






Topic	COLLABORATIVE DOCUMENT : INTERACTION	
Class Description	Students will be able to understand how to interconnect several users to make collaborative documents with Twilio.	
Class	ADV-C271	
Class Time	60 mins	
Goals 	<ol style="list-style-type: none"> 1. Implemented the concept : setTimeout and event listener of javascript. 2. Learned how to send and receive messages to the twilio server and display them on the console screen. 	
Resources Required	<ul style="list-style-type: none"> • Teacher Resources: <ul style="list-style-type: none"> ○ Use Gmail login credentials. ○ Laptop with camera ○ Earphone with mic ○ Notepad and Pen • Student Resources: <ul style="list-style-type: none"> ○ Use Gmail login credentials. ○ Laptop with camera ○ Earphone with mic (optional) ○ Notepad and Pen 	
Class Structure	Warm Up Teacher-Led Activity Student-Led Activity Wrap Up Project Pointers and Cues	5 Mins 15 Mins 30 Mins 5 Mins 5 Mins
NOTE: Teacher should execute the same code in two different browsers and check the interaction working, before conducting this class.		
Class Steps	Say	Do

		
Step 1: Warm up (5 mins)	<p>In the previous class 270, we created an account on Twilio, added the Account SID, Sync service SID, API key and Secret key in our python code and linked Twilio with our code.</p> <p>Q What Account SID holds?</p> <p>A Account SID holds the address of our Twilio account.</p> <p>Q What Account service SID holds?</p> <p>A Account service SID holds the sync service (collaborative service) ID of your Twilio account.</p> <p>Q Why is the API key used?</p> <p>A API key is used to access Twilio APIs and services.</p> <p>Q What secret key holds for?</p>	

	<p>A Secret key holds the security when we use Twilio features.</p> <p>Also in the previous class we had defined syncClient, syncStream, text_area, select_element, background_color variables in our javascript code.</p> <p>Great effort, understanding about Twilio is not an easy thing but you cracked it at a young age. Keep it up.</p> <p>We will continue our coding for creating a collaborative document, and In this class we are going to publish interacted messages on the console using Javascript code with the same Twilio API key (locally just for testing purpose).</p> <p>In the next class we will complete this collaborative document creation and deploy it on cloud so that everyone can access and use it, just like google docs.</p>	
Teacher Initiates Screen Share		
<p>Step 2: Teacher-Led Activity (15 mins)</p> 		<p>NOTE: Course outcomes are the activities which are giving the idea of concept to be implemented so precisely</p>

		<p>convey it to learners before the Teacher Activity delivery</p> <p>SAY: Students will come to know how multiple functions will be used to publish the data across all the users.</p>
<p>Before we start our class let's understand the concept of how data is transferred between multiple users using the same account IDs and APIs.</p> <p>In the school computer lab: students perform various tasks and work on worksheets, extra activities, assessments, projects in their computers. But this data can be accessed by every student in a class because a common password is shared to login with everyone.</p>  <p>In companies, some credentials are shared commonly with all the employees to access datas and files.</p>		



In homes, offices, we can share and use the internet with the common wifi password.



We can store our photos, audios, videos, files through Google Drive, Microsoft One Drive, Dropbox and we can share to others.



In the above examples we see with the same credentials multiple users can access the data and files and edit it.

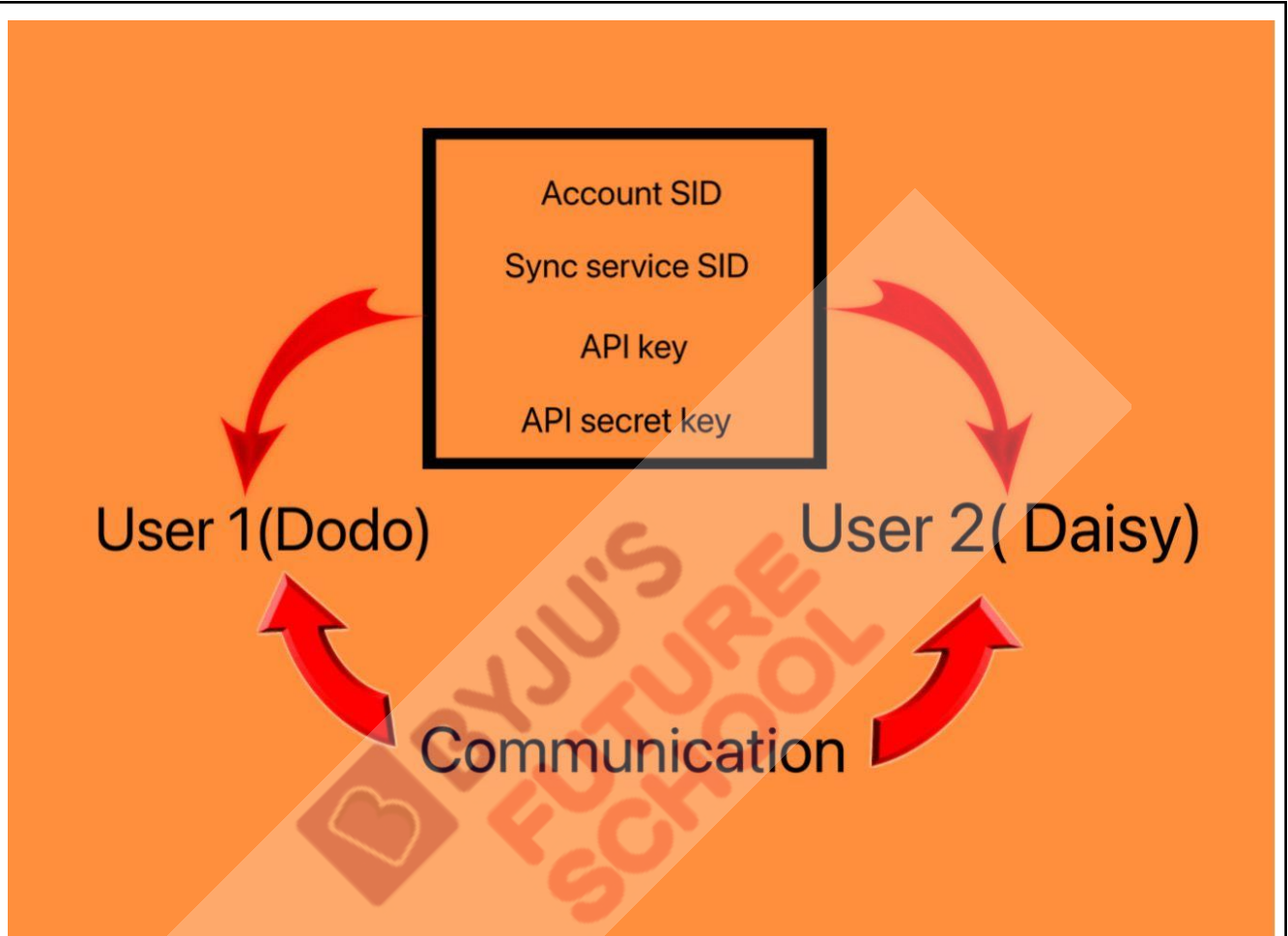
Similarly in our collaborative document using the same Twilio account ID and API key's multiple users can access data and files and edit it.

In the previous class we created our Twilio account and incorporated our ID and API key for secure connection.

So everytime a new user comes on our collaborative documents the ID and API key will remain the same which we had created, only the username will change for uniquely identifying the users.

Let's see this with an example for this:

Imagine there are two users using the same collaborative document with the same keys. User1(Dodo) and User2(Daisy).



When User1 (Dodo) write's anything it should be synced with the document of User2(Daisy), vice versa.

So the basic reason that the ID and API key should be the same is that the content can be transferred between multiple users.

