Design Document

Creating Web Apps

Bradford Simpson

April 28, 2021

Version 1.0

Executive Summary

Provide an overview of the basic steps and concepts for creating web apps for technical and non-technical team members.

Goals/Objectives

Goal

Demonstrate how easy it is to start building web apps using the basic template and SDK tools and resources available. In addition, gain knowledge on the fundamental settings for web apps and terminology for the WebRTE/RTC environment.

Objectives:

By completing this course, participants will be able to:

- Complete basic setup steps of a web app using the SDK using either Windows or MacOS environments.
- Apply an AppID to a web app.
- Update the Token information for a web app.
- Create a channel and update related channel modes for a web app.
- Leverage Mode options to optimize performance.

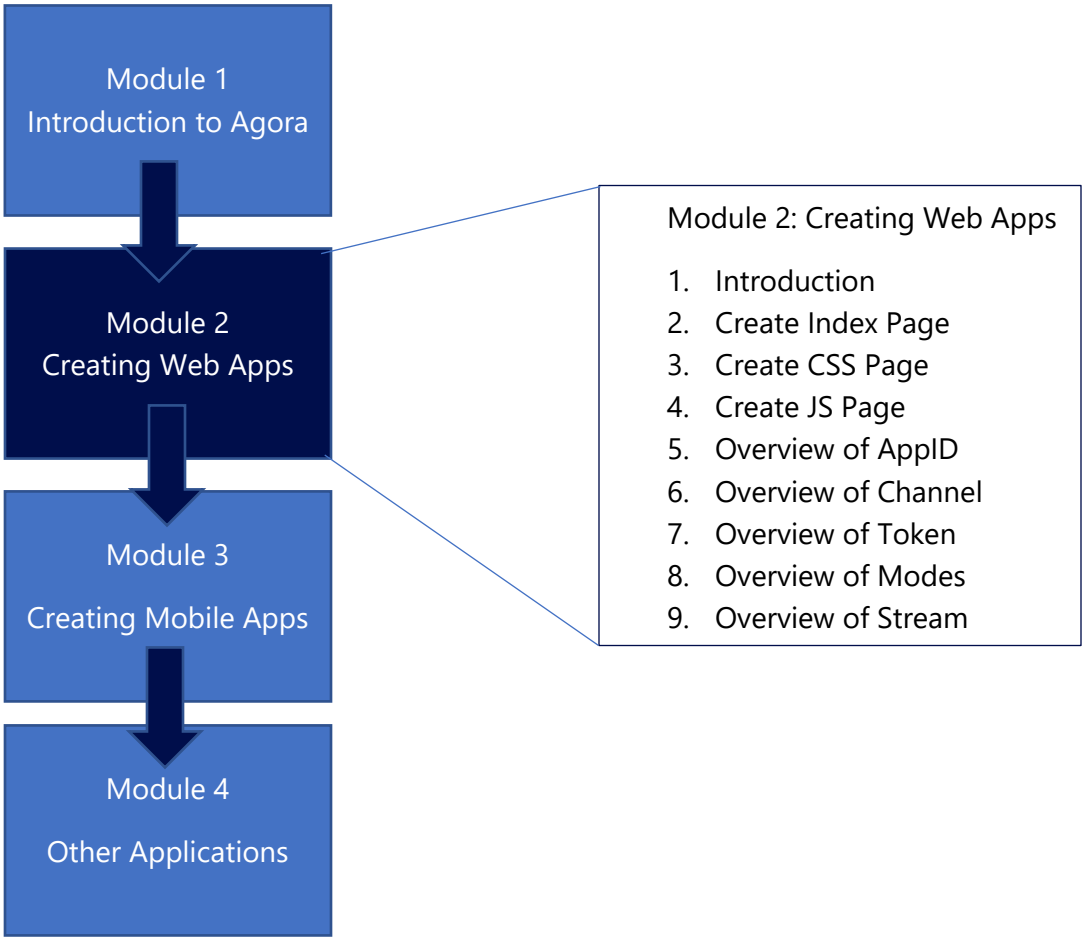
Course Metadata

Course Name	Creating Web Apps
Expected Due Date	April 28, 2021
Tags	Agora, Agora Project, Agora Console, HTM, JS, CSS, AppID, Channel, Token, TokenID, Stream, Host, SDK
Prerequisites	Completion of the Introduction to Agora course. Setup of the Agora Console.
Group Visibility	All LMS Groups

