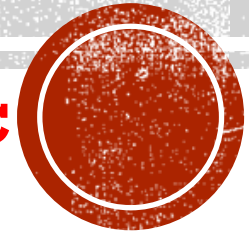
HTML MODULE 08

CREATING INTERACTIVE PAGES BY USING HTML5 APIS



Summer 2021 – Web Development using ASP .Net Core MVC

HTML AUDIO

- Use the **<audio> element** to add an **audio file** to a web page.
  - Add text between the **<audio>** & **</audio>** tags. It will only get displayed if the browser does not support the <audio> element.

- Useful **attributes**:
  - Use the **src attribute** to specify the URL of the audio file (one can also use <source> element to specify alternative sources …)
  - Use the **controls attribute** adds audio controls (such as play, pause, volume).
  - Use the **autoplay attribute** if you want the audio file to start playing automatically
  - Use the **muted attribute** (after **autoplay**) if you want the audio file to start playing automatically, but muted.
  - Use the **loop attribute** if you want the audio file to play repeatedly …

- Example:

```
<!DOCTYPE html>
<html>
<body>

<audio  src="horse.mp3" controls autoplay muted >
    This browser does not support our audio file.
</audio>

</body>
</html>
```

- See also: https://www.w3schools.com/html/tryit.asp?filename=tryhtml5_audio_autoplay
- Source: https://www.w3schools.com/html/html5_audio.asp

HTML VIDEO

- Use the **<video> element** to add a **video** to a web page.
  - Add text between the **<video>** & **</ video>** tags. It will only get displayed if the browser does not support the < **video**> element.

- Useful **attributes**:
  - Use the **src attribute** to specify the URL of the video file (one can also use <source> element to specify alternative sources …)
  - Use the **controls attribute** adds video controls (such as play, pause, volume).
  - Use the **autoplay attribute** if you want the video file to start playing automatically
  - Use the **muted attribute** (after **autoplay**) if you want the video file to start playing automatically, but muted.
  - One can also use the **width**, **height**, and **loop attributes**.

- Example:

```
<video width="600" src="movie.mp4" autoplay muted controls>
  This browser does not support the video tag.
</video>
```

- See also: https://www.w3schools.com/html/tryit.asp?filename=tryhtml5_video_autoplay_mute
- Source: https://www.w3schools.com/html/html5_video.asp

HTML VIDEO

- We can use JavaScript to create the video element shown earlier.


```
<video width="600" src="movie.mp4" autoplay muted controls>
  This browser does not support the video tag.
</video>
```

- Example:

```
<p id="demo"> </p>

<script>
    let myVideo1 = document.createElement("video");
    myVideo1.width="600";
    myVideo1.src = "movie.mp4";
    myVideo1.autoplay = true;
    myVideo1.muted = true;
    myVideo1.controls = true;

    document.getElementById("demo").appendChild(myVideo1);

</script>
```

- See also: https://www.w3schools.com/html/tryit.asp?filename=tryhtml5_video_autoplay_mute
- Source: https://www.w3schools.com/html/html5_video.asp

DOCUMENT.VISIBILITYSTATE

```
<!DOCTYPE html>
<html>
<body>

<p id="demo"> </p>


<script>
    document.addEventListener("visibilitychange", function() {
      console.log( document.visibilityState );
      if (document.visibilityState === 'visible') {
        myVideo1.play();
      } else {
        myVideo1.pause();
      }
    });


    let myVideo1 = document.createElement("video");
    myVideo1.width="600";
    myVideo1.src = "movie.mp4";
    myVideo1.autoplay = true;
    myVideo1.muted = true;
    myVideo1.controls = true;

    document.getElementById("demo").appendChild(myVideo1);

</script>

</body>
</html>
```

- Possible values:
- visible: the page content it at least partially visible.
- hidden: the page content is not visible to the user.
- prerender: "this was removed from the standard."

