



More HTML5 Features

2020/21 COMP3322 Modern Technologies on WWW

Content

- HTML5 Canvas
- HTML5 Audio & Video
- HTML5 Drag & Drop
- HTML5 Local Storage
- HTML5 Geolocation


HTML5 Canvas

- The <canvas> element is a **two-dimension drawing surface** that uses JavaScript coding to **perform drawing**:
 - for example: creating lines, shapes, fills, text, animations, and so on.
 - all using the native browser behavior and without proprietary plug-ins like Flash.
- Add a canvas space with the <canvas> element and specify the dimension with the width and height attributes.
 - By default, no border and no content.
 - For browsers that don't support the canvas element, provide some fallback content inside the tags.

```
<canvas width="150" height="150">
```

```
  The browser does not support HTML5 canvas.
```

```
</canvas>
```



Fallback
content

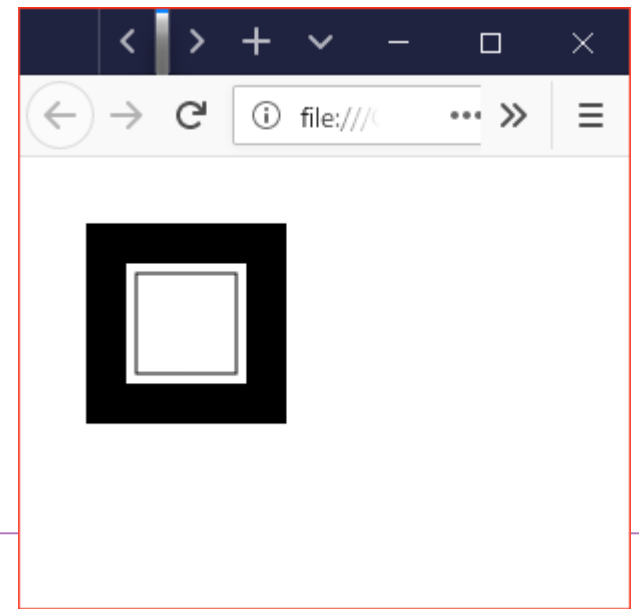
HTML5 Canvas

- To display something, a script first needs to access the rendering context and draw on it.
- Example to draw the rectangles:

Obtain the rendering context
and its drawing functions

Drawing

Define the space



```
<html>
<head>
<script>
  function draw() {
    var canvas = document.getElementById('canvas');
    if (canvas.getContext) {
      var ctx = canvas.getContext('2d');
      ctx.fillRect(25, 25, 100, 100);
      ctx.clearRect(45, 45, 60, 60);
      ctx.strokeRect(50, 50, 50, 50);
    }
  }
</script>
</head>
<body onload="draw();" >
  <canvas id="canvas" width="150" height="150">
  </canvas>
</body>
</html>
```

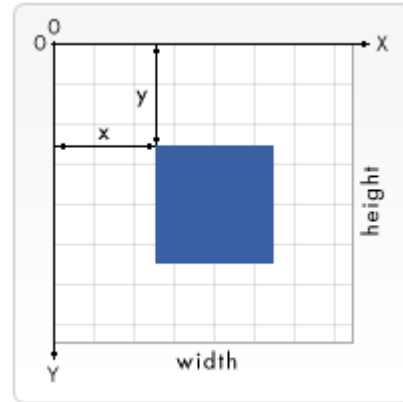
Rendering Contexts

```
canvas.getContext(contextType);
```

- The `getContext()` method returns an object that provides methods and properties for drawing on the canvas.
- There are currently two widely supported contexts: "2d" for 2-D graphics and "webgl" for 3-D graphics through the OpenGL interface.
 - Only "2d" will be discussed.

Drawing

- Canvas grid



- Drawing Rectangle

Method	Description
fillRect(x,y,w,h)	Draws a rectangle and fill with color (default: black) starting at coordinate: (x,y) with width w and height h
strokeRect(x,y,w,h)	Draws a rectangular outline.
clearRect(x,y,w,h)	Clears the rectangle defined by (x,y) and size (w,h)

- Define a path and draw lines/curves/arcs

Method	Description
beginPath()	Starts a path drawing
closePath()	Ends a path drawing
moveTo(x₁,y₁)	Moves to location (x ₁ ,y ₁) without drawing
lineTo(x₂,y₂)	Sets the end point of the line to (x ₂ ,y ₂)
arc(x,y,r,s,e,d)	Draws a circle or arc path, centre at (x,y) with radius r, starting the arc at angle s (in radians), and ending at angle e. [Default of d is false which is clockwise direction]
quadraticCurveTo()	Creates a quadratic Bézier curve path
bezierCurveTo()	Creates a cubic Bézier curve path
stroke()	Draws the path
fill()	Fills the current drawing

Text and Color

Method / Property	Description
font	Sets or returns the current font properties for text content (same syntax as the CSS font property)
textAlign	Sets or returns the current alignment (start end center left right) for the text content. The alignment is relative to the x value of the fillText() method. https://www.w3schools.com/tags/tryit.asp?filename=tryhtml5_canvas_textalign
fillText(text,x,y)	Draws and fills text on the canvas
strokeText(text,x,y)	Draws text on the canvas (no fill)
fillStyle	Sets or returns the color, gradient, or pattern used to fill the drawing
strokeStyle	Sets or returns the color, gradient, or pattern used for strokes
createLinearGradient()	Creates a linear gradient (to use on canvas content)
createRadialGradient()	Creates a radial/circular gradient (to use on canvas content)