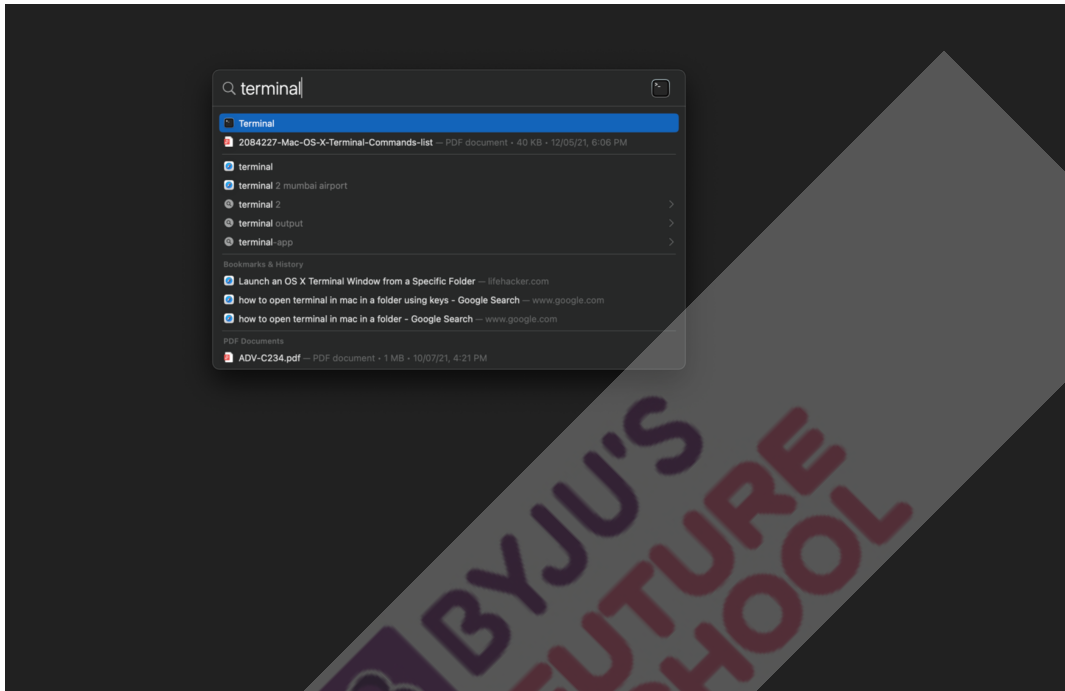
Let's see this on a demo:

Note: Download code from [Teacher-Activity-2](#) and move the folder into Documents/cloud folder.

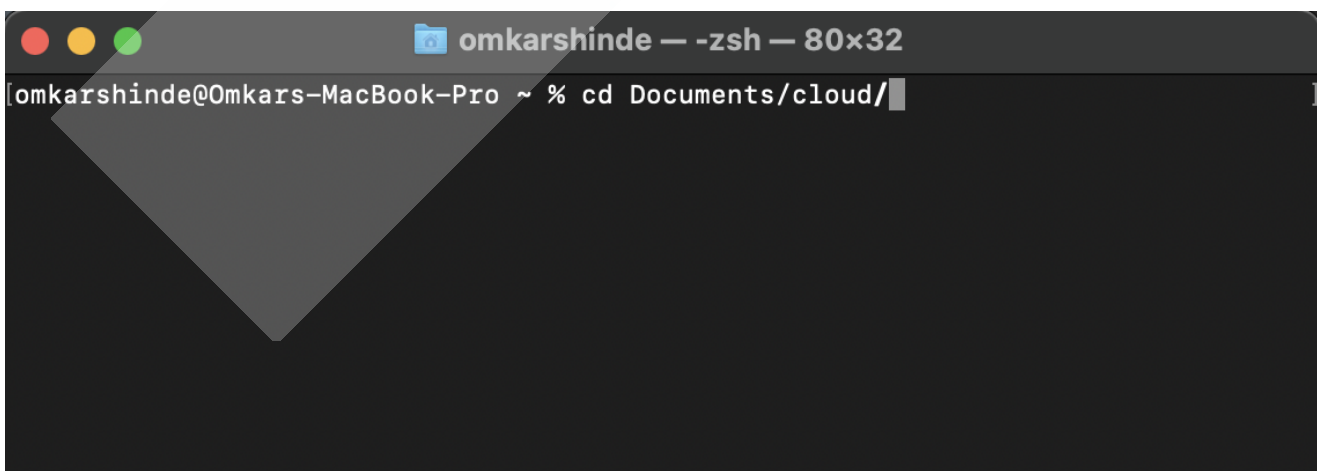
Please add YOUR Twilio Account SID, Sync service SID, API key, API secret key in your 'app.py' file.

Testing the code For Mac:

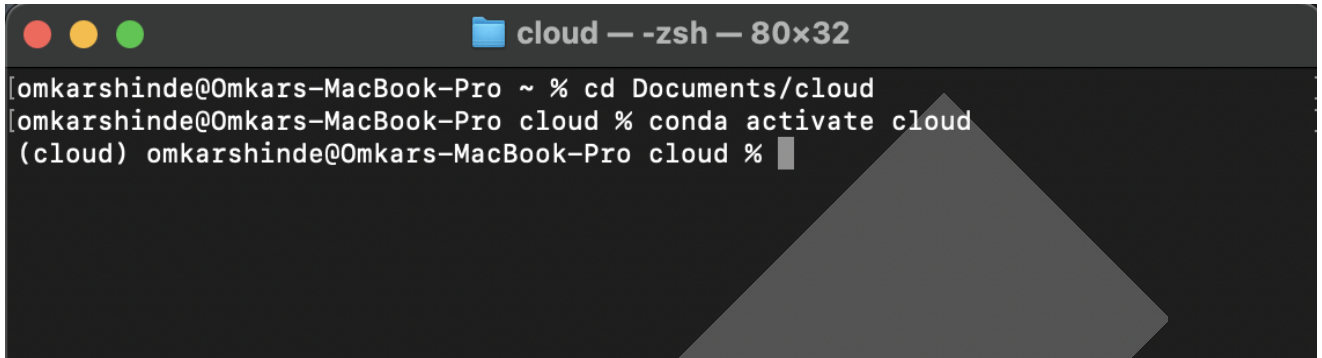
1. Go to “cloud” folder in Documents directory and open the cloud folder
2. Now to run file in terminal first open terminal by pressing command + SPACEBAR key and type ‘terminal’



