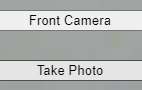
Tutorial 05 – Camera Mobile Website

Preamble

* The Camera Mobile Website application, which is a web application that can be run in browser of mobile device, is created with HTML, CSS, and JavaScript for taking photo with the connected camera.
* After completed this tutorial, you should be able to
  + Implement a web application to access connected camera
  + Familiarize with the use of HTML, CSS and JavaScript
* To test the application, access the web application with browser of desktop computer or mobile device
  + Note:
    - The page has been uploaded to a web server for testing
      * E.g., https://hkmulab.github.io/camera-web-app/

User Interface Design

* The following shows the screen captured from browser of desktop computer
  + 
  + The image located at the center represents the video streaming obtained from the camera
  + The image positioned at the top left corner is the display of the photo taken
  + There are two buttons located at the bottom the page
    - 
    - The upper one is for toggling between front and back camera (is available)
    - The lower one is for photo taken

File Structure

* The project consists of three files
  + index.html
    - Represent the layout of the web page
  + style.css
    - Style sheet language used for describing the presentation style of different elements declared in the .html file
  + app.js
    - The .js file contains Javascript code
    - Include the logic implementation of the web application

Implementation procedures

* Create a new folder on the computer
  + E.g., camera-web-app
* Create and edit the three files (index.html, style.css and app.js) with favorite text editor
  + E.g., Notepad++, Visual Studio Code
* Upload the three files to web server
* Test the application with browser of mobile device

Experiment 1. Edit the HTML file

* Create and edit the “index.html” file as follows

|  |
| --- |
| <html lang="en">  <head>  <meta charset="utf-8">  <meta name="viewport" content="width=device-width, initial-scale=1">  <!-- Name of the app -->  <title>Camera Mobile Website</title>  <!-- Link to main style sheet -->  <link rel="stylesheet" href="style.css">  </head>  <body>  <!-- Main content of the HTML -->  <main id="camera">  <!-- Camera view -->  <!-- canvas: Area for drawing graphics on a web page -->  <canvas id="camera-view"></canvas>    <!-- Camera device -->  <!-- Play the video streaming in "camera-view" -->  <video id="camera-device" autoplay playsinline></video>    <!-- Photo display -->  <img alt="" id="photo-display">  <!-- Front camera button -->  <button id="front-camera-button">Front Camera</button>  <!-- Take photo button -->  <button id="take-photo-button">Take Photo</button>  </main>  <!-- Reference to the JavaScript file -->  <script src="app.js"></script>  </body>  </html> |

* <link rel="stylesheet" href="style.css">
  + Responsible to link to the specific .css file
* <script src="app.js"></script>
  + Refer to an external Javascript file with the “src” attribute in the <script> tag
* The “main” element consists of
  + A “canvas” element (with id=“camera-view”)
    - Display the video stream that obtained from camera
  + A “video” element (with id=”camera-device”)
    - Access the camera
  + An <img> element (with id="photo-display)
    - Display the image taken by the camera
  + A button (with id="front-camera-button")
    - Labeled with “Front Camera” (default) / “Back Camera”
    - Select front or back camera
  + A button (with id="take-photo-button")
    - Labeled with “Take Photo”
    - Snap a picture if clicked

Experiment 2. Edit the CSS file

* Create and edit the “style.css” file as follows

|  |
| --- |
| html, body{  margin: 0px;  padding: 0px;  }  /\*  Style the element with id="camera", "camera-view", "camera-device", "photo-display"  \*/  #camera, #camera-view, #camera-device, #photo-display{  position: fixed; /\* Position of element is relative to the viewport \*/  object-fit: contain; /\* The content is scaled to maintain its aspect ratio while fitting within the element's content box \*/  height: 100%;  width: 100%;  }  /\*  Style the element with id="take-photo-button"  \*/  #take-photo-button{  position: fixed;  bottom: 30px;  width: 100%;  }  /\*  Style the element with id="take-photo-button"  \*/  #front-camera-button{  position: fixed;  bottom: 80px;  width: 100%;  }  /\*  Class "photo-taken", which is added to "cameraOutput" in file "app.js"  \*/  .photo-taken{  height: 100px!important; /\* !important => Overrides the predefined height property (e.g., 100%) \*/  width: 100px!important; /\* !important => Overrides the predefined width property (e.g., 100%) \*/  } |