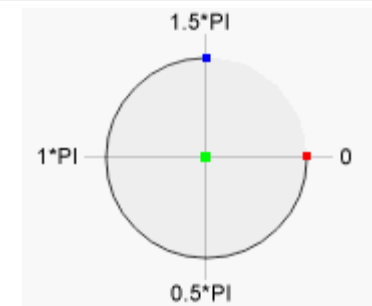
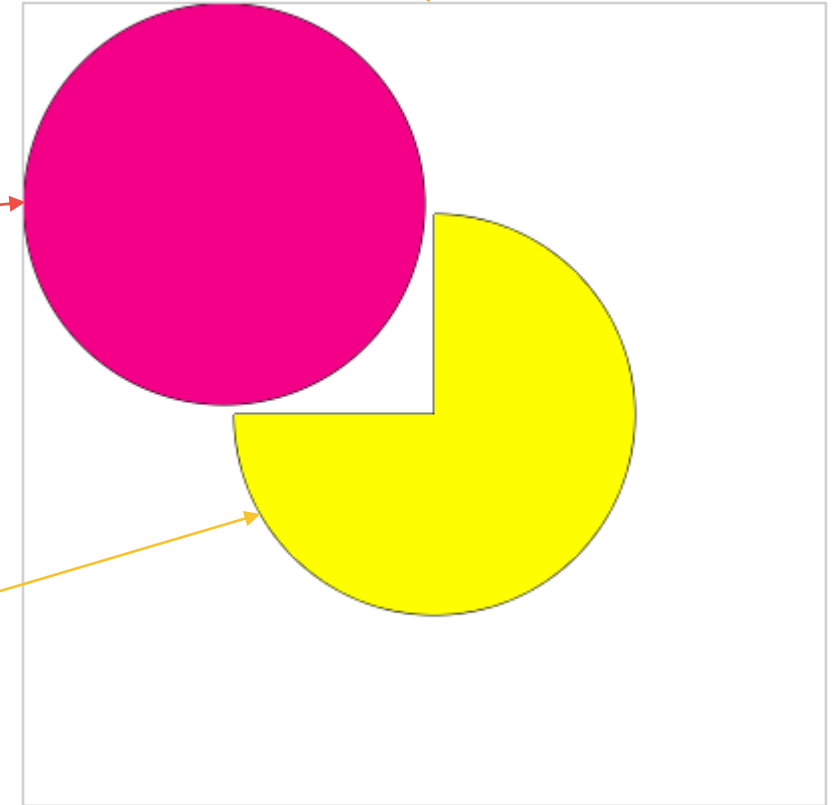
Example

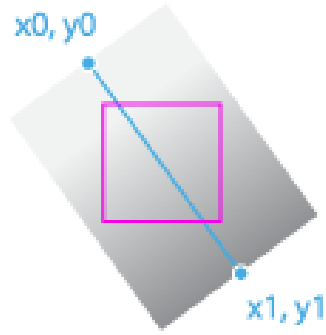
```
<script>
  var canvas1 = document.getElementById("Canvas1");
  var ctx = canvas1.getContext("2d");
```

```
  ctx.beginPath();
  ctx.arc(100,100,100,0,2*Math.PI);
  ctx.stroke();
  ctx.fillStyle = "#FF0088";
  ctx.fill();
```

```
  ctx.beginPath();
  ctx.moveTo(205,205);
  ctx.arc(205,205,100,1.5*Math.PI,Math.PI);
  ctx.closePath();
  ctx.stroke();
  ctx.fillStyle = "#FFFF00";
  ctx.fill();
```

```
<canvas id="Canvas1" width="400" height="400"
style="border:1px solid #c3c3c3;">
  Your browser does not support the canvas element.
</canvas>
```





```
var grd = ctx.createLinearGradient(0, 305, 0, 380);  
grd.addColorStop(0, "#FF0088");  
grd.addColorStop(1, "#FFFFFF");  
  
ctx.font = "bold 100px Candara";  
ctx.strokeText("PacMan!!", 10, 380);  
ctx.fillStyle = grd;  
ctx.fillText("PacMan!!", 10, 380);
```

</script>



Images

- Using the `drawImage()` method, we can take an image object and draw it onto a canvas.
 - The image can be loaded from a server or even extracted from a canvas.
- Importing images into a canvas involves two steps:
 - **Get a reference** to an `HTMLImageElement` object or to another canvas element as a source.
 - It is also possible to use images by providing a URL.
 - **Draw the image** on the canvas using the `drawImage()` function.