3. Then locate Documents folder using cd command as
cd Documents/cloud/

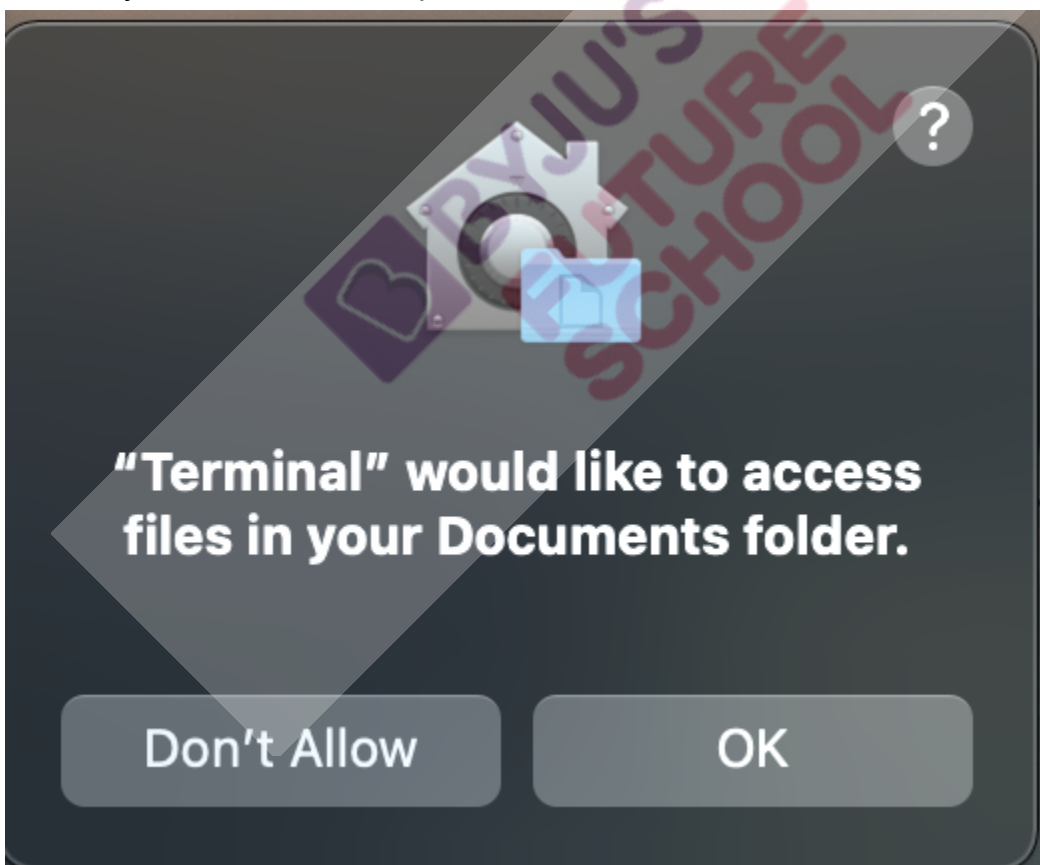


4. Now activate blockchain environment as
conda activate cloud

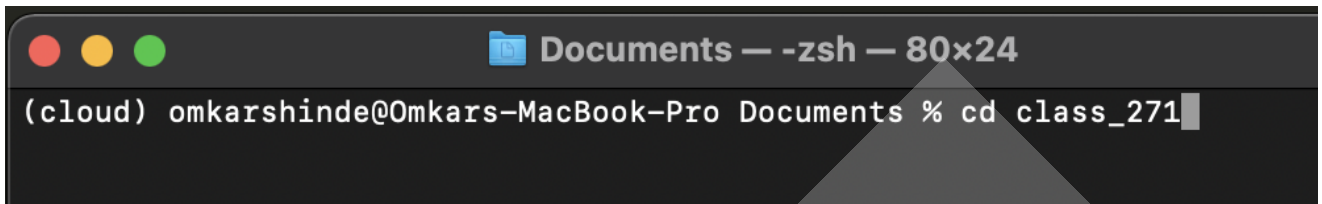


```
cloud — -zsh — 80x32
[omkarshinde@Omkars-MacBook-Pro ~ % cd Documents/cloud
[omkarshinde@Omkars-MacBook-Pro cloud % conda activate cloud
(cloud) omkarshinde@Omkars-MacBook-Pro cloud %
```

5. If you see this window please click on OK



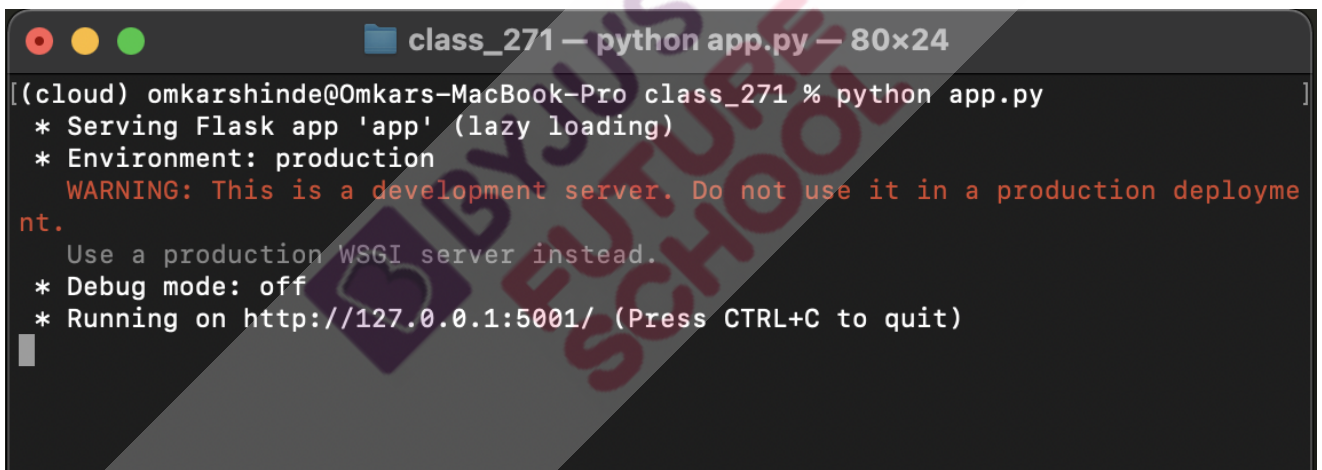
6. And then go to class_271 folder by **cd class_271**



```
Documents — -zsh — 80x24
(cloud) omkarshinde@Omkars-MacBook-Pro Documents % cd class_271
```

7. Now run python file as
python app.py

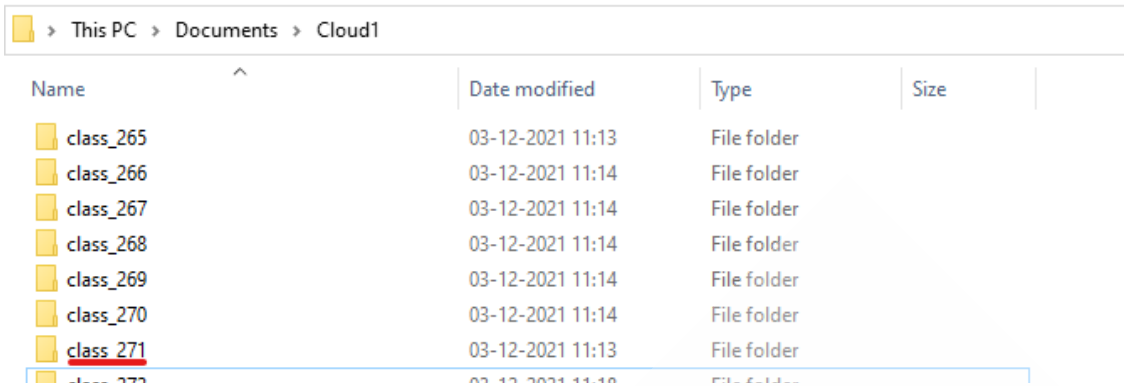
8. It will look like this,



```
class_271 — python app.py — 80x24
[(cloud) omkarshinde@Omkars-MacBook-Pro class_271 % python app.py]
* Serving Flask app 'app' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5001/ (Press CTRL+C to quit)
```

Testing the code For Windows :

1. Now open the **CMD** in **class_271** folder which is inside the **Cloud** folder which is located in **Document/Cloud** directory.