* Different styles are added to the page with this file to ensure the UI elements are arranged properly
* An element selector for html, body is added
  + The element selector selects HTML elements based on the element name
  + E.g., html, body { ... }
    - Margin and padding of the html and body element in the html page is set to “0”
* Three id selector are declared afterwards
  + The id selector selects a unique element (as the id of an element is unique within a page) with the “id” attribute of an HTML element
  + To select an element with a specific id
    - Write a hash (#) character
    - Followed by the id of the element
  + E.g.,
    - #camera, #camera-view, #camera-device, #photo-display{ ... }
      * Specify the position, object-fit, height and width attributes
    - #take-photo-button{ ... }
      * Specify the position, bottom and width attributes
    - #front-camera-button{ ... }
      * Specify the position, bottom and width attributes
* A class selector is declared lastly
  + The class selector selects HTML elements with a specific class attribute
  + To select element(s) with a specific class
    - Write a period (.) character
    - Followed by the class name
  + E.g., .photo-taken{ ... }
    - Specify the height and width attributes
* Note:
  + Selector is used to "find" (or select) the HTML elements that you want to style
  + “!important” is used to override the predefined property

Experiment 3. Edit the Javascript file

* Create and edit the “app.js” file as follows

|  |
| --- |
| var frontCamera = false;  var currentStream;  // Define constants  // Get the element in the document with id="camera-view", "camera-device", "photo-display", "take-photo-button" and "front-camera-button"  const  cameraView = document.querySelector("#camera-view"),  cameraDevice = document.querySelector("#camera-device"),  photoDisplay = document.querySelector("#photo-display"),  takePhotoButton = document.querySelector("#take-photo-button");  frontCameraButton = document.querySelector("#front-camera-button");  // Access the device camera and stream to cameraDevice  function cameraStart() {  // Stop the video streaming before access the media device  if (typeof currentStream !== 'undefined') {  currentStream.getTracks().forEach(track => {  track.stop();  });  }  // Set constraints for the video stream  // If frontCamera is true, use front camera  // Otherwise, user back camera  // "user" => Front camera  // "environment" => Back camera  var constraints = { video: { facingMode: (frontCamera? "user" : "environment") }, audio: false };    // Access the media device, camera in this example  navigator.mediaDevices  .getUserMedia(constraints)  .then(function(stream) {  currentStream = stream;  cameraDevice.srcObject = stream;  })  .catch(function(error) {  console.error("Error happened.", error);  });  }  // If takePhotoButton clicked => Take and display a photo  takePhotoButton.onclick = function() {  cameraView.width = cameraDevice.videoWidth;  cameraView.height = cameraDevice.videoHeight;  cameraView.getContext("2d").drawImage(cameraDevice, 0, 0);  photoDisplay.src = cameraView.toDataURL("image/webp");  photoDisplay.classList.add("photo-taken");  };  // If Front/Back camera is click => Change to front/back camera accordingly  frontCameraButton.onclick = function() {  // Toggle the frontCamera variable  frontCamera = !frontCamera;  // Setup the button text  if (frontCamera) {  frontCameraButton.textContent = "Back Camera";  }  else {  frontCameraButton.textContent = "Front Camera";  }  // Start the video streaming  cameraStart();  };  // Start the camera and video streaming when the window loads  // 1st parameter: Event type  // 2nd parameter: Function to be called when the event occurs  window.addEventListener("load", cameraStart); |

* Variable “frontCamera”
  + Variable represents the user selects front or back camera
    - True => Front camera
    - False => Back camera
* Constants “cameraView”, “cameraDevice”, “photoDisplay”, “takePhotoButton” and “frontCameraButton”
  + Represent the objects corresponding to the elements declared in the html file
* cameraStart()
  + Function that accesses the connected camera and starts video streaming
  + The “getUserMedia” method is used to access the camera together with the constraints defined
  + Set the “cameraDevice” as the source for the stream
  + To report any error happened during the application execution, a “.catch” is added
* takePhotoButton.onclick = function() { ... }
  + Event handler when the “Take Photo” button is clicked
  + i.e., grab a frame from video stream that can be used as the image output
* frontCameraButton.onclick = function() { ... }
  + Event handler when the “Front Camera” button is clicked
* window.addEventListener("load", cameraStart);
  + Specify the method “cameraStart” is invoked when the window is loaded