Example

```
context.drawImage(img,x,y);
```

```
<canvas id="Canvas1" width="350" height="350"  
style="border:1px solid #c3c3c3;">
```

Your browser does not support the canvas element.

```
</canvas>
```

```
<script>
```

```
draw();
```

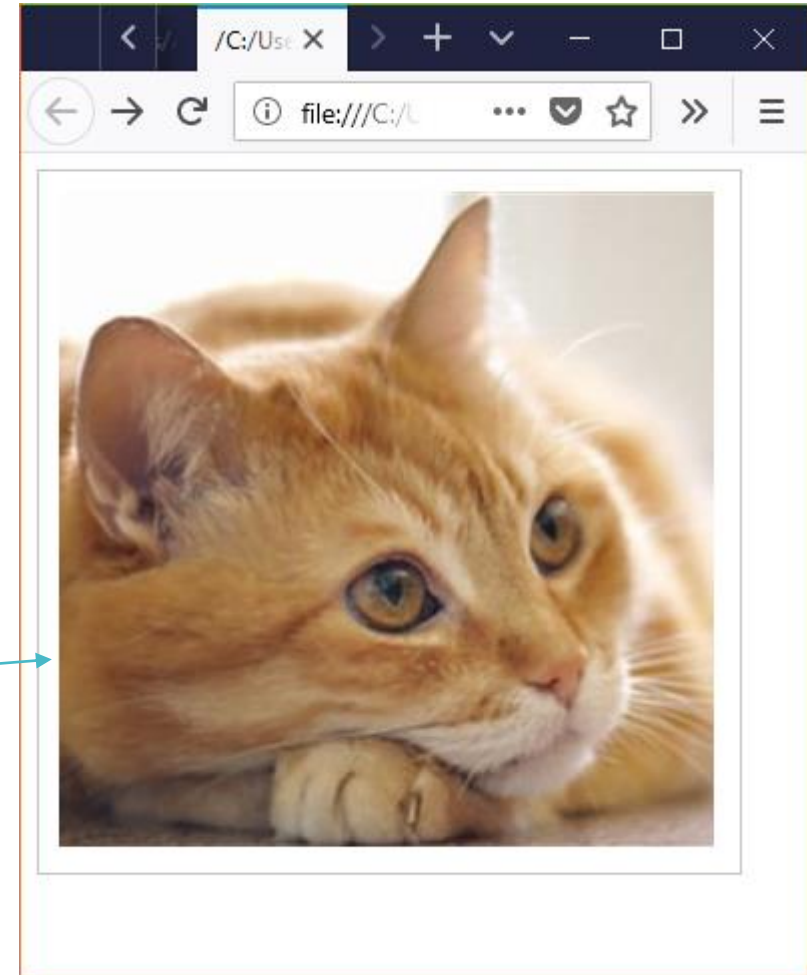
```
function draw() {
```

```
var ctx = document.getElementById('Canvas1')  
        .getContext('2d');
```

```
var img = new Image();  
img.onload = function() {  
    ctx.drawImage(img, 10, 10);  
};  
img.src = 'cat.jpg';
```

```
}
```

```
</script>
```



Example

```
context.drawImage(img, x, y, width, height);
```

```

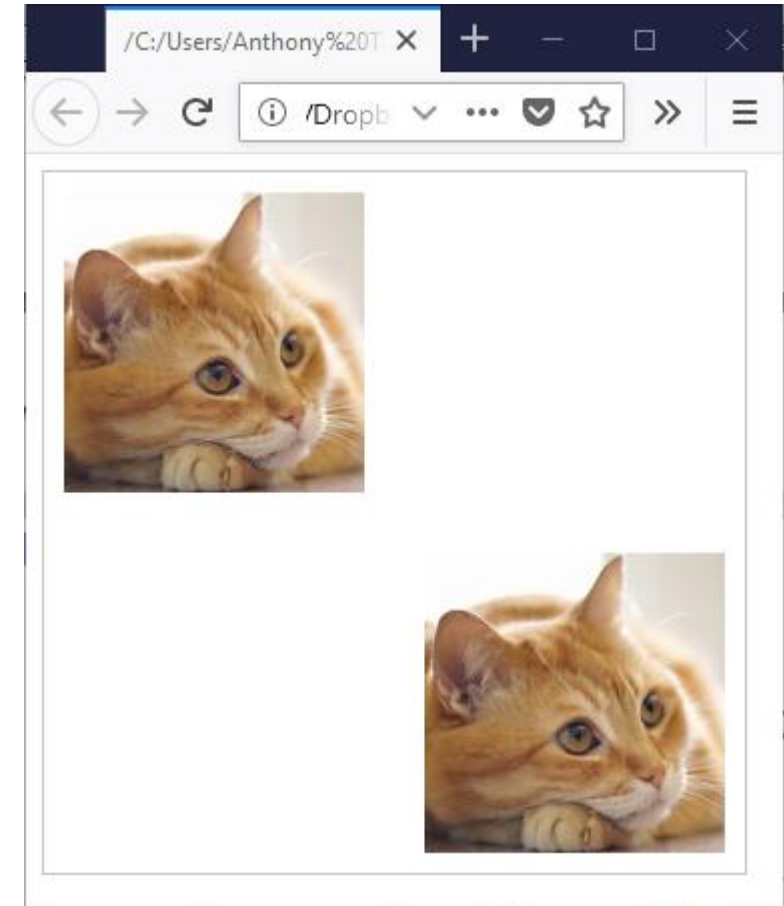
```

```
<canvas id="Canvas1" width="350" height="350"
       style="border:1px solid #c3c3c3;">
```

