INT106 Lab – HTML Canvas, Geolocation, Media and Local storage.

**HTML Canvas**

|  |
| --- |
| **Canvas** is a medium for oil painting but that is not what we are going to learn about.  On the HTML canvas, you can draw all kinds of graphics, from simple lines, to complex graphic objects.  **The HTML <canvas> element** (introduced in HTML5) is a container for canvas graphics.  An HTML canvas is a rectangular area on an HTML page, specified with the <canvas> element.  Canvas has several methods for drawing paths, boxes, circles, text, and graphic images.  HTML canvas is a set of JavaScript methods (APIs) for drawing graphics (lines, boxes, circles, shapes). |

|  |
| --- |
| **Canvas example:**  <!DOCTYPE html>  <html>  <body>  <canvas id="myCanvas" width="200" height="100"  style="border:1px solid #000000;">  Your browser does not support the HTML5 canvas tag.  </canvas>  </body>  </html> |

|  |
| --- |
| **Drawing with JavaScript:**  <!DOCTYPE html>  <html>  <body>  <canvas id="myCanvas" width="200" height="100"  style="border:1px solid #c3c3c3;">  Your browser does not support the HTML5 canvas tag.  </canvas>  <script>  var c = document.getElementById("myCanvas");  var ctx = c.getContext("2d");  ctx.fillStyle = "#FF0000";  ctx.fillRect(0,0,150,75);  </script>  </body>  </html> |

|  |
| --- |
| **Draw a Line:**  <!DOCTYPE html>  <html>  <body>  <canvas id="myCanvas" width="200" height="100"  style="border:1px solid #d3d3d3;">  Your browser does not support the HTML5 canvas tag.</canvas>  <script>  var c = document.getElementById("myCanvas");  var ctx = c.getContext("2d");  ctx.moveTo(0,0);  ctx.lineTo(200,100);  ctx.stroke();  </script>  </body>  </html> |

|  |
| --- |
| **Draw a Circle:**  <!DOCTYPE html>  <html>  <body>  <canvas id="myCanvas" width="200" height="100"  style="border:1px solid #d3d3d3;">  Your browser does not support the HTML5 canvas tag.</canvas>  <script>  var c = document.getElementById("myCanvas");  var ctx = c.getContext("2d");  ctx.beginPath();  ctx.arc(95,50,40,0,2\*Math.PI);  ctx.stroke();  </script>  </body>  </html> |

|  |
| --- |
| **Draw a Text:**  <!DOCTYPE html>  <html>  <body>  <canvas id="myCanvas" width="200" height="100"  style="border:1px solid #d3d3d3;">  Your browser does not support the HTML5 canvas tag.</canvas>  <script>  var c = document.getElementById("myCanvas");  var ctx = c.getContext("2d");  ctx.font = "30px Arial";  ctx.fillText("Hello World",10,50);  </script>  </body>  </html> |

|  |
| --- |
| **Stroke Text:**  <!DOCTYPE html>  <html>  <body>  <canvas id="myCanvas" width="200" height="100"  style="border:1px solid #d3d3d3;">  Your browser does not support the HTML5 canvas tag.</canvas>  <script>  var c = document.getElementById("myCanvas");  var ctx = c.getContext("2d");  ctx.font = "30px Arial";  ctx.strokeText("Hello World",10,50);  </script>  </body>  </html> |

|  |
| --- |
| **Draw Gradient:**  <!DOCTYPE html>  <html>  <body>  <canvas id="myCanvas" width="200" height="100"  style="border:1px solid #d3d3d3;">  Your browser does not support the HTML5 canvas tag.</canvas>  <script>  var c = document.getElementById("myCanvas");  var ctx = c.getContext("2d");  // Create gradient  var grd = ctx.createLinearGradient(0,0,200,0);  grd.addColorStop(0,"red");  grd.addColorStop(1,"white");  // Fill with gradient  ctx.fillStyle = grd;  ctx.fillRect(10,10,150,80);  </script>  </body>  </html> |