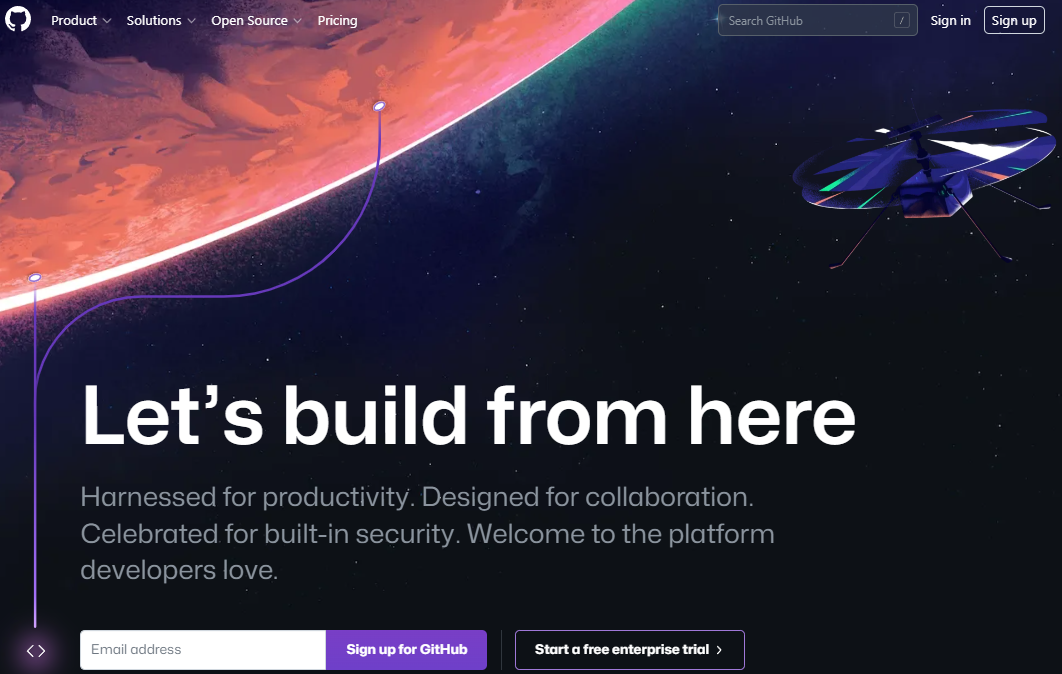
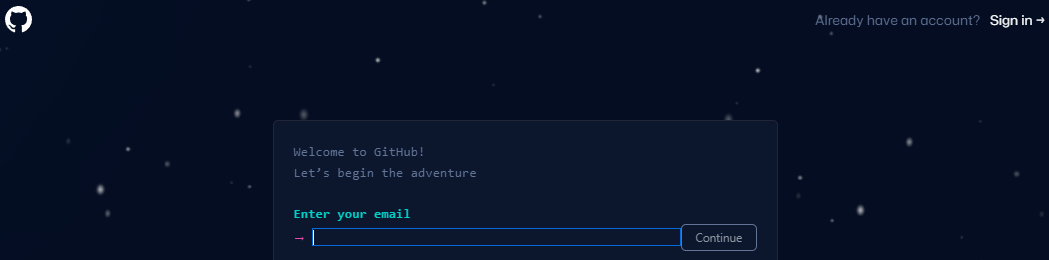
Upload the files to server

* Host the application on a secure HTTPS server
* 1. Locally on your machine
  + Not include in this tutorial
* 2. Provider of Internet hosting for software development and version control
  + E.g., GitHub

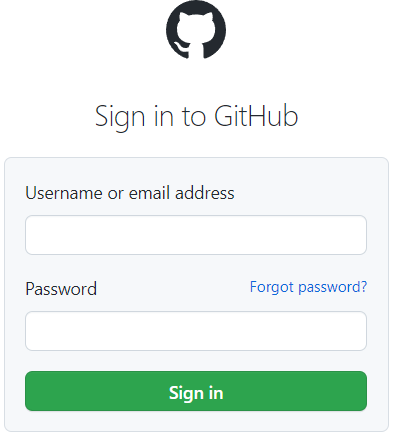
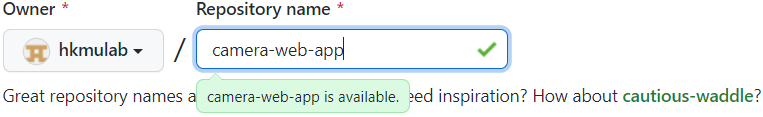
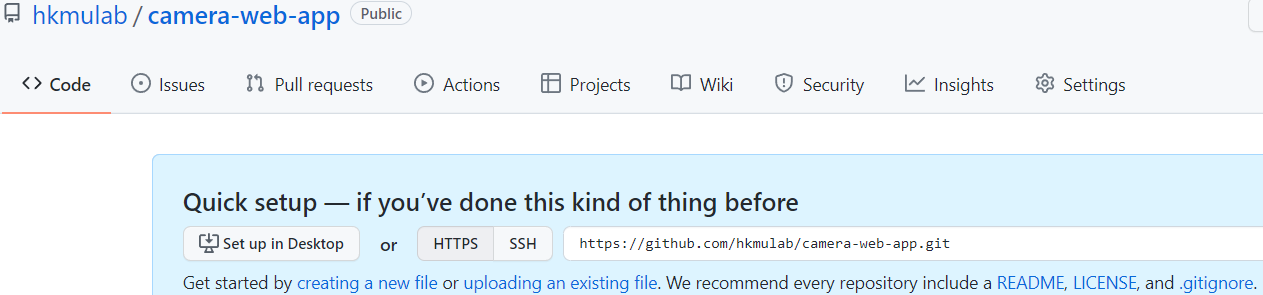
Using GitHub