Your browser does not support the canvas element.

```
</canvas>
```

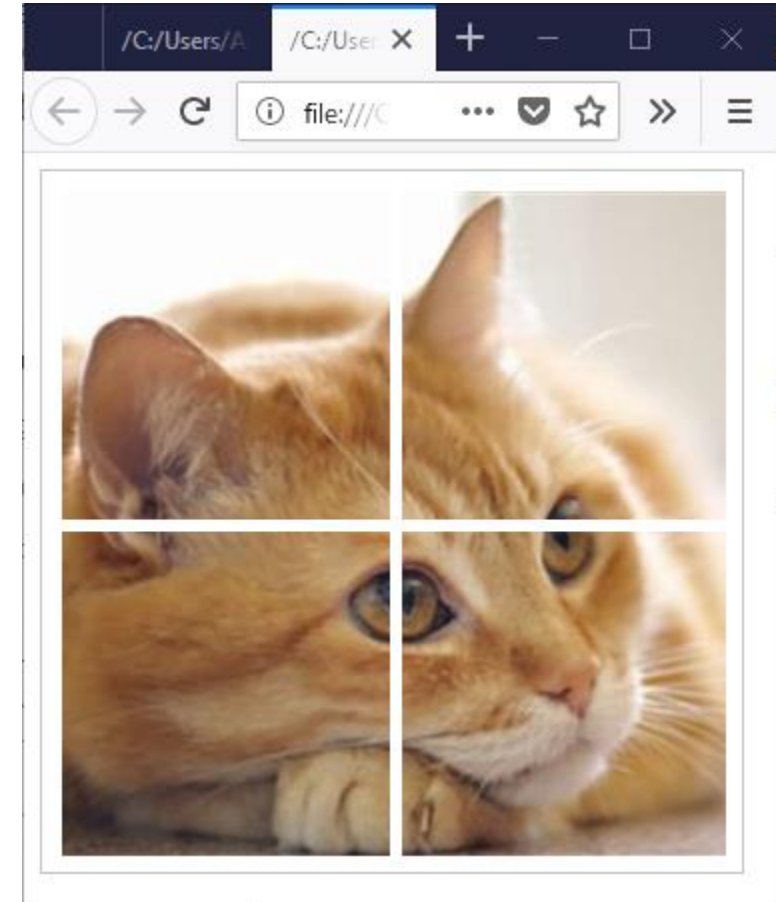
```
<script>
  draw();
  function draw() {
    var ctx = document.getElementById('Canvas1')
               .getContext('2d');
    var img = document.getElementById("photo1");
    img.onload = function() {
      ctx.drawImage(img, 10, 10, 150, 150);
      ctx.drawImage(img, 190, 190, 150, 150);
    };
  }
</script>
```



Example

```
context.drawImage(img, sx, sy, swidth, sheight,  
x, y, width, height);
```

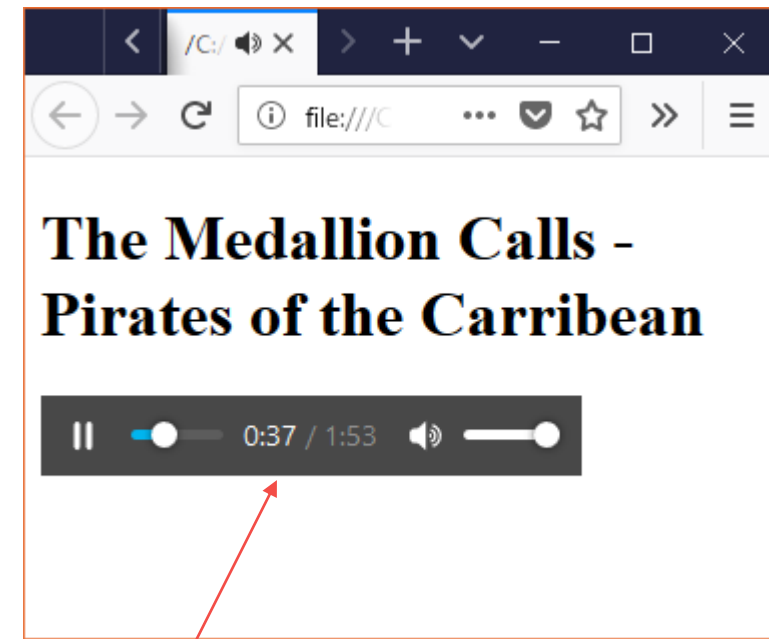
```
<script>  
draw();  
function draw() {  
  var ctx = document.getElementById('Canvas1')  
    .getContext('2d');  
  var img = document.getElementById("photo1");  
  img.onload = function() {  
    ctx.drawImage(img, 0, 0, 164, 164, 10, 10, 164, 164);  
    ctx.drawImage(img, 165, 0, 164, 164, 180, 10, 164, 164);  
    ctx.drawImage(img, 0, 165, 164, 164, 10, 180, 164, 164);  
    ctx.drawImage(img, 165, 165, 164, 164, 180, 180, 164, 164);  
  };  
}  
</script>
```



HTML5 Audio

- The `<audio>` and `<source>` tags specify a standard way to embed audio in a web page.
 - There are 3 supported file formats for the `<audio>` element: MP3, WAV, and OGG.
 - The `<source>` element allows you to specify alternative audio files which the browser may choose from.

