Storing Images

Now we get to the heart of it. Let's upload images to the cloud! This lesson covers the syntax necessary to upload images to Firebase Storage.

We'll cover the following

Some applications require uploads of videos, images, pdfs, or other types of files from users. The example application we build is centered around images but could have been any type of file, a video-sharing site for instance. Ready to get started?

Full Source Code

This lesson contains the full source code of the application. Please refer to the tabs in the widget below to look at the code and comments.

Here are the main features of the application:

Upload File Syntax

This uploads an image, and when the image is uploaded, it provides you with a download URL, which can be used directly inside an image element as the `src` attributes value.

Preview of Upload

A great way to make sure everything worked correctly is to inject the uploaded image into a `div`. You will see in the code below a `div` with the id `preview-of-upload`. In the JavaScript on line 69, we inject the uploaded image into that `div`. In the next lesson, we will remove this because it covers a real-time listener that

shows all uploaded images by looping through them. This is just a nice in between to make sure all is well and have some visual results.

Optional Progress Bar

The application includes a progress bar to enhance the UI/UX. It's only a few lines of HTML, CSS, and JavaScript.

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <meta http-equiv="X-UA-Compatible" content="ie=edge">
        <title>Image Sharing Application</title>
        <script src="https://www.gstatic.com/firebasejs/6.3.0/firebase-app.js"></script>
        <script src="https://www.gstatic.com/firebasejs/6.3.0/firebase-storage.js"></script>
</head>
<body>
        <div id="container">

                <div id="progress-container">
                        <progress max="100"></progress>
                </div>


                <div class="buttons-grid">
                        <div>
                                <input type="file" id="upload-button">
                        </div>
                </div>

        </div>
</body>
</html>
```

HTML

```
var firebaseConfig = {
        apiKey: "provided apiKey",
        authDomain: "provided authDomain",
        databaseURL: "provided databaseURL",
        projectId: "provided projectId",
        storageBucket: "provided storageBucket",
        messagingSenderId: "provided messagingSenderId",
        appId: "provided appId"
}

// Initialize Firebase
firebase.initializeApp(firebaseConfig);

// Get element that is the input we will click to upload images
const uploadButton = document.querySelector('#upload-button')
```

```
// Get element that shows the progress of the image uploading action
const progressBar = document.querySelector('progress')


// imageFile is global so we can access it after it uploads
let imageFile

// Event listener for if upload image button is clicked and a file has been selected
uploadButton.addEventListener('change', (event) => {

        // Access the chosen file through the event
        let file = event.target.files[0];

        // Define a var just for the name of the file
        let name = event.target.files[0].name;

        // Create a storage reference to the database using the name of the chosen file
        let storageRef = firebase.storage().ref(name)

        // Attach the put method to the storageRef
        storageRef.put(file).on("state_changed",
                snapshot => {
                        let percentage = Number(snapshot.bytesTransferred / snapshot.totalBytes * 
                        progressBar.value = percentage
                },
                error => {
                        console.log('error', error.message)
                },
                () => {

                        // Once upload is complete make a second request to get the download URL
                        storageRef.put(file).snapshot.ref.getDownloadURL()
                                .then((url) => {
                                        // We now have the uploaded url
                                        console.log(url);

                                        // reset the progress bar to zero percent after one second
                                        setTimeout(() => {
                                                progressBar.removeAttribute('value')
                                        }, 1000)
                                })
                })
})
```

JavaScript

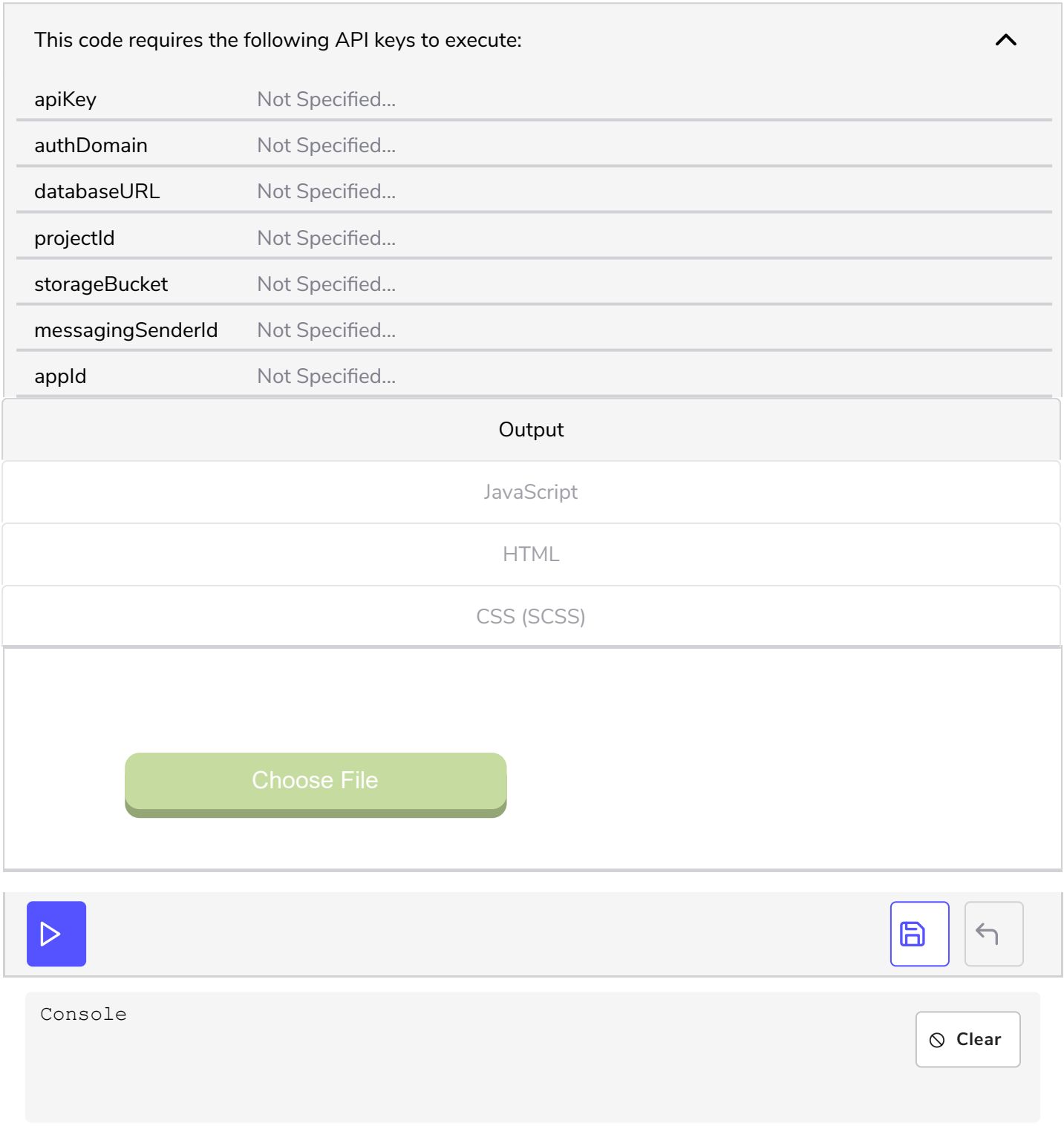
```
#container{
    max-width: 550px;
    margin: auto;
    margin-top: 30px;
}