- Source: https://developer.mozilla.org/en-US/docs/Web/API/Document/visibilitychange_event

HTML YOUTUBE VIDEOS

- To play YouTube videos
  - Use an **<iframe> element** in your web page
  - Set the **src attribute** to point to the video URL you want played
    - Use the format: https://www.youtube.com/embed/uniqueID?optionalParams
  - Set the **width** and **height** attributes to the desired dimensions of the player

- Several other parameters can be added to the URL as shown below:
  - Use **controls=0** if you want to exclude video controls
  - Use **autoplay=1** if you want to video to start playing automatically
  - Use **mute=1** if you want to video to be muted

- As an example, take the following YouTube video: https://www.youtube.com/watch?v=ciNHn38EyRc
  - Then we can use code similar to:

```
<iframe width="600" height="200" src="https://www.youtube.com/embed/ciNHn38EyRc?autoplay=1&mute=1">
</iframe>
```

- Source: https://www.w3schools.com/html/html_youtube.asp

HTML GEOLOCATION API

- used to locate a user's geographical position.
  - For privacy reasons, the position is only available if the user approves it.

- Use the following method:
  - **navigator.geolocation.getCurrentPosition**(***yourOKCallBack***,*yourErrorCallBack*);
  - If the above call is successful, ***yourOKCallBack*** function will receive a coordinates object
  - If the above call is unsuccessful, ***yourErrorCallBack*** function will receive an error object

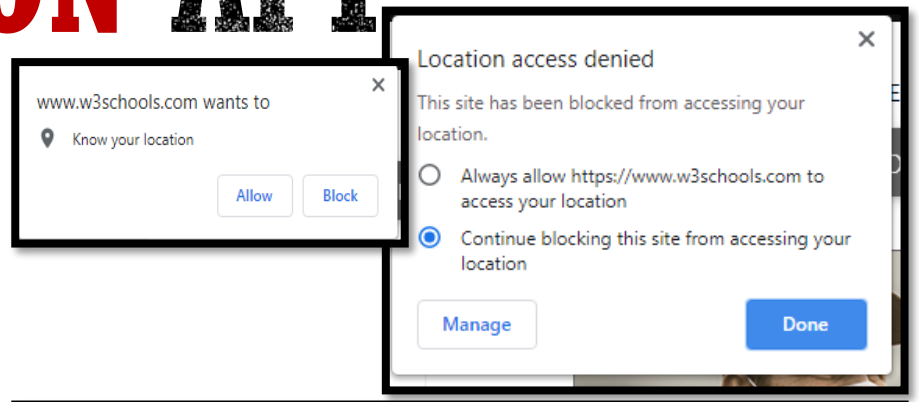
- Example:
  - https://www.w3schools.com/html/tryit.asp?filename=tryhtml5_geolocation_error

- The coordinates object includes the following **properties**:
  - coords.latitude ← the **latitude** as a decimal number
  - coords.longitude ← the **longitude** as a decimal number
  - coords.accuracy ← the **accuracy** of position
  - coords.altitude ← the **altitude** (in meters) - if available
  - coords.heading ← the **heading** as degrees clockwise from North - if available
  - coords.speed ← the **speed** in meters per second - if available
  - timestamp ← the **date/time** of the response - if available

- The coordinates object includes the following **methods**:
  - watchPosition() ← continuously returns updated position as the user moves around (similar to the GPS).
  - clearWatch() ← stops the watchPosition() method.

- The error object includes the following **properties**:
  - code ← such as: PERMISSION_DENIED, POSITION_UNAVAILABLE, TIMEOUT, …
  - message ← a human-readable message containing details of the error