Attribute	Description
autoplay	Starts playing as soon as the audio is ready.
controls	Specifies that audio controls should be displayed.
loop	Specifies that the audio will be repeated.
muted	The audio output should be muted



```
<h1>The Medallion Calls - Pirates of the Carribean</h1>
```

```
<audio autoplay controls>
```

```
<source src="Medallion.ogg" type="audio/ogg">  
<source src="Medallion.mp3" type="audio/mpeg">
```

Your browser does not support the audio element.

```
</audio>
```

Fallback
content

HTML5 Video

- Like the audio embedding, the HTML5 <video> and <source> tags provide a way to embed a video in a web page.
- Attributes:
 - autoplay, controls, loop, muted
 - poster: specifies the URL of an image, which will be displayed before the video is played.
 - preload: this attribute is for buffering large files. It can take one of 3 values:
 - "none" does not buffer the file.
 - "auto" buffers the media file.
 - "metadata" buffers only the metadata for the file.
 - width and height: It is a good idea to always include width and height attributes.
 - If the aspect ratio is not maintained by the sizes you set, the video will grow to fill the space horizontally, and the unfilled space will just be given a solid background color by default.

Example

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<video width="640" height="480"
```

```
poster="Jasper.jpg" controls>
```

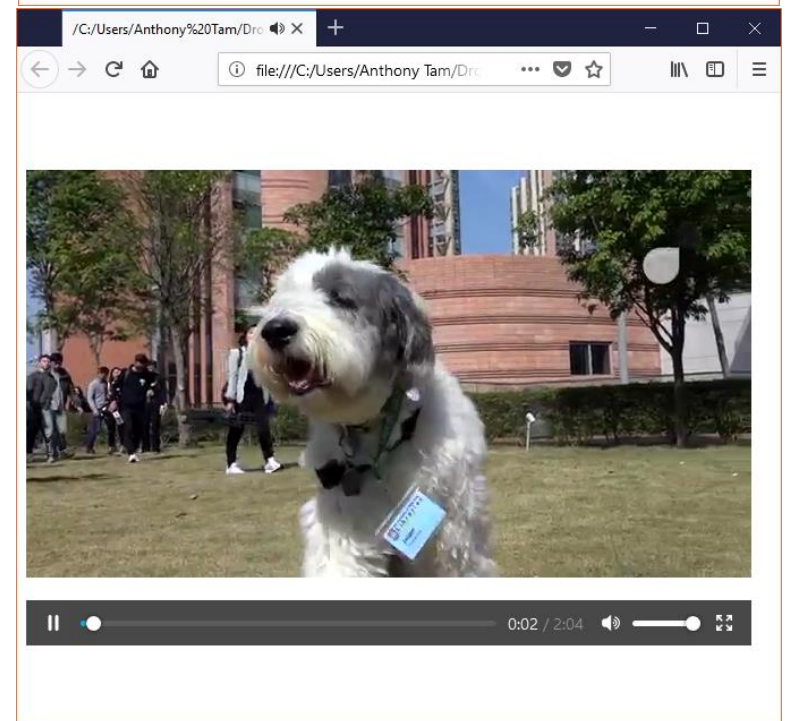
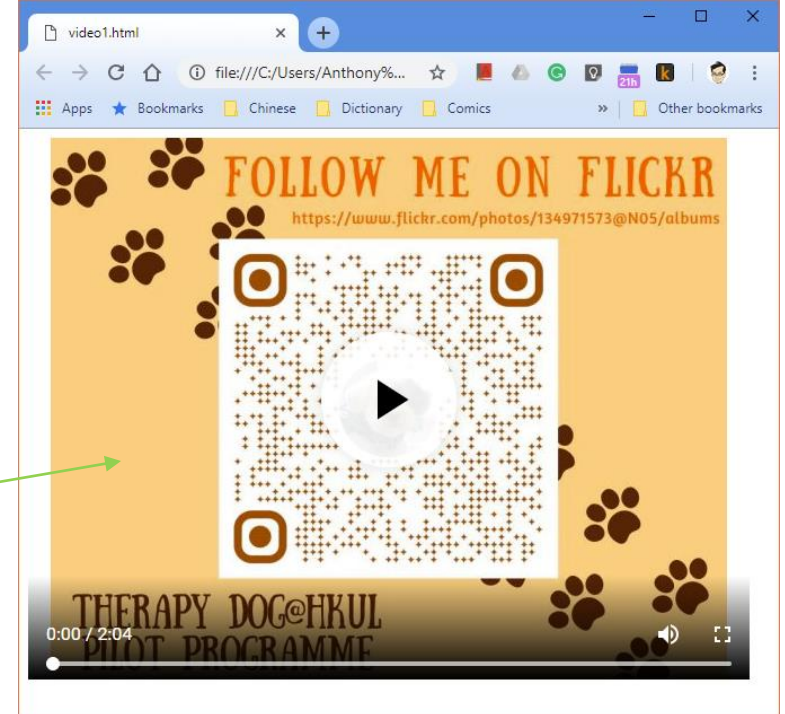
```
<source src="Jasper.mp4" type="video/mp4">
```

Your browser does not support the video tag.