Name	Date modified	Type	Size
class_265	03-12-2021 11:13	File folder	
class_266	03-12-2021 11:14	File folder	
class_267	03-12-2021 11:14	File folder	
class_268	03-12-2021 11:14	File folder	
class_269	03-12-2021 11:14	File folder	
class_270	03-12-2021 11:14	File folder	
class_271	03-12-2021 11:13	File folder	

2. Now active the cloud environment by using the command as **conda activate cloud**

```
C:\Windows\System32\cmd.exe - python app.py
Microsoft Windows [Version 10.0.19042.1348]
(c) Microsoft Corporation. All rights reserved.

C:\Users\DELL\Downloads\class_271\class_271>conda activate cloud
```

3. Now run python file as **python app.py**

```
C:\Windows\System32\cmd.exe - python app.py
Microsoft Windows [Version 10.0.19042.1348]
(c) Microsoft Corporation. All rights reserved.

C:\Users\DELL\Downloads\class_271\class_271>conda activate cloud
(cloud) C:\Users\DELL\Downloads\class_271\class_271>python app.py
```

4. It will look like this,

```
C:\Windows\System32\cmd.exe - python app.py
Microsoft Windows [Version 10.0.19042.1348]
(c) Microsoft Corporation. All rights reserved.

C:\Users\DELL\Downloads\class_271\class_271>conda activate cloud

(cloud) C:\Users\DELL\Downloads\class_271\class_271>python app.py
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5001/ (Press CTRL+C to quit)
```

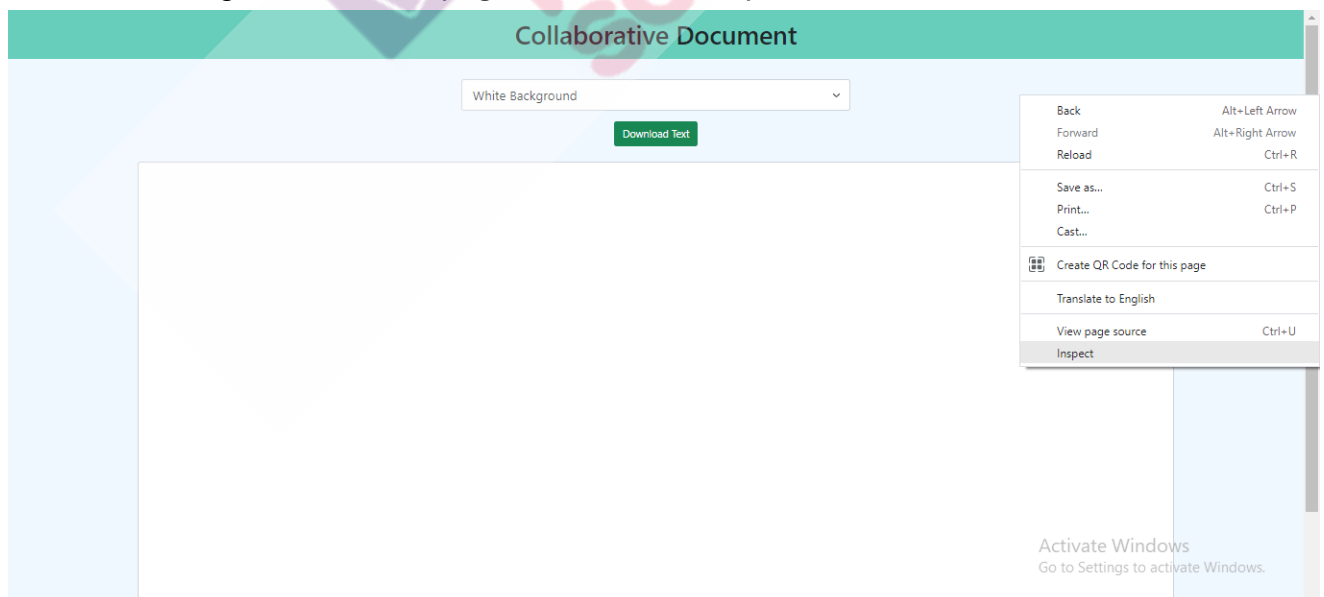
Output Common for both MAC and WINDOWS

Now open two different browsers (Example: Browser 1 - Google Chrome, Browser 2 - Mozilla Chrome / Microsoft Edge / Opera / open Google Chrome in new Incognito window).

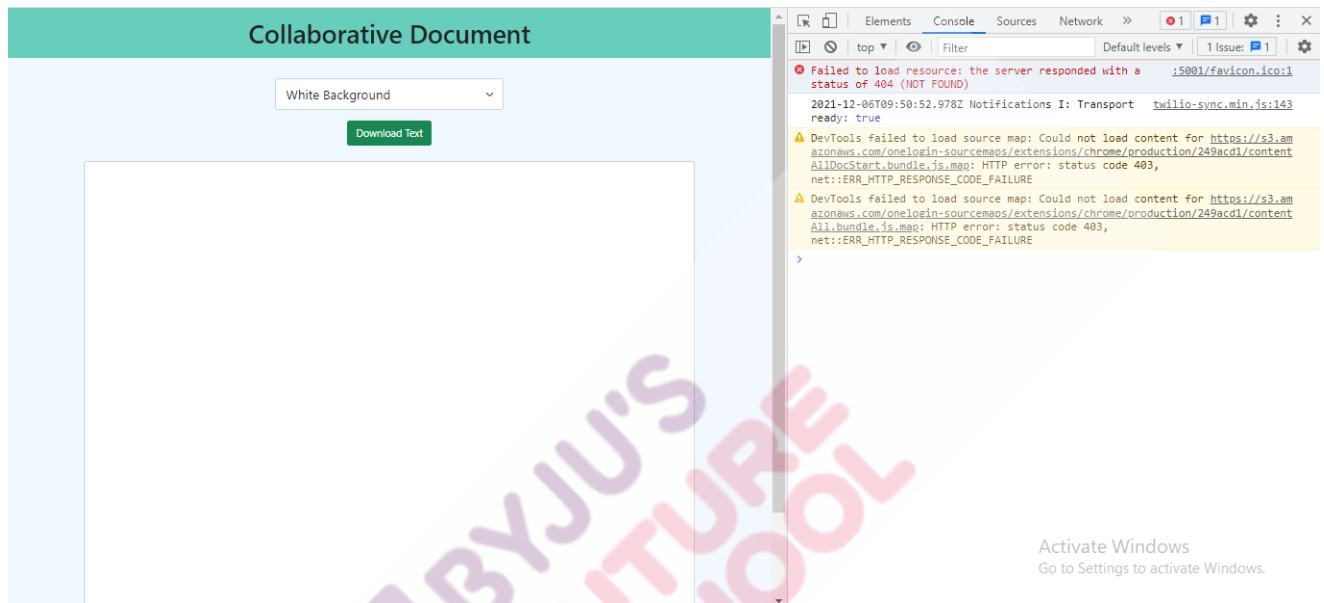
And paste this address : 127.0.0.1:5001

Here 127.0.0.1 is our host and 5001 is our port number to run our flask application.

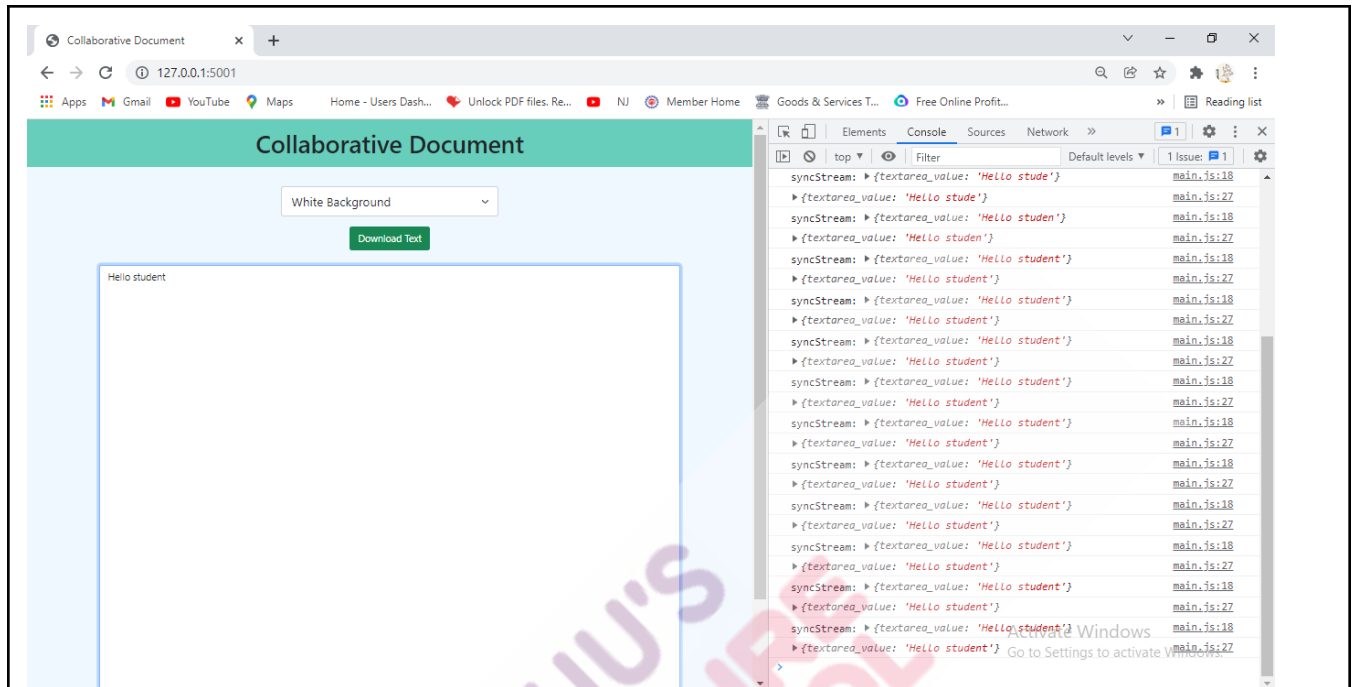
In Browser 1, right click on the page and choose 'Inspect'.