Delivery Modality

- Instructor-Led Training (Virtual or Onsite)
- Online Participant Guide with Self-Service Options hosted on Team Fusion.
- Supporting presentation for concepts

Overall Structure



Agenda

1. Introduction

Review

- Platforms: Video and audio calls are available on:
  - Android
  - iOS
  - MacOS
  - Web
  - Windows

- Framework: These are template coding applications that allow designers to reuse code across development teams. These provide consistency in coding experience and speeds up development time avoiding greenfield code development. Supported frameworks include:
    - Flutter
    - React Native
    - Cocos
    - Creator
    - Cocos2d-x
- Console: The Agora Console provides a cloud-based application and dashboard functionality to manage projects, billing, and usage across the enterprise. Note: For this course, participants need to have created an account and have access to the Console.

Setup Project Folder Structure

To start our development, developers (and participants) need to create a folder structure to use to build out the app. Note: This is the default method based on the SDK and normal website development processes. Individual customers will have different setups. We are introducing the basic concept of what is needed.

- Create project folder.
- Create two sub-folders:
    - Styles: Location where we can reference different style sheets.
    - Scripts: Location where JavaScript and related code files can be referenced.
- Demonstrate
- Hands-on: Ask participants create a project folder.

2. Create the Index Page

Introduction

The index.html file is what browsers access to display webpages for users. Specifically, browsers will automatically look for the index.html page as the start (or beginning) of the website journey. Developers need to create an index.html page. For our web app, will need to create an index.html page as well.

Create an index.html page

- Use and access a text editor.
- Provide code for participants to leverage.
- Copy and paste code from website to text editor.
- Save the file as a html page to project folder.

- Provide opportunity for participants to learn more about the index page created.

3. Create the Style Sheet Page

Introduction

Style.CSS pages (otherwise known as style sheets) control the look and formatting of the webpage when leveraged. Most webpages using HTML use style sheets to quickly make wide sweeping changes to an entire site. Likely, our customers that we will be interacting with will use a style sheet. With this, our participants need to be familiar with style sheets.

Create a style.css page

- Use and access a text editor.
- Provide code for participants to leverage.
- Copy and paste code from website to text editor.
- Save the file as a css/html page to styles sub-folder.

4. Create the JavaScript Configuration Page

Introduction

JavaScript (JS) pages have unique code that used on the site to generate different actions on the web page. We use this to do two things. First, we use this script.js page to set key information used by the generic SDK code. This information is unique to each customer and possibly to each project through the Console. Developers will create these pages, so their (and ultimately our) end users do not have to manually enter these configuration settings each time they are accessing the app or page. We need to be familiar with updating this information and specifically why we are updating this information which will be explored in different sections of this lesson.

Create a script.js page

- Use and access a text editor.
- Provide code for participants to leverage.
- Copy and paste code from website to text editor.
- Save the file as a js page to script sub-folder.

5. AppID Overview and Introduction

Introduction

AppIDs play an important role within the app. They allow us to identify the project and ultimately our customer that using the service. Primary areas that we use AppIDs for are:

- Identify: Using the Console, customers identify themselves using the AppID which also can link to different projects for our customer.
- Grants Permission: With this in mind, we use it to verify access to Agora services as well as check what level of services our customers can access.
- Billing: We use this to then link to the usage and track that for billing purposes.

If an organization develops independently different apps (such as by different teams or departments), the apps should use different IDs. However, if the apps need to communicate with each other, a single AppID should be used.

This is important as only users in the apps with the same AppID can join the same channel and communicate with each other.

Accessing the AppID

- Open the Agora Console.
- Edit the project. If one is not created, create a project.
- Access the project details.