```
</video>
```

```
</body>
```

```
</html>
```



HTML5 Audio and Video

- HTML5 provides DOM methods, properties and events for us to manipulate the audio and video elements using JavaScript.
 - Methods: play(), load(), pause() & addTextTrack()
 - Properties: currentTime, duration, ended, readState, volume, etc.
 - Events: play, pause, ended, canplay, waiting, playing, etc.

HTML5 Drag and Drop

- HTML Drag and Drop interfaces enable applications to use drag and drop features.
 - In HTML5, any element can be draggable.
- **Basic steps:**
 - A *draggable* element is selected with a mouse.
 - Drag the element to a *droppable* element.
 - Release the mouse button to drop the element.
- During the operations, **several events are fired**. Actions are installed in this event handlers to implement this drag and drop feature.

Drag Events



Event	On Event Handler	Description
drag	ondrag	Fired when an element is being dragged.
dragend	ondragend	Fired at the source when a drag operation is being ended.
dragenter	ondragenter	Fired when a dragged element enters a valid drop target.
dragleave	ondragleave	Fired when a dragged element leaves a valid drop target.
dragover	ondragover	Fired when an element is being dragged over a valid drop target.
dragstart	ondragstart	Fired when the user starts dragging an element.
drop	ondrop	Fired when an element is dropped on a valid drop target.

A Draggable Element

- To make a HTML element draggable, do this:

- Set the **draggable** attribute to true.
- Add a listener for the **dragstart** event.
- Set the drag data within the listener defined above.

```
<img draggable="true"  
    ondragstart="drag(event)" >
```

```
function drag(event) {  
    event.dataTransfer.setData("text/plain", event.target.id);  
}
```

- **All drag events** have a property called **dataTransfer** which holds the drag data.
- When the drag begins, store **the type** and a **DOMString** (presenting the data) to the **dataTransfer** property of the drag event object.

A Droppable Element

- By default, the browser **prevents** anything from happening when **dropping** something onto the HTML element.
- To make the HTML element as a droppable zone, do this:
 - *Set the element's **ondragover** and **ondrop** attributes point to the corresponding event handlers.*
- The dragover event handler has to **override the non-droppable** behavior by calling the preventDefault() method.
- The drop event handler has to
 - **Override the default action** on the dropped object.
 - Get the dragged element.
 - **Append** the element into the drop element.

```
function allowDrop(event) {  
    event.preventDefault();  
}
```

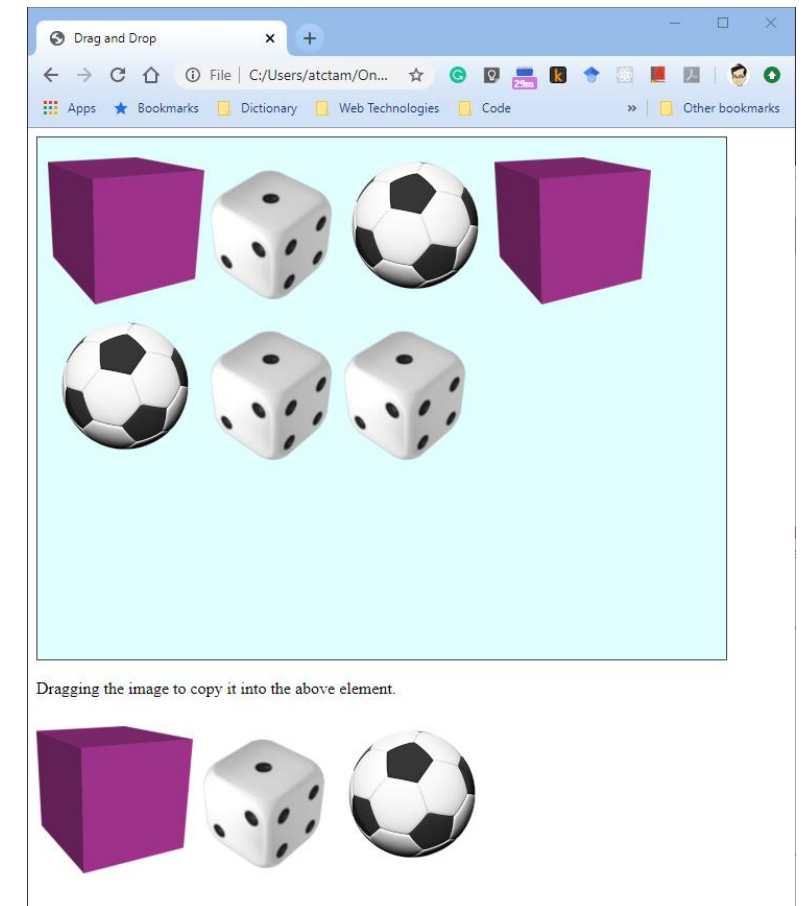
```
function drop(event) {  
    event.preventDefault();  
    var data = event.dataTransfer.getData("text/plain");  
    event.target.appendChild(document.getElementById(data));  
}
```

Example

```
<div id='dest' ondrop='drop(event)' ondragover='allow(event)'></div><br>  
Dragging the image to copy it into the above element.  
br><br>
```

```
<img id='source1' src='cube.png' draggable='true' ondragstart='drag(event)'>  
<img id='source2' src='Dice2.png' draggable='true' ondragstart='drag(event)'>  
<img id='source3' src='ball.png' draggable='true' ondragstart='drag(event)'>
```