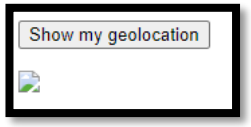
- Source: https://www.w3schools.com/html/html5_geolocation.asp and see also this.




```
<input type="button" onclick="getLocation()" value="Show my geolocation">
<p id="demo"></p>

<script>
var x = document.getElementById("demo");

function getLocation() {
  if (navigator.geolocation) { //if geolocation is supported
    navigator.geolocation.getCurrentPosition(yourOKCallBack, yourErrorCallBack);
  }
  else {  //if geolocation is not supported
    x.innerHTML = "Geolocation is not supported by this browser.";
  }
}

function yourOKCallBack(position) {
  x.innerHTML = "Latitude: " + position.coords.latitude +
  "<br>Longitude: " + position.coords.longitude;
}

function yourErrorCallBack(error) {
  switch(error.code) {
    case error.PERMISSION_DENIED:
      x.innerHTML = "User denied the request for Geolocation."
      break;
    case error.POSITION_UNAVAILABLE:
      x.innerHTML = "Location information is unavailable."
      break;
    case error.TIMEOUT:
      x.innerHTML = "The request to get user location timed out."
      break;
    case error.UNKNOWN_ERROR:
      x.innerHTML = "An unknown error occurred."
      break;
  }
}
</script>
```

HTML GEOLOCATION API

- To display the geolocation on a map, one can use Google Maps api.




```
Failed to load resource: the server responded with a status of 403   staticmap:1
()
```

API Key and Billing Errors

Under certain circumstances, a darkened map, or 'negative' Street View image, watermarked with the text "for development purposes only", may be displayed. This behavior typically indicates issues with either an API key or billing. In order to use Google Maps Platform products, billing must be enabled on your account, and all requests must include a valid API key. The following flow will help troubleshoot this:

- See below how to create a Google API Key and use it:
  - https://www.youtube.com/watch?v=B4p3A00uXAs
  - https://www.youtube.com/watch?v=2_HZObVbe-g

- Sources:
  - https://www.w3schools.com/html/html5_geolocation.asp
  - https://developers.google.com/maps

```html
<input type="button" onclick="getLocation()" value="Show my geolocation">
<p id="demo"></p>

<script>
var x = document.getElementById("demo");

function getLocation() {
  if (navigator.geolocation) { //if geolocation is supported
    navigator.geolocation.getCurrentPosition(yourOKCallBack, yourErrorCallBack);
  }
  else {  //if geolocation is not supported
    x.innerHTML = "Geolocation is not supported by this browser.";
  }
}

function yourOKCallBack(position) {
  x.innerHTML = "Latitude: " + position.coords.latitude +
  "<br>Longitude: " + position.coords.longitude;

  let your_loc_Key = position.coords.latitude + "," + position.coords.longitude;

  var img_url = "https://maps.googleapis.com/maps/api/staticmap?center="+
                    your_loc_Key+"&zoom=14&size=400x300&sensor=false&key=YOUR_KEY";

  x.innerHTML = "<img src='"+img_url+"'>";
}

function yourErrorCallBack(error) {
   x.innerHTML = error.message;
}
```

HTML DRAG AND DROP API

- One can **drag** *HTML elements* and *files*.
  - To make an HTML element draggable, set the **draggable attribute** to true.
  - You may want to also set up an id to identify those elements.

```
<h2 draggable="true">Draggable H2 element</h2>
<p  draggable="true">Draggable P element</p>
```

- Then, specify what happens when you drag an element
  - use the **ondragstart** attribute to call a function that specifies what is being dragged.

- Identify elements where items can be dropped.
  - Use the **ondragover event**
  - "By default, data/elements cannot be dropped in other elements. To allow a drop, we must prevent the default handling of the element."

- Do the actual drop:
  - Use the **ondrop event**

```
<head>
  <style>
      div {
        float: left;
        border: 10px solid blue;
        text-align: center;
      }
  </style>
</head>
<body>

<h2 id="myh2" draggable="true" ondragstart="whatIsBeingDragged(event)">Draggable H2 element</h2>
<p  id="my_p" draggable="true" ondragstart="whatIsBeingDragged(event)">Draggable P element</p>
<img id="my_i" src="img_w3slogo.gif" draggable="true" ondragstart="whatIsBeingDragged(event)">

<div ondragover="event.preventDefault();" ondrop="doTheDrop(event);">
    drag over me!
</div>

<script>
    function whatIsBeingDragged(ev) {
        ev.dataTransfer.setData("text", ev.target.id); //saving the id of the element dragged
    }

    function doTheDrop(ev) {
        //ev.preventDefault();
        let elemID = ev.dataTransfer.getData("text"); //getting the id of the element being dragged
        ev.target.appendChild(document.getElementById(elemID));
    }
</script>
```