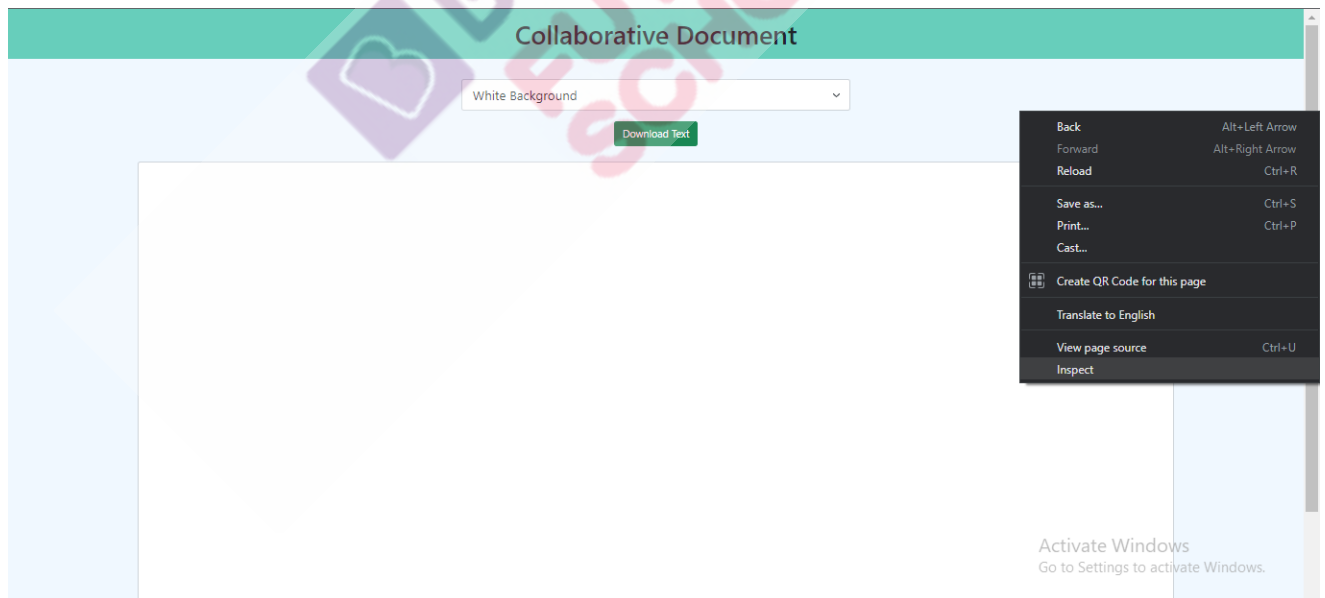
You will get an Inspect tab on the right side of the Browser 1 now click on the 'Console' from the Inspect tab.



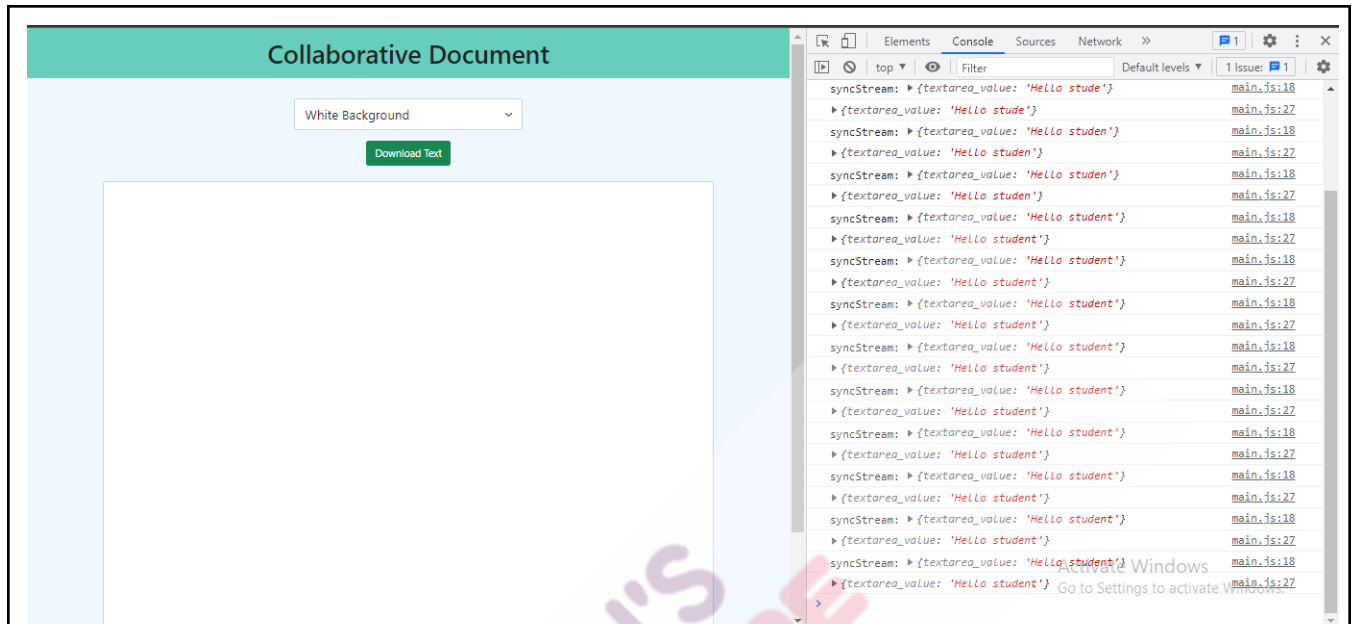
Now type something (Hello student) in the TextArea - Browser 1. The message will get displayed on the console.