* 1. Account registration
* 2. Create repository
* 3. Upload files
* 4. Setup GitHub pages

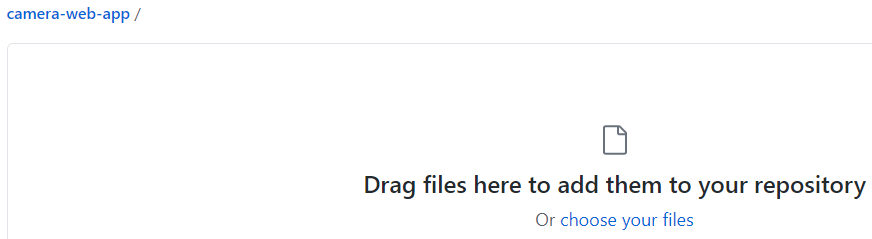
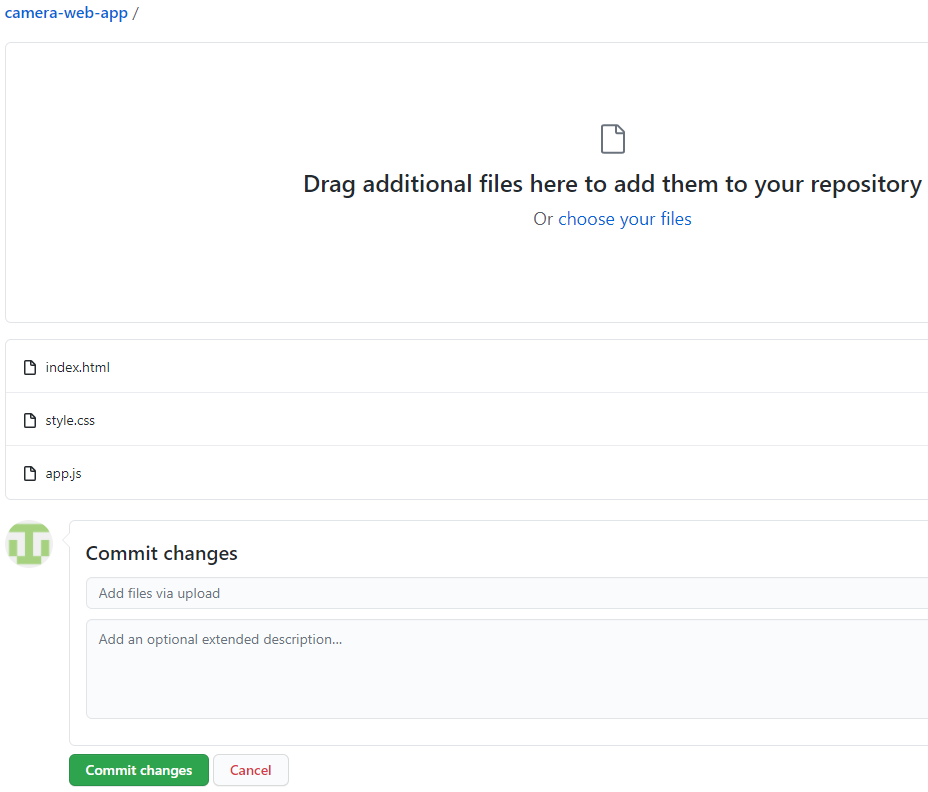
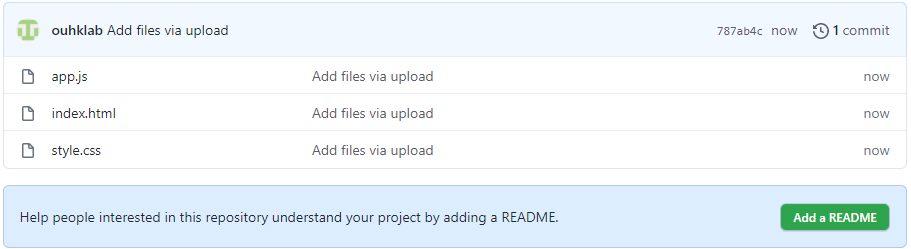
Experiment 1. Account registration