- See an example:
  - https://www.w3schools.com/html/tryit.asp?filename=tryhtml5_draganddrop2
- Source: https://www.w3schools.com/html/html5_draganddrop.asp

HTML DRAG AND DROP API

- Let's modify the previous example so it works with files.

```
function doTheDrop(ev) {
  console.log('File(s) dropped');

  // Prevent default behavior (Prevent file from being opened)
  ev.preventDefault();

  if (ev.dataTransfer.items) {
      // Use DataTransferItemList interface to access the file(s)
      for (var i = 0; i < ev.dataTransfer.items.length; i++) {
        // If dropped items are files
        if (ev.dataTransfer.items[i].kind === 'file') {
          var file = ev.dataTransfer.items[i].getAsFile();
          console.log('... file[' + i + '].name = ' + file.name);
        }
        else {// otherwise, we assume we dragged elements ...
          let elemID = ev.dataTransfer.getData("text"); //getting the id of the element being dragged
          ev.target.appendChild(document.getElementById(elemID));
          console.log('element with id ' + elemID + ' was dragged into');

        }
      }
  } else {
      // Use DataTransfer interface to access the file(s)
      for (var i = 0; i < ev.dataTransfer.files.length; i++) {
        console.log('... file[' + i + '].name = ' + ev.dataTransfer.files[i].name);
      }
  }
}
```

```
<!DOCTYPE HTML>
<html>
<head>
  <style>
      div {
        float: left;
        border: 10px solid blue;
        text-align: center;
      }
  </style>
</head>
<body>

<h2 id="myh2" draggable="true" ondragstart="whatIsBeingDragged(event)">Draggable H2 element</h2>
<p  id="my_p" draggable="true" ondragstart="whatIsBeingDragged(event)">Draggable P element</p>
<img id="my_i" src="img_w3slogo.gif" draggable="true" ondragstart="whatIsBeingDragged(event)">

<div ondragover="event.preventDefault();" ondrop="doTheDrop(event);">
    drag over me!
</div>

<script>
    function whatIsBeingDragged(ev) {
        ev.dataTransfer.setData("text", ev.target.id); //saving the id of the element dragged
    }

    function doTheDrop(ev) {
      console.log('File(s) dropped');

      // Prevent default behavior (Prevent file from being opened)
      ev.preventDefault();

      if (ev.dataTransfer.items) {
          // Use DataTransferItemList interface to access the file(s)
          for (var i = 0; i < ev.dataTransfer.items.length; i++) {
            // If dropped items are files
            if (ev.dataTransfer.items[i].kind === 'file') {
              var file = ev.dataTransfer.items[i].getAsFile();
              console.log('... file[' + i + '].name = ' + file.name);
            }
            else {// otherwise, we assume we dragged elements ...
              let elemID = ev.dataTransfer.getData("text"); //getting the id of the element being dragged
              ev.target.appendChild(document.getElementById(elemID));
              console.log('element with id ' + elemID + ' was dragged into');

            }
          }
      } else {
          // Use DataTransfer interface to access the file(s)
          for (var i = 0; i < ev.dataTransfer.files.length; i++) {
            console.log('... file[' + i + '].name = ' + ev.dataTransfer.files[i].name);
          }
      }
    }
</script>

</body>
</html>
```

- Source:
  - https://developer.mozilla.org/en-US/docs/Web/API/HTML_Drag_and_Drop_API/File_drag_and_drop