```
<script>  
function allow(event)  
{  
    event.preventDefault();  
}  
  
function drag(event)  
{  
    event.dataTransfer.setData('image/png', event.target.id);  
}  
  
function drop(event)  
{  
    event.preventDefault();  
    var data=event.dataTransfer.getData('image/png')  
    event.target.appendChild(document.getElementById(data).cloneNode(true));  
}  
</script>
```



HTML5 Local Storage

- Web storage provides a way for your web applications to store data locally within the user's browser.
- Prior to HTML5, Web applications use cookies to keep state information.
 - Cookies are included with every HTTP request.
 - Cookies are limited to about **4 KB** of data.
- With web storage, storage limit is larger than that of cookies with up to **5 MB** of data per domain and information is *never transferred to the server*.

Types of Web Storage

- Two main web storage types:
 - Local storage
 - The data in local storage **would persist** even when the browser is closed and reopened.
 - **Each site** has its own separate storage area. The data is available to **all scripts loaded from pages** from the same site.
 - Examples: a page visit count, user-authored documents, etc.
 - Session storage
 - Only stores data for one session; will be deleted from the browser once the browser/tab is closed.
 - Stored data will be accessible to **any page from** the same site opened in that window/tab.

Usage (for both localStorage and sessionStorage)

- Check Storage support in browser:

```
if (typeof(Storage) !== "undefined") {  
    // Code for localStorage/sessionStorage.  
}
```

- Store, retrieve & remove a localStorage:

```
// Store  
localStorage.setItem("lastname", "Smith");  
localStorage.firstname = "John";  
// Retrieve  
localStorage.getItem("lastname");  
localStorage.firstname  
// Remove  
localStorage.removeItem("lastname");
```

- To clear the all items in localStorage:

```
localStorage.clear();
```

- To lookup a key name, use

```
localStorage.key(index);
```

https://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_storage_loop

Example

```
<body>
  <script>
    if( localStorage.hits ) {
      localStorage.hits = Number(localStorage.hits) +1;
    } else {
      localStorage.hits = 1;
    }
    document.write("Total Hits :" + localStorage.hits );
  </script>

  <p>Refresh the page to increase number of hits.</p>
  <p>Close the window and open it again and check the result.</p>

</body>
```

HTML5 Geolocation API

- The HTML Geolocation API is used to *get the geographical position* of a user.
- Since this can compromise privacy, the position is not available unless the user **approves** it.

Method / Property	Description
navigator.geolocation	The geolocation interface implemented by the <u>Navigator object</u>
getCurrentPosition()	Determines the device's current location and gives back a Position object with the data.
watchPosition()	Returns the current position of the user and continues to return updated position as the user moves.
clearWatch()	Removes the particular handler previously installed using watchPosition().

getCurrentPosition()

```
navigator.geolocation.getCurrentPosition(success, error)
```

- **success()** - A **callback** function that takes the return **Position object** as its sole input parameter.
- **error()** - An optional callback function that takes a **PositionError** object as its sole input parameter.
- **Position object**:
 - **coords property** - Returns a **Coordinates object** defining the current location.
 - **coords.latitude** - The latitude as a decimal number.
 - **coords.longitude** - The longitude as a decimal number.
 - **timestamp property** - Returns a **DOMTimeStamp** representing the time at which the location was retrieved.

Working with Google Map

- Google provides the **Maps JavaScript API** for us to customize maps for display on our web pages.
- We can use the information obtained from the Geolocation as an input to the Google Maps.
- To use Google Maps JavaScript API, we must **get an API key** in order to request the services provide by Google.
 - To learn how to use and get the API key, please visit this link:
 - <https://developers.google.com/maps/documentation/javascript/get-api-key>

Load the Google Map

1. Load the Maps JavaScript API to your web page using the `<script>` tag.

```
<script async defer
  src="https://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY&callback=getMap">
</script>
```

2. Define a placeholder in your web page for the map; use CSS to set the size of your map.