Updating the script.js file

- Open the script.js file.
- Locate the placeholder text.
- Replace it with the copied AppID from the Console.
- Save the updated file.

Channel Overview and Introduction

Introduction

Channels are a way of controlling access to specific sessions within the app. Technically, this allows users to send data over an active connection to a fellow user (peer). It is also used to grant permission that one peer can share information to another peer using the same channel settings.

Channels can be related to television channels. If I am access one channel, I have permissions to view that content. If I switch channels to another station, I may not have access to that content.

Note: Channels and tokens are similar in setup within the script.js. It is easier to update both at the same time other than individually. Also, there is a relationship between channels and tokens.

Token Overview and Introduction

Introduction

Tokens are in many ways a password for the AppID and related Channel options. It is a series of alphanumeric characters that are used to ensure that the user has access to the app and services. It also verifies the level of service. You must have a token to ride different public transportation services or video games. In a sense, this is no different with the tokens that we use to ensure that our customers have access.

If our customers are using a free developer account or demo account, the tokens may be temporary and will require updating until they become a paying customer.

Accessing Channel and Token Information

Accessing the Channel and Token

- Open the Console.
- Open the Project Details.
- Update the Channel text field.
- Create the Token.

Update the Channel and Token Information

- Open the script.js file.
- Locate the *yourToken* placeholder text.
- Copy the Token from the project in the Console.
- Update the placeholder text.
- Update the channel information and then save.

Modes Overview and Introduction

Introduction

Modes allow our customers to optimize the connection type ensure the best experience to their end uses (our end users). There are two channels that we can leverage:

- Live
    - Designed for live broadcasting where one to many with limited to nondirect dialog occurs.

    - Optimized for quality high video and audio
    - There is an assumption that minor freezes on lower network conditions is acceptable.
- RTC
    - Designed for one to one or one to some communications.
    - Optimized for smoothness.
    - Less latency and minimal freezes
    - Assumption that quality may be reduced at times to ensure better dialog.

By default, RTC used. That is completely acceptable for our hands-on project example.

Stream Overview and Introduction

Introduction

Provides the settings of where this is going to be hosted. This can be hosted locally on the workstation or device where it is being developed. It can also be hosted remotely up on the cloud where multiple networked users can access. There are impacts of doing both which we will explore in a later session. For now, consider the approach of impact of latency and access for the stream option.

Codec Overview and Introduction

Introduction

Codec provides the underlining logic of how we encode and decode the video, audio, and related content through the app. It controls the mathematical ways that we compress and fold the information from the broadcaster to the receiver. There is a specific logic that used to then decode that information to play on the receiver's app and device. This provides clear information to the hardware where to play this content (if it be video or audio). We can also use this information to use our backend system to display other information (such augmented reality or gaming applications).

There are two Codec options used by Agora.

- vp8: This is the default standard that most browsers and apps use. It is free for us to use and was originally sourced by Google.
- H.264 (or h264 in the js code) is used by older Apple Safari browsers (version 12.1 and older).

For most of development and support of our customers, vp8 is perfect. There is currently development for vp9 to take better advantage of 5G networks and better optimization for mobile, higher-bandwidth network environments.

Review

Recap the concepts of this lesson showing the hierarchy of the different topics presented:

- Codec
- Stream
- Modes
- Token
- Channel
- AppID
- Project

Analytics

Knowledge Transfer

Create two knowledge checks within the lesson. Recommended locations will be immediately after the AppID or Token. This could be tracked within Team Fusion using normal quizzing functionality. Ensure it addresses the key objectives of the lesson.

Effectiveness

L1: Create a survey at the end of the lesson to capture immediate reaction to the content.

L2: Incorporate this lesson as well as other lessons together into a larger 30, 60, 90-day survey process.

L3: Capture feedback and additional knowledge checks within this lesson as well as other lessons through the course ensuring that there is transfer of knowledge. Also, collect gaps in application of knowledge in performance.

Version Controls

Version	Developer	Date
1.0	Bradford Simpson	April 28, 2021