FILEREADER

- Use **FileReader** to read the contents of a file/blob.
  - Use **readAsDataURL** method to get the URL of the source file …
  - The results are stored in the **result** property.

```
<!DOCTYPE HTML>
<html>
<head>
</head>
<body>

<input type='file' accept='image/*' onchange='getTheSelectedFile(event)'><br>
<img id="demo">

<script>
  function getTheSelectedFile(event) {
        let reader = new FileReader();  //create a FileReader object

        reader.onload = function(){     //define it's onload method
           var dataURL = reader.result; //use the result property to get the url
           document.getElementById("demo").src = dataURL;
        };

         reader.readAsDataURL(event.target.files[0]);
  };
</script>


</body>
</html>
```

- Source: https://www.javascripture.com/FileReader

FILEREADER

- Use **FileReader** to read the contents of a file/blob.
  - Use **readAsText** method to get the contents of a file/blob
  - The results are stored in the **result** property.

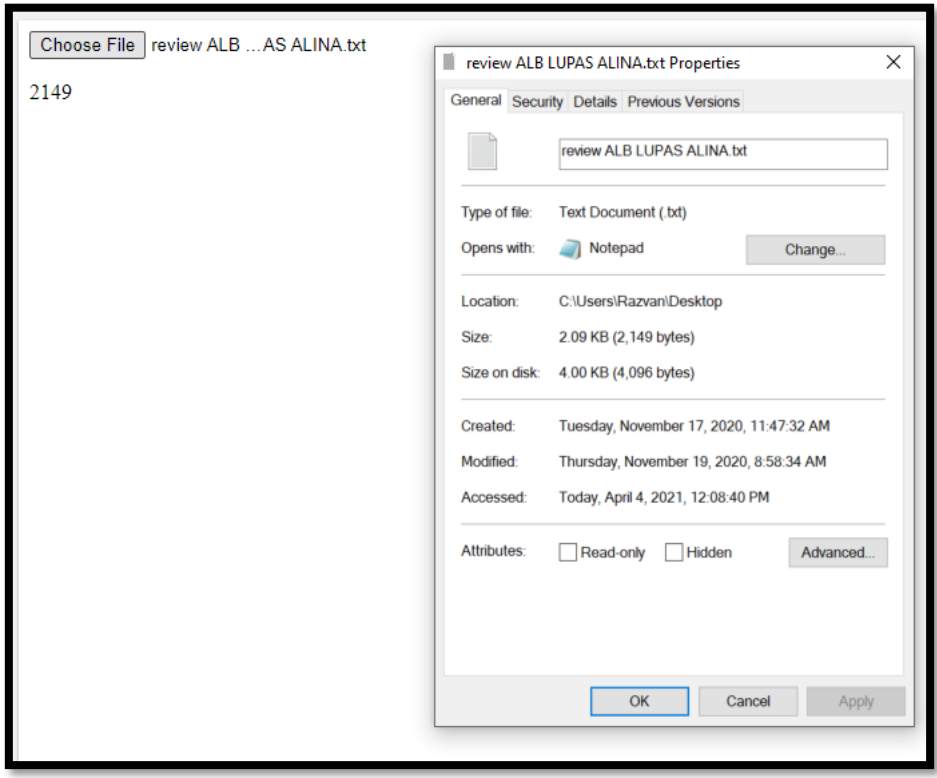
```
<!DOCTYPE HTML>
<html>
<head>
</head>
<body>

<input type='file' accept='text/*' onchange='getTheSelectedFile(event)'><br>
<p id="demo">

<script>
  function getTheSelectedFile(event) {
        let reader = new FileReader();  //create a FileReader object

        reader.onload = function(){     //define it's onload method
           var dataText = reader.result; //use the result property to get the text
           document.getElementById("demo").innerHTML = dataText;
        };

         reader.readAsText(event.target.files[0]);
  };
</script>


</body>
</html>
```

