');
» '
catch(function () {
console.log('Some error has occurred');
»;
```

Explanation:

Here we have passed a function inside the promise which will basically check the given condition and either mark the function as resolved or rejected

Further to handle the state returned by promise is handled by .then() and catch() methods this methods are the methods to handle promises after knowing the state of promises promises return only three state

1. Pending: it is the initial state of promises
2. Resolved: it shows that our promise is completed
3. Rejected: means promise is failed

When the promise is pending it means our function is initiated When our promise get resolved we can perform further action on that promise or result by using the .then() When there is an error in our code or something else due to which our promise get rejected that error is handled in the .catch() block which specifies more about the bug

Async-awaits:

Async await is an method used in ES6 version of javascript for maintaining the work flow of functions basically this is a method which gives the sense of synchronous programming And synchronous is defined as the continuous execution or flow of the function
It is the best practice to write our API call or promises in async-await

This method works as first the Await will stop the execution of the function which follows it till the promises are resolved, same as with a synchronous operation. This allows other tasks or functions to run in the background but the code awaited is blocked. For example:

```
Complexity is 3 Everything is cool! async function
makeResult(items) { I let newArr = [];
for(let i=0; i < items.length; i++) {
newArr.push('word_'+i);
>
return newArr;
}
async function getResult() {
let result = await makeResult(items); // Blocked on this
line useThatResult(result); // Will not be executed before
makeResult() is done
```

In the given example the variable result is awaited while will block the execution of current code but other functions in the application will keep executing in background

And the useThatResult( ) function will execute after the execution of the makeResult function

Timer (25-30 mins):

Timers are nothing but the event which occurs at a specific time intervals

Generally used timer methods are

- setTimeout(function,time)
- setInterval(function,time)
- clearInterval(id)
- clearTimeout(id)

1. setTimeout()

This function start execution after the provided time, which is provided in milliseconds

This function takes two parameters among which the first is a function that is to be executed and the second argument is time

Example:

On clicking a button alert will be shown after 3 seconds

```
<button onclick="setTimeout(myFunction,

3000)">Try it</button> <script>

function myFunction() { alert('Hello');
>
@/script0
```

2. setInterval()

This function repeats the execution of the given function over the interval of the time which is provided in milliseconds

This function takes two parameters among which the first is a function that is to be executed and the second argument is time

Example:
Display the current time

```
setInterval(myTimer, 1000);

function myTimer() { const d = new Date();
 document.getElementByld("demo").innerHTML
         = d.toLocaleTimeString();
0
```

Output:

This function will execute continuously after 1 sec and give the current time

3. clearInterval()

In case we need to stop the execution of this function there is another method called clearInterval()
This function is used to stop the continuous execution of a setinterval function

Example:

let id= setInterval(function,time);
clearInterval(id);

```
setInterval(myTimer, 1000);

function myTimer() { const d = new Date();
 document.getElementByld("demo").innerHTML
         = d.toLocaleTimeString();
0
```

4.clearTimeout()

To stop the execution of setTimeout() function clearTimeout() method is used basically it stops the the execution of the setTimeout function this is done by passing the id returned from a function which we want to stop

Example:

let id= setTimeout(function,time); clearTimeout(id);

Create an App for API calling and async programming: - (30 mins)

See the app here
We will see an example of the above topics such as API calls, asynchronous programming

Example:
We will fetch the data of the current covid patients from an API

Html

Press! + enter to bring up the basic boilerplate code of HTML. Discuss how to create a webpage:

```
!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Document</title>
</head>
<style>
body {
background-color: Iantiquewhite;
>
h3,h2 0
text-align: center; font-size: 60px; margin-top: 100px; border: 2px solid Clblack;
width: fit-content; margin-left: 100px;
```

```
I
</style>
<body>
<div id="container">
<h3>Total Covid Cases</h3>
<h2 id="total-cases"X/h2>
</div>
<script src="index.js"></script>
</body>
</html>
```

CSS

- You can style your form by adding code in styles.css or you can style internally by using a style tag. Eg: <style></style>.

JS

- Write a function to fetch the data from an API
- This function will be written in async and await to follow the asynchronous programming
- After this handle the promises by using .then() method
- Load the data from API to our application
- Output the result in an HTML element
- Check if the output is valid or not

Implement the code as shown below

```
console.logCindex.html 7 | Get Covid Data");

const getCovidData = async () => {
console.logCindex.html 10 | Processing...");
const request = await fetch("https://covid19.mathdro.id/api");
const data = await request.json();
return data;
};

getCovidData().then(covidData => {
console.logCindex.html 16 | covid data", covidData);
document.getElementByld("total-cases").innerText = covidData.confirmed.value;
});
```

Create a countdown timer app: - (30 mins)

Create a simple HTML page for displaying the timer

We will see an example of the above topics such as timers

App link here

HTML

- Press! + enter to bring up the basic boilerplate code of HTML.
- Discuss how to create a webpage:

CSS

- You can style your form by adding code in styles.css or you can style internally by using a style tag. Eg: <style></style>.

JS

Write a function to construct a new date constructor
Assign the date up to which you want to set the timer for
Calculate the time for days, hours, minutes, second
Output the result in an HTML element
Check if the count is valid or not

Implement the code as shown below

```
window.onload = function () {

var seconds = 00; var tens = 00;

var appendTens = document.getElementById("tens") var appendSeconds =

document.getElementById("seconds") var buttonStart =

document.getElementById('button-start'); var buttonStop =

document.getElementById('button-stop'); var buttonReset =

document.getElementById('button-reset'); var Interval ;
```

```
buttonStart.onclick = function() {

clearlnterval(Interval);
Interval = setInterval(startTimer, 10);
}

buttonStop.onclick = function() { clearlnterval(Interval);
}


buttonReset.onclick = function() { clearlnterval(Interval); tens = "00"; seconds = "00"; appendTens.innenHTML = tens;
```

```
  appendSeconds.innerHTML = seconds;
```

```
}

function startTimer () { tens++;

if(tens <= 9){
appendTens.innerHTML = "0" + tens;
}
if (tens > 9){
appendTens.innerHTML = tens;
}
if (tens > 99) {
console.log("seconds"); seconds++;
appendSeconds.innerHTML = "0" + seconds; tens = 0;
appendTens.innerHTML = "0" + 0;
}
```

Practice/HW:

Modify the above application as below:

1. For the first app try to fetch data from other APIs like weather API, news API
2. Add input fields for taking input for the time and date and the countdown should start from now to the date and time provided in the input
3. Take the input from the user for the time and date from where to start and stop the countdown

Thank You