* Access the following URL:
  + https://github.com/
  + 
* Click “Sign Up” button
  + 
* Fill in the information

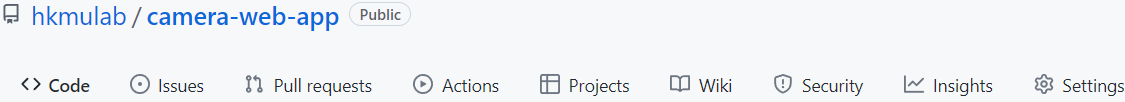
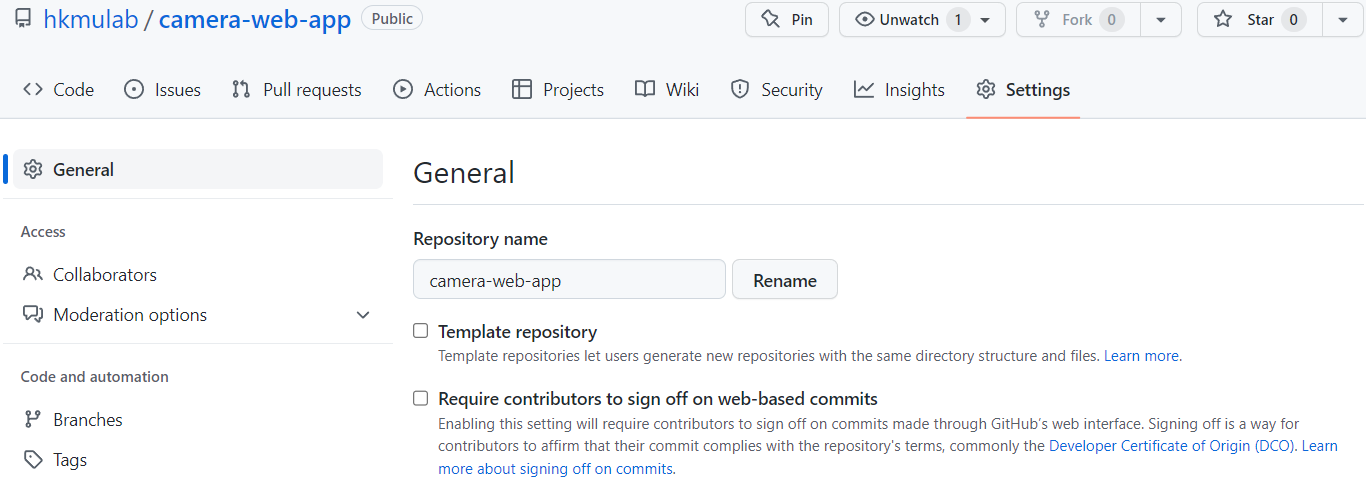
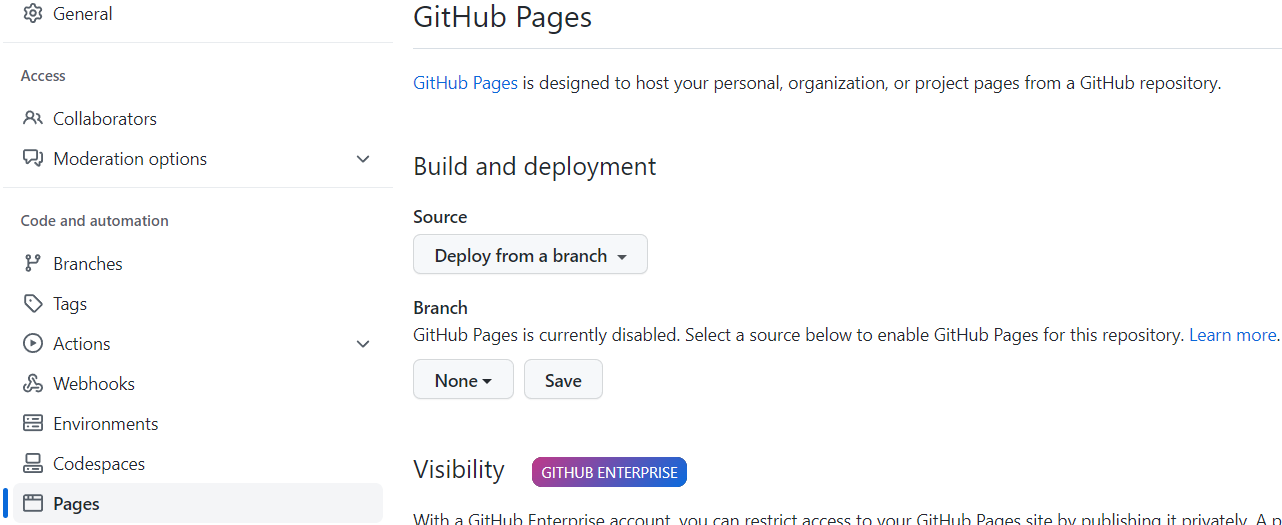
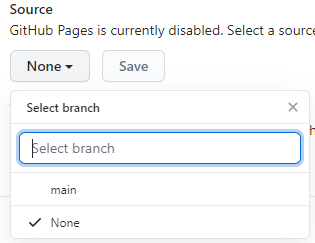
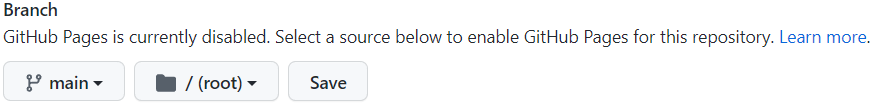
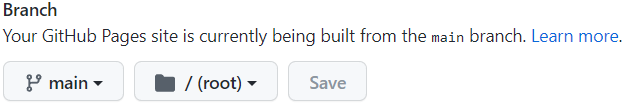
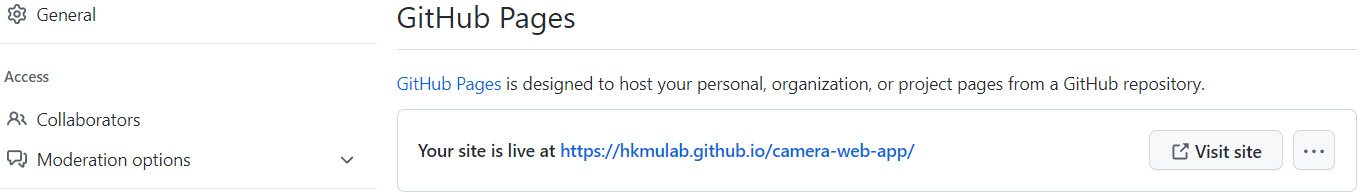
Experiment 2. Create repository