.buttons-grid{
    display: grid;
    grid-template-columns: 1fr 1fr;
    grid-gap: 30px;
}
```

```css
#progress-container{
   height: 10px;

   margin-bottom: 30px;
}

progress:not([value]) {
    display: none;
 }

progress[value]::-webkit-progress-bar {
    background-color: #EFEFEF;
    border-radius: 20px;
}

progress {
    width: 100%;
    height: 10px;
    background-color: #EFEFEF;
    border-radius: 20px;

}

progress::-webkit-progress-value {
    width: 100%;
    height: 10px;
    border-radius: 20px;
    background-image: -webkit-linear-gradient(left, #b1d3ff, #f1acf8);
}


input[type=file]::-webkit-file-upload-button {
    color: #fff;
        text-align: center;
        border: 0px;
        border-radius: 10px;
        padding: 10px 86px 10px 86px;
        font-size: 16px;
    outline: 0;
    cursor: pointer;
    background-color: #c5db9f;
    box-shadow: 0 6px #94a576;
    margin-bottom: 20px;
}

input[type=file]::-webkit-file-upload-button:hover {
    box-shadow: 0 4px #94a576;
}

input[type=file]::-webkit-file-upload-button:active {
    box-shadow: 0 0 #94a576;
}

input[type=file] {
    margin: auto;
    display: inline;
    width: 260px;
}
```

CSS

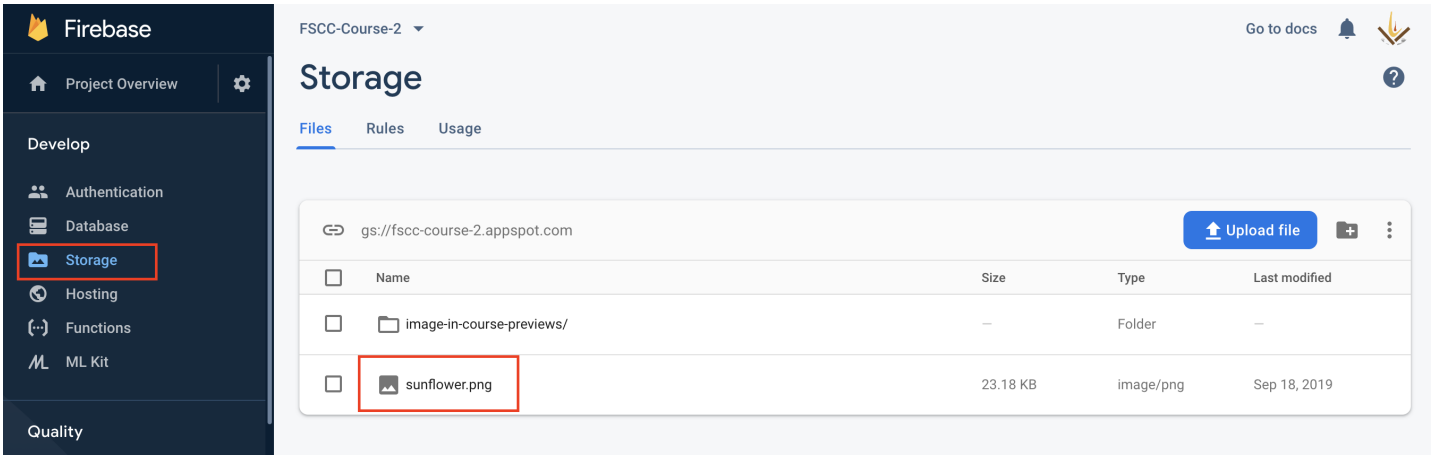
The Image Sharing Application

Finally, let's run the code for the application! Use the upload button to select an image to upload. You should see the progress bar appear, show progress, then disappear when the file has completed uploading.



Check your storage

After uploading an image go to your Firebase console > Storage and verify that the image is there. You can see from the image below I uploaded an image named *sunflower.png*



In the next lesson, I will show you how to take the final download URL of an image that we uploaded to storage and store it in the database so we can access it later dynamically. This gives our app the ability to find and show images to the user.