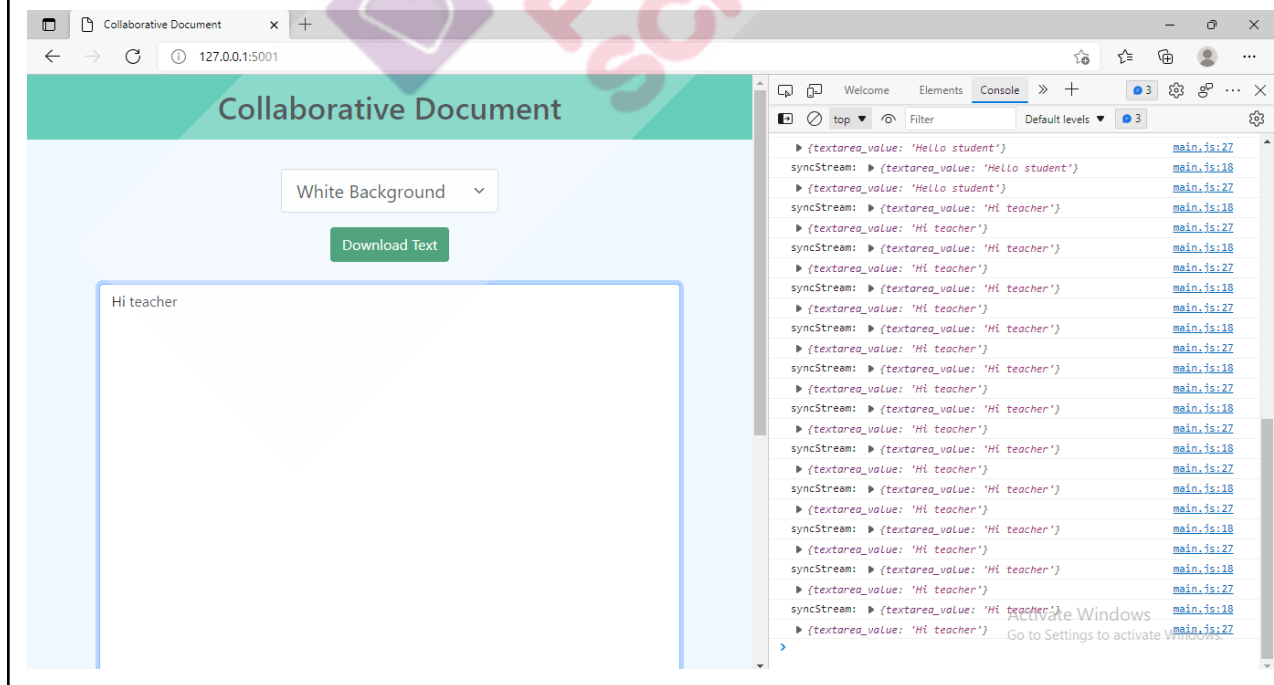
Now in Browser 2, right click on the page and choose 'Inspect'.



You will get an Inspect tab on the right side of the Browser 2 and click on the 'Console' from the Inspect tab. And you can see the message on the Console that you typed in Browser 1.



Now type something (Hi teacher) in the TextArea - Browser 2. And you can see the same message getting displayed in the Browser 1 console screen.



© 2021 - BYJU'S Future School.

Note: This document is the original copyright of BYJU'S Future School.
Please don't share, download or copy this file without permission.

NOTE - Do this activity 2-3 times to show message sync to the student.

Here you can see both these websites are working on different browser tabs, but still they are sync together. The reason is that both collaborative documents have the same Twilio account ID and API key.

In the last class we learnt how to get a Twilio Account SID, Service SID, API key and Security key and also we have added them in our python code.

In today's class we are going to add the code for syncing different users so that anyone can access and edit the shared document which is seen by everyone.

Before we start coding, let's do the `setTimeout()` function of javascript which we will be using in today's class.


setTimeout():


`SetTimeout()` function is used to run a specific line of code after a specific interval of time. Example : imagine if you want to print "Hello world" after every one second.

Let understand a example of **setTimeout** function via code:

NOTE: Open Teacher-Activity-1

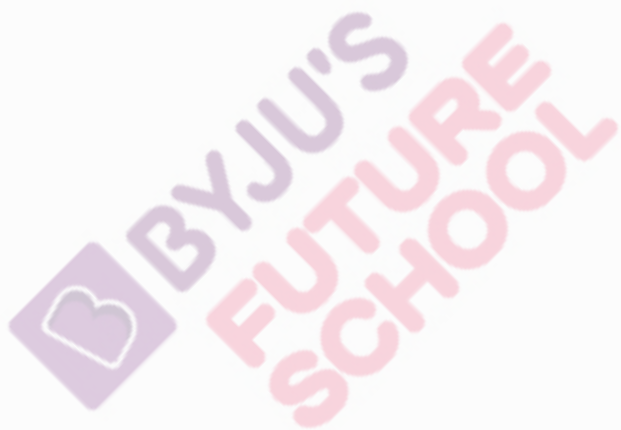
In this activity we will use the `SetTimeout` function to increase the count variable by 1. So when you run the code in repl you will increase the count variable in the web console. Press

the  button and on console the value will keep on increasing by 1

 <https://setTimeout.mahdiwhitehat.repl.co>  

Practice setTimeout

Try it



Console

Shell

```
1  
2  
3  
4
```





```
1  <html>
2  <head>
3  </head>
4  <body>
5  <h1>Practice setTimeout
6  <br><br>
7  <button onclick="myFunction()">Try it</button>
8
9  <script>
10
11  var count = 0
12  function myFunction()
13  {
14      setTimeout(function(){
15          update()
16      },
17      1000);
18  }
19
20  function update()
21  {
22      count = count + 1
23      console.log(count)
24      myFunction();
25  }
26 </script>
27 </body>
28
```

In this activity we are going to understand how the **setTimeout** function works. And we are going to print numbers from 1 and increase the value by 1 after every 1000 millisecond and print the values on the console.

- At line 7, we are initializing myfunction() function on click on button

```
<button onclick="myFunction()">Try it</button>
```

- Here onclick is a event which will execute myFunction() on click of the button "Try it"

```
var count = 0
function myFunction()
{
    setTimeout(function(){
        update()
    },
    1000);
}

function update()
{
    count = count + 1
    console.log(count)
    myFunction();
}
```

- At line 11, we are defining a variable 'count' and assigning a value 0 to it. This variable will be used to store the counter value `var count = 0`

- At line 12, we are defining a function named myfunction().

```
function myFunction()
```

Here, we are creating a function to set the specific interval of time to display the counter value.

- At line 13, we are defining the setTimeout() function for the specific interval of time.

```
setTimeout(function(){
    update()
},
1000);
```

- Here we are using the setTimeout function to set the specific interval of time.

- Inside the setTimeout function we are passing an update() function to get the updated counter value (increase by 1) and 1000 value which will call update() function after every 1000 milliseconds(1 second).
- At line 20, we are defining a function named update()

```
function update()
```

 - Here, we are creating a function to increase the counter value by 1 and printing the value on the console.
- At line 22, we are increasing the count variable by 1.

```
count = count + 1
```

This will result in updating count variable value by 1 every time when the update() function is called.
- At line 23, we are printing the count value.

```
console.log(count)
```

Here, the console.log() function is used to print the values on the console when the button “Try it” is clicked.
- At line 24, we are calling myFunction() function.

```
myFunction();
```

Here, for displaying values for the specified interval of time, we are calling this function.

Code & Explanation

Today we will be performing the below mentioned tasks:

1. Some of the Javascript code is predefined for you, you need to complete the code.

Let's write following Javascript code in main.js file which stored in Documents\cloud\class_271\static\main.js

Javascript code:

```

1  function () {
2      var syncClient;
3      var syncStream;
4      var message = document.getElementById('message');
5      var text_area = document.getElementById('text_area');
6      var select_element = document.getElementById('select');
7      var background_color;
8
9
10     $.getJSON('/token', function(tokenResponse) {
11         syncClient = new Twilio.Sync.Client(tokenResponse.token, { logLevel: 'info' });
12
13         // create the stream object
14         syncClient.stream('messageData').then(function(stream) {
15             syncStream = stream;
16             // listen update and sync drawing data
17             syncStream.on('messagePublished', function(event) {
18                 console.log('syncStream:', event.message.value);
19                 syncDrawingData(event.message.value);
20             });
21         });
22     });
23
24     function syncDrawingData(data) {
25         console.log(data);
26     }
27
28     function messageSync() {
29         {
30             text = document.getElementById("text_area").value;
31             setTimeout(function(){
32                 SettingsSyncData()
33             },
34             1700);
35         }
36     }
37
38     function SettingsSyncData(){
39         syncStream.publishMessage({
40             textarea_value:text
41         });
42     }
43
44     text_area.addEventListener("keyup", messageSync)
45
46
47
48

```

Variables set in previous class

Predefined code

To print data on console

To get text from HTML page

To publish text data to twilio server

To define event listener

Let's do a quick revision of the variables we set in previous class:



What does syncClient variable hold for?



syncClient variable holds for storing synchronized text data which we want to share with twilio server and share with other users.