```
<div id="map"></div>
```

3. Design the callback function to render the map and display it in the placeholder element.

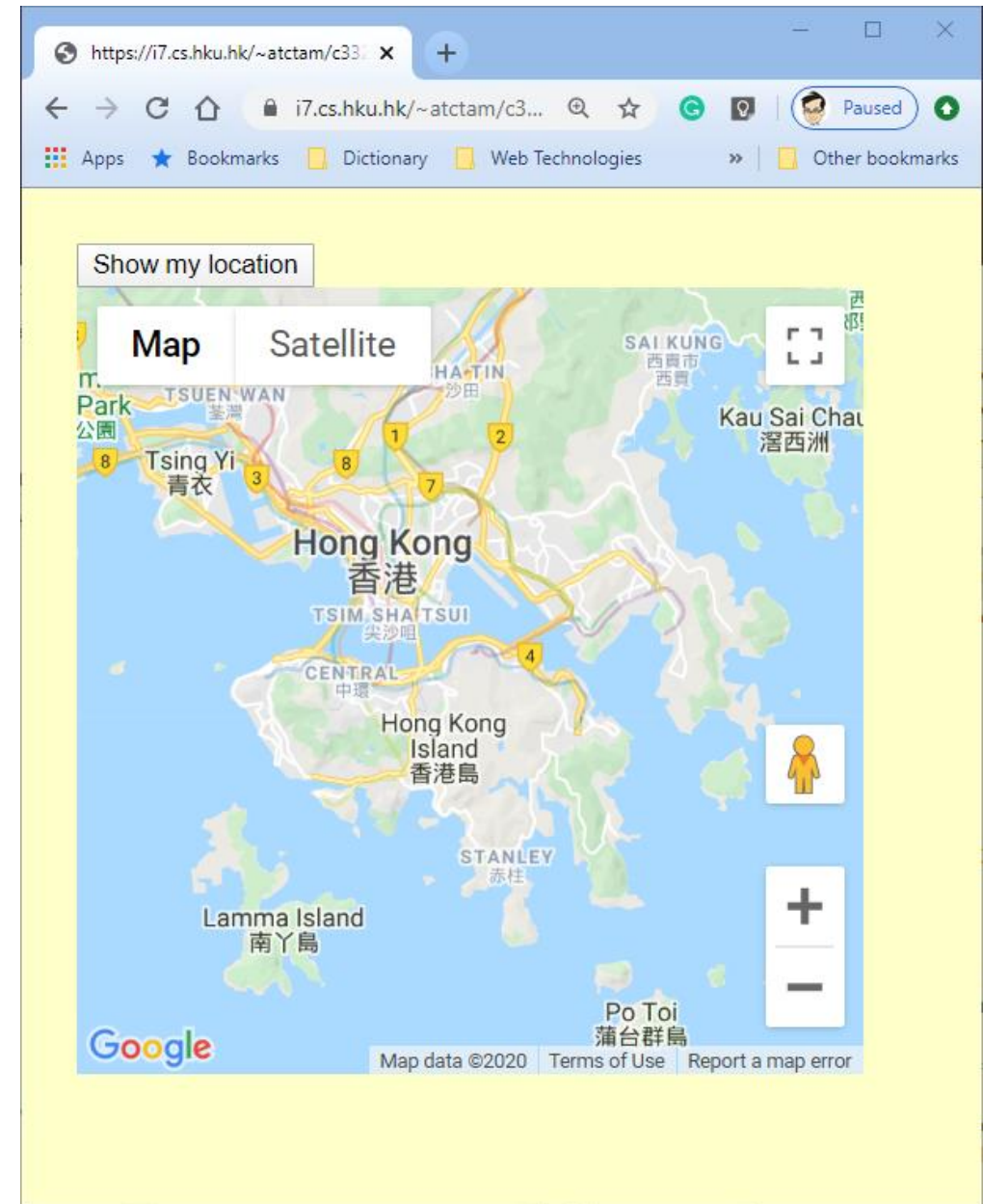
- There are two required options for every map: center and zoom.

```
map = new google.maps.Map(document.getElementById('map'), {
  center: {lat: -34.397, lng: 150.644},
  zoom: 8
});
```

```

<head>
  <meta charset="UTF-8">
  <style>
    body {
      padding: 20px;
      background-color:#ffffc9
    }
    p { margin : 0; }
    #map {
      width:400px;
      height:400px;
    }
  </style>
</head>
<body>
  <p><button onclick="geoFindMe()">Show my
location</button></p>
  <div id="out"></div>
  <div id="map"></div>
  <script>
    : //Show the Geolocation and Maps
  </script>
  <script async defer
src="https://maps.googleapis.com/maps/api/js?key=xxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx&callback=getMap">
  </script>
</body>


```



```
<script>
  var gmap;

  // Callback function
  function getMap() {
    // Initially at some location in Hong Kong
    gmap = new google.maps.Map(document.getElementById("map"), {
      center: {lat: 22.28, lng: 114.2},
      zoom: 11 });
  }

  function geoFindMe() {
    // When the user clicks the button, this function is executed
    // to get the geolocation data and update the map
  }
</script>
```



Create a new map inside the placeholder element; set the center position and the zoom factor of the map.


```
function geoFindMe() {  
  var output = document.getElementById("out");  
  output.innerHTML = "<p>Locating...</p>";
```

```
  if (!navigator.geolocation){  
    output.innerHTML = "<p>Geolocation is not supported by your browser</p>";  
    return;  
  }
```

```
  function success(position) {  
    var pos = {  
      lat: position.coords.latitude,  
      lng: position.coords.longitude  
    };  
    output.innerHTML = '<p>Latitude is ' + pos.lat + '° <br>Longitude is ' + pos.lng + '°</p>';  
    gmap.setZoom(14);  
    gmap.setCenter(pos);  
  }
```

Retrieve
position
coordination

Display position and
update the map

```
  function error() {  
    output.innerHTML = "Unable to retrieve your location";  
  }
```

```
  navigator.geolocation.getCurrentPosition(success, error);
```

Get
geolocation
info

```
}
```

References

- https://www.w3schools.com/graphics/canvas_intro.asp
- https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial
- https://developer.mozilla.org/en-US/docs/Learn/HTML/Multimedia_and_embedding/Video_and_audio_content
- https://developer.mozilla.org/en-US/docs/Web/API/HTML_Drag_and_Drop_API
- https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API/Using_the_Web_Storage_API
- <https://www.tutorialrepublic.com/html-tutorial/html5-web-storage.php>
- https://developer.mozilla.org/en-US/docs/Web/API/Geolocation_API
- <https://developers.google.com/maps/documentation/javascript/tutorial>