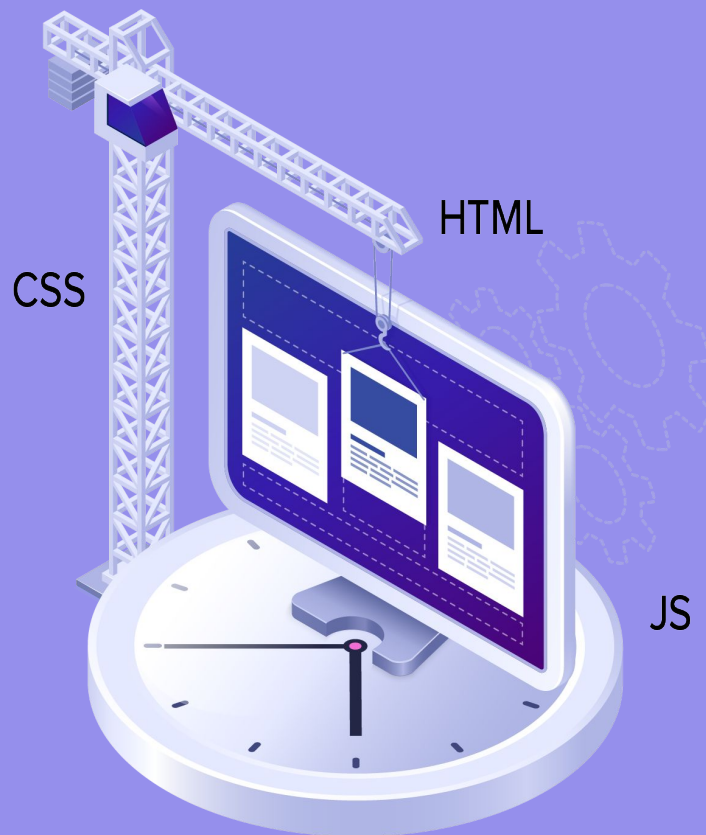


Countdown timer app

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



The countdown will end in

32d 10h 4m 27s

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 a 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(contliracfe) {
3     setTimeout(function() {
4       Let continent = 'Asia'; callback(continent)
5     }, 2000)
6   }
7
8
9 /unction getCountries(continent, callback) {
10   setTimeout(/unction() {
11     Let countries = ["India"* "Pakistan",
12                     "Bangladesh", "Sri Lanka", "Afghanistan"];
13     callback(countries);
14   }, 2000)
15 }
16
17 /unction getStates(country* callback) {
18   setTimeout(function() {
19     Let states = ["UP", "HP"* "AP"* "TK"];
```



```

21     }
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
```

```
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(resolve, 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 a 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

Output:

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

```
setInterval(myTimer, 1000);  
  
function myTimer() { const d = new Date();  
  document.getElementById("demo").innerHTML  
    = d.toLocaleTimeString();  
}
```


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.getElementById("demo").innerHTML  
    = d.toLocaleTimeString();  
}
```

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: antiquewhite;
>
h3,h2 0
text-align: center; font-size: 60px; margin-top: 100px; border: 2px solid black;
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.getElementById("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() {  
  
    clearInterval(Interval);  
    Interval = setInterval(startTimer, 10);  
}  
  
buttonStop.onclick = function() { clearInterval(Interval);  
}  
  
buttonReset.onclick = function() { clearInterval(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