|  |
| --- |
| **Draw Circular Gradient:**  <!DOCTYPE html>  <html>  <body>  <canvas id="myCanvas" width="200" height="100"  style="border:1px solid #d3d3d3;">  Your browser does not support the HTML5 canvas tag.</canvas>  <script>  var c = document.getElementById("myCanvas");  var ctx = c.getContext("2d");  // Create gradient  var grd = ctx.createRadialGradient(75,50,5,90,60,100);  grd.addColorStop(0,"red");  grd.addColorStop(1,"white");  // Fill with gradient  ctx.arc(50, 50, 50, 0, 2\*Math.PI);  ctx.fillStyle = grd;  ctx.fill();  </script>  </body>  </html> |

|  |
| --- |
| **Draw Image:**  <!DOCTYPE html>  <html>  <body>  <p>Image to use:</p>  <img id="scream" src="img\_the\_scream.jpg" alt="The Scream"  width="220" height="277">  <p>Canvas to fill:</p>  <canvas id="myCanvas" width="250" height="300"  style="border:1px solid #d3d3d3;">  Your browser does not support the HTML5 canvas tag.</canvas>  <p><button onclick="myCanvas()">Try it</button></p>  <script>  function myCanvas() {  var c = document.getElementById("myCanvas");  var ctx = c.getContext("2d");  var img = document.getElementById("scream");  ctx.drawImage(img,10,10);  }  </script>  </body>  </html> |

**What can Canvas do?**

* HTML Canvas Can Draw Text
  + Canvas can draw colorful text, with or without animation.
* HTML Canvas Can Draw Graphics
  + Canvas has great features for graphical data presentation with an imagery of graphs and charts.
* HTML Canvas Can be Animated
  + Canvas objects can move. Everything is possible: from simple bouncing balls to complex animations.
* HTML Canvas Can be Interactive
  + Canvas is drawn by JavaScript. Because of this, canvas can respond to JavaScript events.
  + Canvas can listen for, and respond to, any user action (key clicks, mouse clicks, button clicks, finger movement).
* HTML Canvas Can Play Games
  + Canvas methods for animations, offer lots of possibilities for HTML gaming applications.

|  |
| --- |
| **The <canvas> element** must have an **id attribute** so it can be referred to by JavaScript.  The **width** and **height** attribute is necessary to define the size of the canvas.  To add a border, use a style attribute:  Example:  <canvas id="myCanvas" width="200" height="100"  style="border:1px solid #000000;">  </canvas> |

|  |
| --- |
| **All drawing on the HTML canvas must be done with JavaScript:** var canvas = document.getElementById("myCanvas"); //Get the canvas name  var ctx = canvas.getContext("2d"); //Define a new canvas object name  ctx.fillStyle = "#FF0000"; // Choose the color for our object  ctx.fillRect(0,0,150,75); // Draw our object (a Rectangle) |

|  |
| --- |
| **The HTML canvas is two-dimensional.**  The upper-left corner of the canvas has the coordinates (0,0)  In the previous chapter, you saw this method used: fillRect(0,0,150,75).  This means: Start at the upper-left corner (0,0) and draw a 150 x 75 pixels rectangle. |

|  |
| --- |
| **Draw a Line:**  var canvas = document.getElementById("myCanvas");  var ctx = canvas.getContext("2d");  ctx.moveTo(0,0); //Define a starting point  ctx.lineTo(200,100); //Define a ending point  ctx.stroke(); // Draw the line between 2 points |

|  |
| --- |
| **Draw a Circle:**  var canvas = document.getElementById("myCanvas");  var ctx = canvas.getContext("2d");  ctx.beginPath();  ctx.arc(95,50,40,0,2\*Math.PI); //arc(x,y,r,start,stop)  ctx.stroke(); |