What does the syncStream variable hold?

A syncStream variable holds for synchronizing the data with the twilio system.

Q What does the message variable hold?

A message variable holds the message which is to be displayed on the webpage.

Q What does the text_area variable hold?

A text_area variable holds the text which is entered on the text area of the webpage.

Q What does the select_element variable hold for?

A select_element variable holds the color names selected from the select tag.

Q What does the background_color hold?

A background_color holds the colors name which needs to be added to the text area

Code to be done in the class:

```
25 // Print data on console
26 function syncDrawingData(data) {
27     console.log(data)
28 }
29
30 // Get text from HTML page
31 function messageSync()
32 {
33     text = document.getElementById("text_area").value;
34     setTimeout(function(){
35         SettingSyncData()
36     },
37     1700);
38 } // Publish text to twilio server
39 function SettingSyncData(){
40     syncStream.publishMessage({
41         textarea_value:text
42     });
43 }
44 // Define event listener
45 text_area.addEventListener("keyup", messageSync)
46
47 });
48
```

syncDrawingData function:

```
26 function syncDrawingData(data) {
27     console.log(data)
28 }
```

- Here at line 26, we are defining syncDrawingData function
function syncDrawingData(data) {
 - This function helps to sync the data between users.
 - This function will be called when it gets data from the Twilio server.
 - So for example, if we are writing "Hello" on an html text box then it will sync with twilio server and then it will replicate on every other document.
 - Here **data** is nothing but the data we want to pass from one document with the rest.
- Here at line 27, we are display the data on the console
console.log(data)
 - Here we are displaying the data variable. Which will display the data on console

- `console.log()` is a method to display data on web console

Define event listener:

- At line 30, we are defining the `addEventListener()` method.

```
text_area.addEventListener("keyup", messageSync)
```

- First we need to mention which HTML element we need to have an event listener, as we want to get the text from textarea, so we will set the event listener on textarea, for this we will mention the variable `text_area` which holds the textarea element.
- Then define `addEventListener` method which takes two parameters:
 - When an event occurs
 - What should be done when an event triggers.
- In the first parameter we will mention the event should trigger on `keyup`: `keyup` event occurs when the user releases a key (on the keyboard).
- In the second parameter we will mention `messageSync`, this will be a function which will get values from the user and share across all the users. This function we will define now.
- So whenever you write anything in the textarea `messageSync()` function is called.

messageSync function:

- Here at line 32, we are defining the `messageSync()` function to keep sending the data across users in a specific interval of time. This function will be called when any one writes anything in the textarea.

```
function messageSync()
```

- Getting the text from text area

```
text = document.getElementById("text_area").value;
```

- Here we are getting value from the textarea of the html tag, for doing this we will use `document.getElementById()` and pass the ID which we have given to the textarea element of HTML which is 'text_area'.
- This will refer to the textarea, and to get the text written inside textarea we need to add the keyword **value** after `document.getElementById()`. And storing in **text** variable

- Here at line 35, we are defining the `setTimeout` function which will call `SettingSyncdata` function after a specific interval of time.

```
setTimeout(function(){  
    SettingSyncData()  
},  
1700);
```

- Here we are using the `setTimeout` function to keep sending the text from `textarea` across all the users after the given interval of time. Inside the `setTimeout` function we are passing a **`SettingSyncData()`** function which will run after every 1700 milliseconds i.e 1.7 second (1000 milliseconds = 1 second).

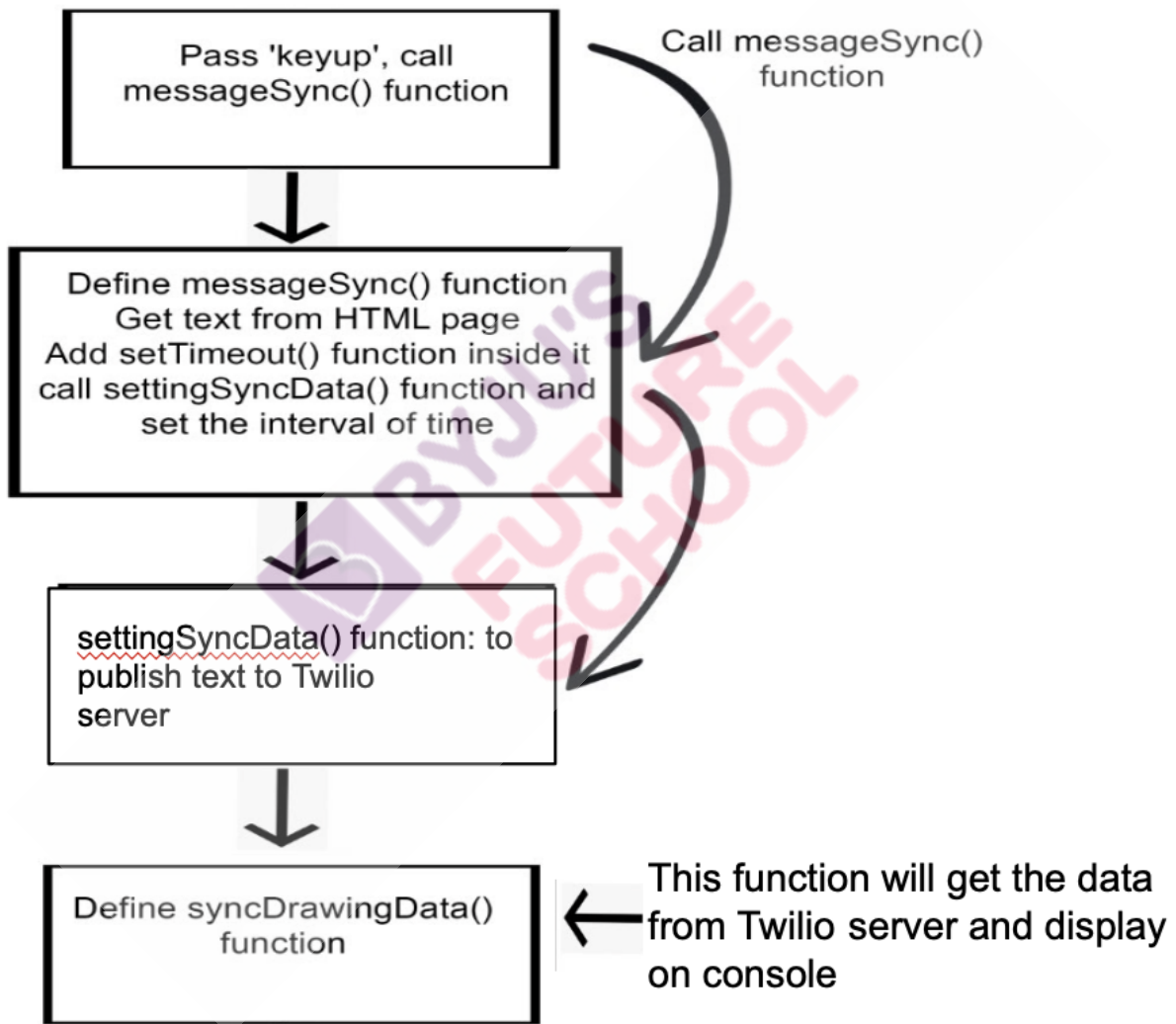
Publish text data to twilio server:

- Here at line 40, we are defining the `SettingSyncData()` function to publish text data on twilio server which will help in sharing the data across all the users. This function is called inside when the `setTimeout()` function, it means this function will be called on a interval of 1.7 seconds

```
function SettingSyncData(){  
    syncStream.publishMessage({  
        textarea_value:text  
    });  
}
```

- In this `SettingSyncData` function we are publishing data which we want to share with all other users.
- Here `syncStream` is a variable to synchronize the Twilio system using the **`publishMessage()`** function to publish data.
- Inside the `publishMessage()` function we are passing **`textarea_value`** as **key**.
- And then we are passing the **value** as **`text`** variable (which holds the texts entered by the user).
- By adding this we can replicate the content from one webpage to the rest of them.