- Source: https://www.javascripture.com/FileReader

FILEREADER

- Use **FileReader** to read the contents of a file/blob.
  - Use **readAsArrayBuffer** method to get the contents of a file/blob, and makes them available as an **ArrayBuffer**
  - The results are stored in the **result** property.

```
<!DOCTYPE HTML>
<html>
<head>
</head>
<body>

<input type='file'  onchange='getTheSelectedFile(event)'><br>
<p id="demo">

<script>
  function getTheSelectedFile(event) {
        let reader = new FileReader();  //create a FileReader object

        reader.onload = function(){     //define it's onload method
           var anArrayBuffer = reader.result; //use the result property to get the ArrayBuffer
           document.getElementById("demo").innerHTML = anArrayBuffer.byteLength;
        };

         reader.readAsArrayBuffer(event.target.files[0]);
  };
</script>


</body>
</html>
```



- Source: https://www.javascripture.com/FileReader

FILEREADER

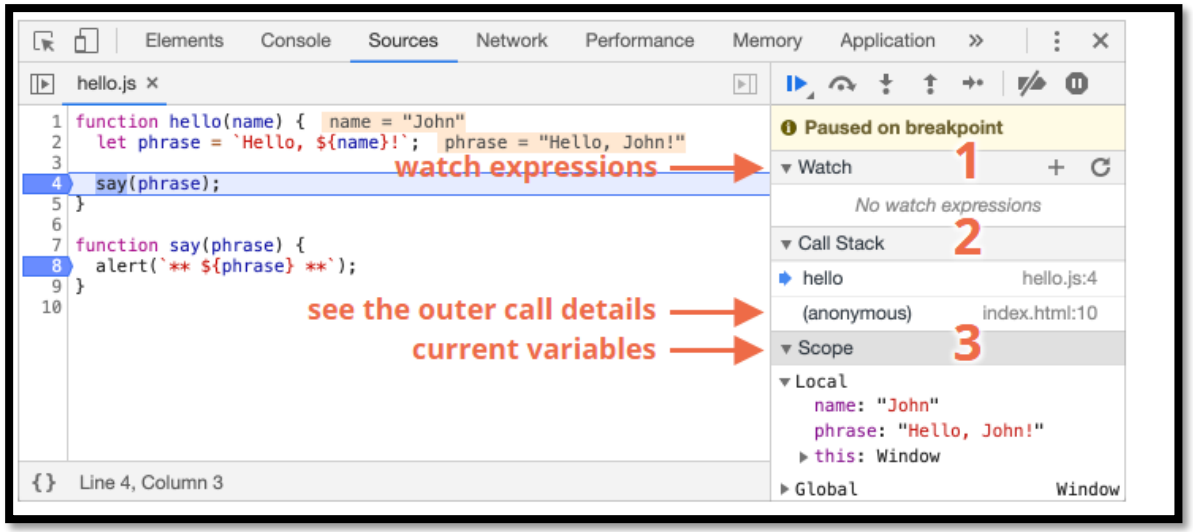
- On your own, you may want to check out the other **events**
  - onabort ← Called when the **read** operation is **aborted** with abort().
  - onerror ← Called when **there is an error** during the **load**.
  - onload ← Called when a **read** operation **successfully completes**.
  - onloadend ← Called after a **read completes** (*successfully* or *unsuccessfully*).
  - onloadstart ← Called after **starting** a **read** operation.
  - onprogress ← Called **during** a **read** operation to report the current progress.