* Login to GitHub
  + 
* Click “Create repository” button
  + 
* Enter “Repository name”
  + E.g., camera-web-app
  + 
* Click “Create repository” button
  + 
* 

Experiment 3. Upload files

* Select “uploading an existing file”
  + 
* Drag or choose the three files to add them to the repository
  + 
* Click “Commit changes” button
  + 

Experiment 4. Setup GitHub pages

* Click the “Settings” tab
  + 
  + 
* Click “Pages” from the panel
  + 
* Click “None”
  + 
* Selection option “main”
  + 
* Click “Save”
  + 
* Refresh the page
  + 
* You application has already been published with the provide URL

Test the application on mobile device

* User browser of mobile device to access the provide URL for testing the web application
  + E.g.,
    - https://<GitHubUsername>.github.io/camera-web-app

More on CSS File

* If the front camera is used, it is more natural to flip the image horizontally, it can be implemented by add the following template to the “style.css” file

|  |
| --- |
| /\*  Style the element with id="camera-view", "camera-device", "photo-display"  \*/  #camera-view, #camera-device, #photo-display{  transform: scaleX(-1);  } |

* To style the “Take Photo” and “Front Camera” button, include the following template to the id selection “#take-photo-button” and “#front-camera-button”

|  |
| --- |
| width: 300px;  background-color: black;  color: white;  font-size: 20px;  border-radius: 30px;  border: 3px solid red;  padding: 15px 20px;  text-align: center;  left: calc(50% - 150px); |

* + Note:
    - Delete the property “width: 100%;”
* To style the display of the snapshot, include the following template with the class selection “.photo-taken”

|  |
| --- |
| top: 20px;  right: 20px; |