Flowchart:



Teacher Stops Screen Share

Now it is your turn.

- Ask the Student to press the ESC key to come back to the panel.
- Guide the Student to start Screen Share.
- The Teacher gets into Fullscreen.

Step 3:
Student-Led
Activity
(30 mins)



[Student-Activity 1- PRACTICE ACTIVITY](#)

[Student Activity 2 - PREDEFINED CODE](#)

[Student Activity 3 - CODE DIAGRAM](#)

Task 0`:

Open, [Student-Activity-1](#). And complete javascript code.

```
function myFunction()
{
    |   setTimeout(function(){
    |       |   update()
    |       |   },
    |       |   1000);
    |   }
}
```

1. Write SetTimeout code in index.html file

```
function update()  
{  
    count = count + 1  
    console.log(count)  
    myFunction();  
}
```

2. Increment the count variable
3. Then run the file.

Task 02:

Now, open Sublime Text, follow the below steps to create a file and save the file.

1. Download the source code form [Student-Activity-2](#)
2. Unzip the downloaded source code in document\cloud folder
3. Some code is given predefined you need to complete the code

Coding:

1. **NOTE:** Ask the student to put their twilio IDs and APIs app.py file, from their previous class C270 app.py file.

```
@app.route('/token')  
def generate_token():  
    #add your twilio credentials  
    TWILIO_ACCOUNT_SID = ''  
    TWILIO_SYNC_SERVICE_SID = ''  
    TWILIO_API_KEY = ''  
    TWILIO_API_SECRET = ''
```

2. Write JS file

Print data on console

```
// Print data on console  
function syncDrawingData(data) {  
    console.log(data)  
}
```

Get text from HTML page

```
// Get text from HTML page
function messageSync()
{
    text = document.getElementById("text_area").value;
    setTimeout(function(){
        SettingSyncData()
    },
    1700);
}
```

Publish text data to twilio server

```
// Publish text to twilio server
function SettingSyncData(){
    syncStream.publishMessage({
        textarea_value:text
    });
}
```

Define event listener




```
// Define event listener
text_area.addEventListener("keyup", messageSync)
```

Keep the file safe. We will use this code file and add more image processing things in the next class and deploy our application.

Testing the code:

1. Activate created environment
 - **conda activate cloud**
2. Run file
 - **python app.py**

NOTE: Now guide the student to record class output and submit in the student panel. The steps are mention in [Student-Reference-Activity-3](#)

Teacher Guides Student to Stop Screen Share		
<p>Step 4: Wrap-Up (5 mins)</p>	<p>You did great today as well. Great! You have two hats off.</p> <p>Q Which function is used to run a specific line of code on a specific interval of time?</p> <p>A setTimeout() function is used to run a specific line of code for a specific timestamp.</p> <p>Q AddEventListener method for?</p> <p>A The AddEventListener method is to click events to the document. In our code we used the “keyup” event, when we press any keys on the keyboard and releasing the key from the keyboard will do its action.</p>	<p>(Give at least 2 hats offs) Press the Hats Off Icon for Creatively Solving Activities.</p>  <p>Press the Hats Off Icon for Great Question.</p>  <p>Press the Hats Off Icon for “Strong Concentration”.</p>  <p>If you don't have time to perform additional activities, ask the student to perform all the additional activities after the class. Additional activities are VERY important for kids, so that they are ready for the next module. And some challenging</p>

		<p>concepts are coming ahead.</p> <p>Also remind the student to refer to the Student Reference activity for increasing his/her knowledge. This should also be done.</p>
Teacher Initiates Screen Share		
<p>Step 5: Project Pointers and Cues (5 mins)</p>	<p>OTP VERIFICATION - LOGIC II</p> <p>Goals of the Project: In today's class we learnt to publish messages from users to twilio server, so that all the users can access and share the data.</p> <p>OTP VERIFICATION - LOGIC II: In this project, we want you to be creating a GUI based application and define the get_otp() function to get and verify the entered OTP by using twilio and pass the this function's value if OTP verified will redirect to the defined url or else display the error statement.</p> <p>Story: In Last Project, you had created project for HumanRic in which you were generating the otp and redirecting the user to otp</p>	<p>Open the Project Solution link and showcase the output.</p>

	<p>verification page, they liked the way you have submitted the project so they want you to add the next step which is verifying the otp and redirecting the users as per that. If otp is correct then the user will be redirected to the website of HumanRic.</p> <p>Good Luck!</p>	
<p>For the solution of all the Additional Activity, open Teacher-Activity-4 and navigate to class number C271.</p> <p>Additional Activity 1 - Run Student-Activity-4 from the panel. The TASK and HINTS are mentioned on the website itself.</p> <p>Additional Activity 2 - Run Student-Activity-5 from the panel. The TASK and HINTS are mentioned on the website itself.</p> <p>Additional Activity 3 - Run Student-Activity-6 from the panel. The TASK and HINTS are mentioned on the website itself.</p> <p>Additional Activity 4 - Run Student-Activity-7 from the panel. The TASK and HINTS are mentioned on the website itself.</p> <p>Additional Activity 5 - Run Student-Activity-8 from the panel. The TASK and HINTS are mentioned on the website itself.</p>		
Teacher Clicks		<div>✕ End Class</div>

Activity	Activity Name	Links
Teacher Activity 1	SETTIMEOUT ACTIVITY	https://replit.com/@mahdiwhitehat/setTimeOut#index.html
Teacher Activity 2	CODE DIAGRAM	https://docs.google.com/document/d/e/2PACX-1vQ6wrAfTr-vwWiXTFWk3E7aLBN01fOqGnEgtZxon4cilfc4k4Ekd9tb3Cb-rrqN7aRvoVBGjNaDXQsY/pub
Teacher Activity 3	SOURCE CODE	https://drive.google.com/drive/folders/1ucESI2RYwEd1I_Mjw2bpR9hySloOoQMP?usp=sharing
Teacher Activity 4	ADDITIONAL ACTIVITY	https://mahdihat791.github.io/v2/additional_activities_solution.html
Student Activity 1	SETTIMEOUT ACTIVITY	https://replit.com/@mahdiwhitehat/ExemplaryPiercingTrust
Student Activity 2	PREDEFINED CODE	https://drive.google.com/drive/folders/15d2BzxP873W69mMfQLpPOPr1W-Rfk2F3?usp=sharing
Student Activity 3	CODE DIAGRAM	https://docs.google.com/document/d/e/2PACX-1vQ6wrAfTr-vwWiXTFWk3E7aLBN01fOqGnEgtZxon4cilfc4k4Ekd9tb3Cb-rrqN7aRvoVBGjNaDXQsY/pub
Student Reference Activity 1	Timeout function	https://medium.com/@chaoren/how-to-time-out-in-python-726002bf2291
Student Reference Activity 2	Inspect Element console	https://developer.chrome.com/docs/devtools/console/
Student Reference Activity 3	RECORD SCREEN REFERENCE	https://curriculum.whitehatjr.com/ADV+Asset/C145-+Guide+to+loom+recording.pdf
Project solution	OTP VERIFICATION -	https://docs.google.com/document/d/e/2PACX-1vQ6wrAfTr-vwWiXTFWk3E7aLBN01fOqGnEgtZxon4cilfc4k4Ekd9tb3Cb-rrqN7aRvoVBGjNaDXQsY/pub

	LOGIC II	CX-1vQ463K3OZ2NMkfldtpjN5yniNLPZfCajxIC-VSo1Fr8cyUI6nBV-5Sd-eC_YEGqRViO6N1EyGRyNun/pub
--	----------	--