|  |
| --- |
| **Canvas – Gradient**  Gradients can be used to fill rectangles, circles, lines, text, etc. Shapes on the canvas are not limited to solid colors.  There are two different types of gradients:  createLinearGradient(x,y,x1,y1) - Creates a linear gradient  createRadialGradient(x,y,r,x1,y1,r1) - Creates a radial/circular gradient  **Example for Linear Gradient:**  <!DOCTYPE html>  <html>  <body>  <canvas id="myCanvas" width="200" height="100"  style="border:1px solid #d3d3d3;">  Your browser does not support the HTML5 canvas tag.</canvas>  <script>  var c = document.getElementById("myCanvas");  var ctx = c.getContext("2d");  // Create gradient  var grd = ctx.createLinearGradient(0,0,200,0);  grd.addColorStop(0,"red");  grd.addColorStop(1,"white");  // Fill with gradient  ctx.fillStyle = grd;  ctx.fillRect(10,10,150,80);  </script>  </body>  </html>  **Example of Circular Gradient:**  <!DOCTYPE html>  <html>  <body>  <canvas id="myCanvas" width="200" height="100"  style="border:1px solid #d3d3d3;">  Your browser does not support the HTML5 canvas tag.</canvas>  <script>  var c = document.getElementById("myCanvas");  var ctx = c.getContext("2d");  // Create gradient  var grd = ctx.createRadialGradient(75,50,5,90,60,100);  grd.addColorStop(0,"red");  grd.addColorStop(1,"white");  // Fill with gradient  ctx.arc(50, 50, 50, 0, 2\*Math.PI);  ctx.fillStyle = grd;  ctx.fill();  </script>  </body>  </html> |

|  |
| --- |
| **Drawing Text**  To draw text on a canvas, the most important property and methods are:  font - defines the font properties for the text  strokeText(text,x,y) - Draws text on the canvas  fillText(text,x,y) - Draws "filled" text on the canvas  Example:  var canvas = document.getElementById("myCanvas");  var ctx = canvas.getContext("2d");  ctx.font = "30px Arial";  ctx.strokeText("Hello World",10,50);  ctx.fillText("Hello World",10,50); |

|  |
| --- |
| Practice #1  1.1 Draw a following picture.  1.2 Draw 4 lines across the circle imitating pizza slicing. |

**HTML Geolocation**

The HTML Geolocation API is used to get the geographical position of a user.

Since this can compromise user privacy, the position is not available unless the user approves it.

|  |
| --- |
| **Using HTML Geolocation**  Use the **getCurrentPosition()** method to get the user's position.  The example below is a simple Geolocation example returning the latitude and longitude of the user's position:   <!DOCTYPE html>  <html>  <body>  <p>Click the button to get your coordinates.</p>  <button onclick="getLocation()">Try It</button>  <p id="demo"></p>  <script>  var x = document.getElementById("demo");  function getLocation() {  if (navigator.geolocation) {  navigator.geolocation.getCurrentPosition(showPosition);  } else {  x.innerHTML = "Geolocation is not supported by this browser.";  } }  function showPosition(position) {  x.innerHTML = "Latitude: " + position.coords.latitude +  "<br>Longitude: " + position.coords.longitude;  }  </script>  </body>  </html>  **What does this code do?**   * Check if Geolocation is supported * If supported, run the getCurrentPosition() method. If not, display a message to the user * If the getCurrentPosition() method is successful, it returns a coordinates object to the function specified in the parameter ( showPosition ) * The showPosition() function gets the displays the Latitude and Longitude |

**Handling Errors and Rejections**

The example above is a very basic Geolocation script, with no error handling.

The second parameter of the getCurrentPosition() method is used to handle errors. It specifies a function to run if it fails to get the user's location:

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <p>Click the button to get your coordinates.</p>  <button onclick="getLocation()">Try It</button>  <p id="demo"></p>  <script>  var x = document.getElementById("demo");  function getLocation() {  if (navigator.geolocation) {  navigator.geolocation.getCurrentPosition(showPosition, showError);  } else {  x.innerHTML = "Geolocation is not supported by this browser.";  }  }  function showPosition(position) {  x.innerHTML = "Latitude: " + position.coords.latitude +  "<br>Longitude: " + position.coords.longitude;  }  **function showError(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>  </body>  </html>  **Error Codes:**   * Permission denied - The user did not allow Geolocation * Position unavailable - It is not possible to get the current location * Timeout - The operation timed out |

**Displaying the Result in a Map**

To display the result in a map, you need access to a map service that can use latitude and longitude, like Google Maps:

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <p id="demo">Click the button to get your position.</p>  <button onclick="getLocation()">Try It</button>  <div id="mapholder"></div>  <script>  var x = document.getElementById("demo");  function getLocation() {  if (navigator.geolocation) {  navigator.geolocation.getCurrentPosition(showPosition, showError);  } else {  x.innerHTML = "Geolocation is not supported by this browser.";  }  }  function showPosition(position) {  var latlon = position.coords.latitude + "," + position.coords.longitude;  var img url = "http://maps.googleapis.com/maps/api/staticmap?center="  +latlon+"&zoom=14&size=400x300&sensor=false";  document.getElementById("mapholder").innerHTML = "<img src='"+img\_url+"'>";  }  function showError(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>  </body>  </html> |

|  |
| --- |
| **The getCurrentPosition() Method - Return Data**  The getCurrentPosition() method returns an object if it is successful. The latitude, longitude and accuracy properties are always returned. The other properties below are returned if available.  Property Description   * 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 above the mean sea level * coords.altitudeAccuracy The altitude accuracy of position * coords.heading The heading as degrees clockwise from North * coords.speed The speed in meters per second * timestamp The date/time of the response |

**Geolocation object - Other interesting Methods**

**watchPosition()** - Returns the current position of the user and continues to return updated position as the user moves (like the GPS in a car).

**clearWatch()** - Stops the watchPosition() method.

The example below shows the watchPosition() method. You need an accurate GPS device to test this (like iPhone):

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <p>Click the button to get your coordinates.</p>  <button onclick="getLocation()">Try It</button>  <p id="demo"></p>  <script>  var x = document.getElementById("demo");  function getLocation() {  if (navigator.geolocation) {  navigator.geolocation.**watchPosition**(showPosition);  } else {  x.innerHTML = "Geolocation is not supported by this browser.";}  }    function showPosition(position) {  x.innerHTML="Latitude: " + position.coords.latitude +  "<br>Longitude: " + position.coords.longitude;  }  </script>  </body>  </html> |

**HTML Multimedia**

**What is Multimedia?**

Multimedia comes in many different formats. It can be almost anything you can hear or see.

Examples: Pictures, music, sound, videos, records, films, animations, and more.

Web pages often contains multimedia elements of different types and formats.

**Multimedia Formats**

Multimedia elements (like sounds or videos) are stored in media files.

The most common way to discover the type of a file, is to look at the file extension. When a browser sees the file extension .htm or .html, it will treat the file as an HTML file. The .xml extension indicates an XML file, and the .css extension indicates a style sheet file. Pictures are recognized by extensions like .gif, .png and .jpg.

Multimedia files also have their own formats and different extensions like: .swf, .wav, .mp3, .mp4, .mpg, .wmv, and .avi.

Multimedia Formats

There are many multimedia formats but only the followings are supported by the newest HTML5 standard.

Video files: MP4, WebM and Ogg

Sound files: MP3, WAV and Ogg

Video Example:  
<http://www.w3schools.com/html/tryit.asp?filename=tryhtml5_video>  
Try this one.

The Code:

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <video width="400" controls>  <source src="mov\_bbb.mp4" type="video/mp4">  <source src="mov\_bbb.ogg" type="video/ogg">  Your browser does not support HTML5 video.  </video>  <p>  Video courtesy of  <a href="http://www.bigbuckbunny.org/" target="\_blank">Big Buck Bunny</a>.  </p>  </body>  </html> |

**How it Works**

* The controls attribute adds video controls, like play, pause, and volume.
* It is a good idea to always include width and height attributes.
* If height and width are not set, the browser does not know the size of the video. The effect will be that the page will change (or flicker) while the video loads.
* Text between the <video> and </video> tags will only display in browsers that do not support the <video> element.
* Multiple <source> elements can link to different video files. The browser will use the first recognized format.