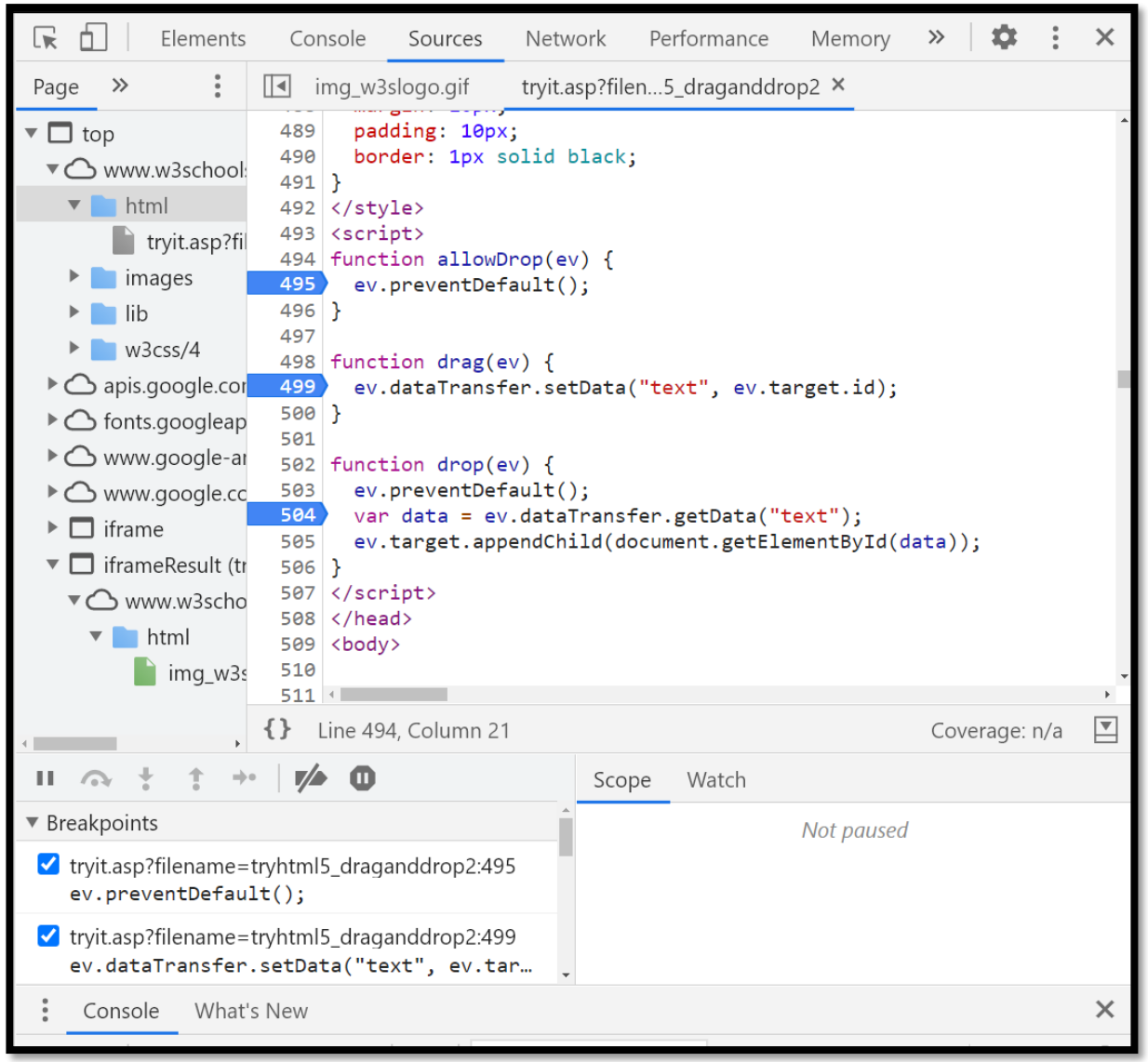
- If time, add event handlers for all the above events (and use **console.log** to see when they fire up)

- Source: https://www.javascripture.com/FileReader

DEBUGGING IN CHROME

- You may want to check this on your own …





- Source: https://javascript.info/debugging-chrome

LAB / HOMEWORK

- **Module 08**
  - Exercise 1: Dragging and Dropping Images
  - Exercise 2: Incorporating Video
  - Exercise 3: Using the Geolocation API to Report the User's Current Location

- You will find the **high-level** steps on the following page:
  https://github.com/MicrosoftLearning/20480-Programming-in-HTML5-with-JavaScript-and-CSS3/blob/master/Instructions/20480C_MOD08_LAB_MANUAL.md
- You will find the **detailed** steps on the following page:
  https://github.com/MicrosoftLearning/20480-Programming-in-HTML5-with-JavaScript-and-CSS3/blob/master/Instructions/20480C_MOD08_LAK.md

- For your homework submit one zipped folder with your complete solution.