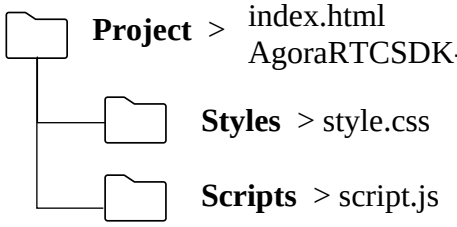
Creating Web Apps
Job Aid

Setup Project Folder Structure
Developers should use their own discretion when creating folder structures. For our development following folder structure needs to be in place:



Index.HTML Page
Developers will have a significant amount of additional code in this page, but the following elements are needed:

<link rel="stylesheet" href="./styles/style.css">
This links the page to a style sheet for formatting control. If this is not present, other style information needs to be on the page.

<script scr="./scripts/script.js"></script>
This code links the page to a different scripts page allowing the developer to update key options.

<script src="./AgoraRTCSDK-3.5.2.js"></script>
Links to the Software Development Kit (SDK) used to run the app. This needs to be after the </body> element. It will appear like this:

```
</body>
<script src="./AgoraRTCSDK-3.5.2.js"></script>
</html>
```

Style.CSS Page
The developer will have many entries on this page than what we have by default. There are some key entries that need to be present or accounted for:

#me video	Container where video plays.
#remote-container video	Container where PiP plays.

Script.JS Page
The generic code for the js page should be included in our customer's js page.

The following code need to be update:

client.init("yourAppID", function()
The default placeholder text of **yourAppID** needs to be replaced with the AppID from the project in the Agora Console.

client.join("yourToken", "myChannel", null, (uid)
The default placeholder text for the following needs to be updated:

- **yourToken**: Copy and replace this text with the Token information from the Agora Console.
- **myChannel**: Update this text with the Channel information from the Agora Console.

```
let client = AgoraRTC.createClient({
    mode: "rtc",
    codec: "vp8",
});
```

Depending on the situation, the option for the Mode and the Codec may need to change:

- Mode:
  - Use **rtc** for communication optimization for one-to-one or one-to-many video and audio calls.
  - Use **live** for broadcast optimization for one-to-many video and audio broadcasts (where one-to-one) communication is not occurring.
- Codec:
  - Use **vp8** for most situations.
  - Use **h264** for Apple Safari Version 12.1 or older.

Script.JS Page Stream Settings

Local Stream

1. Locate the following code:
   client.join("yourToken", "myChannel", null, (uid)

2. Add the following immediate after **{**
   let localStream = AgoraRTC.createStream({
   audio: true,
   video: false, });
   // Initialize the local stream
   localStream.init(()=>{
   // Play the local stream
   localStream.play("me");
   // Publish the local stream
   client.publish(localStream, handleError);
   }, handleError);

3. Save the file.

Remote Stream

By default, the script.js and the SDK are configured for remote streaming. The SDK will look for and triggers the stream-added event. Then, it will attempt to connect to the stream based on the script.js file.

This is based on the AppID, Channel, and Token information updated in the script.js file.

Testing Projects

We recommend that all testing is completed through a local web server. Depending on our customer's configuration, they will need to launch this locally using their IT standards and protocols. Where possible, we recommend using NPM live package server. Refer to Setting Up NPM Server for more information.

Notes:

Running the web app through a local server (localhost) is for testing purposes only. In production, ensure that you use the HTTPS protocol when deploying your project.

Due to security limits on HTTP addresses except 127.0.0.1, Agora Web SDK only supports HTTPS or http://localhost (http://127.0.0.1). Do not deploy your project over HTTP.

1. Open **Terminal.**
2. Type the following command in the terminal to the install live-sever: **npm I live-server -g**
3. Change the directory to your project. Use the cd commands such as **cd c:/project**.
4. Type the following command to run the app: **live-server.**
5. When prompted, allow the browser to access your microphone.
6. Open a new tab and copy/paste the URL from the original tab to the new tab.
7. Test the app. You should hear echo.