**HTML <video> Autoplay**

To start a video automatically use the autoplay attribute:

<http://www.w3schools.com/html/tryit.asp?filename=tryhtml5_video_autoplay>

|  |
| --- |
| <video width="320" height="240" **autoplay**>  <source src="movie.mp4" type="video/mp4">  <source src="movie.ogg" type="video/ogg">  Your browser does not support the video tag.  </video> |

**HTML5 Video Tags**

<video> Defines a video or movie

<source> Defines multiple media resources for media elements, such as <video> and <audio>

<track> Defines text tracks in media players

**The HTML <audio> Element**

To play an audio file in HTML, use the <audio> element:

<http://www.w3schools.com/html/tryit.asp?filename=tryhtml5_audio_all>

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <audio controls>  <source src="horse.ogg" type="audio/ogg">  <source src="horse.mp3" type="audio/mpeg">  Your browser does not support the audio element.  </audio>  </body>  </html> |

**HTML Audio - How It Works**

* The controls attribute adds audio controls, like play, pause, and volume.
* Text between the <audio> and </audio> tags will display in browsers that do not support the <audio> element.
* Multiple <source> elements can link to different audio files. The browser will use the first recognized format.

**HTML5 Audio Tags**

<audio> Defines sound content

<source> Defines multiple media resources for media elements, such as <video> and <audio>

**HTML5 Local Storage**

**What is HTML Local Storage?**

With local storage, web applications can store data locally within the user's browser.

Before HTML5, application data had to be stored in **cookies**, included in every server request. Local storage is more secure, and large amounts of data can be stored locally, without affecting website performance.

Unlike cookies, the storage limit is far larger (at least 5MB) and information is never transferred to the server.

Local storage is per domain. All pages, from one domain, can store and access the same data.

**HTML Local Storage Objects**

HTML local storage provides two objects for storing data on the client:

window.localStorage - stores data with no expiration date

window.sessionStorage - stores data for one session (data is lost when the tab is closed)

Before using local storage, check browser support for localStorage and sessionStorage:

|  |
| --- |
| if(typeof(Storage) !== "undefined") {  // Code for localStorage/sessionStorage.  } else {  // Sorry! No Web Storage support..  } |

**The localStorage Object**

The localStorage object stores the data with no expiration date. The data will not be deleted when the browser is closed, and will be available the next day, week, or year.

|  |
| --- |
| <!DOCTYPE html>  <html>  <body>  <div id="result"></div>  <script>  // Check browser support  if (typeof(Storage) != "undefined") {  // Store  localStorage.setItem("lastname", "Smith");  // Retrieve  document.getElementById("result").innerHTML = localStorage.getItem("lastname");  } else {  document.getElementById("result").innerHTML = "Sorry, your browser does not support Web Storage...";  }  </script>  </body>  </html> |

**Example explained:**

* Create a localStorage name/value pair with name="lastname" and value="Smith"
* Retrieve the value of "lastname" and insert it into the element with id="result"

**The example above could also be written like this:**

|  |
| --- |
| // Store  localStorage.lastname = "Smith";  // Retrieve  document.getElementById("result").innerHTML = localStorage.lastname; |

**The syntax for removing the "lastname" localStorage item is as follows:**

|  |
| --- |
| localStorage.removeItem("lastname"); |

**Note:** Name/value pairs are always stored as strings. Remember to convert them to another format when needed!

**The sessionStorage Object**

The sessionStorage object is equal to the localStorage object, except that it stores the data for only one session. The data is deleted when the user closes the browser window.

The following example counts the number of times a user has clicked a button, in the current session:

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <script>  function clickCounter() {  if(typeof(Storage) !== "undefined") {  if (sessionStorage.clickcount) {  sessionStorage.clickcount = Number(sessionStorage.clickcount)+1;  } else {  sessionStorage.clickcount = 1;  }  document.getElementById("result").innerHTML = "You have clicked the button " + sessionStorage.clickcount + " time(s) in this session.";  } else {  document.getElementById("result").innerHTML = "Sorry, your browser does not support web storage...";  }  }  </script>  </head>  <body>  <p><button onclick="clickCounter()" type="button">Click me!</button></p>  <div id="result"></div>  <p>Click the button to see the counter increase.</p>  <p>Close the browser tab (or window), and try again, and the counter is reset.</p>  </body>  </html> |