****

**Javascript Introduction**

**- Playing with some HTML5 APIs –**

**edX – W3CX(Microsoft)**

1. **Module 3 Outline**

.

### **What you will learn in Module 3**

Note that this module will be more "project oriented", i.e., we will introduce less "fundamental concepts".

* **JavaScript arrays and strings**: in this module, we continue to study fundamental JavaScript concepts, and this time we look a bit deeper into JavaScript arrays and strings.
* **New HTML5 APIs - geolocation and video/audio APIs**: in module 2, we looked at some HTML5 APIs already: the selector and the DOM APIs, respectively for selecting and manipulating HTML elements dynamically. We also had a taste of the HTML5 canvas API for drawing and animating. This time, we will look at the audio and video elements APIs, and the geolocation API.

1. **Arrays (part 2): iterators**
   1. **Arrays**

**[ -- Live coding video: arrays (indexed, length property, sort, splice, push and pop methods) -- ]**

#### ERRATA in the above video:

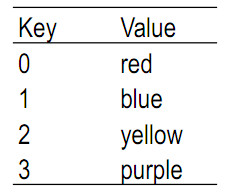
* In the part that explains the splice method for removing element, I say "splice(0, 1) removed the element that was in the middle" is a mistake, it removes the FIRST element, the one that was at index 0. Not the element that was "in the middle", as I say in the video. (the element was in the middle of the array before we sorted it).

### **JavaScript arrays**

In JavaScript, arrays represent a collection of "things", which may be strings, integer values, decimal values, boolean values, or any sort of JavaScript object.

1. > var myarr = ['red', 'blue', 'yellow', 'purple'];
2. undefined
4. > myarr;
5. ["red", "blue", "yellow", "purple"]
7. > myarr[0];
8. "red"
10. > myarr[3];
11. "purple"
12. "purple"

Each element of an array has a key/index and a value. Here are the keys/indexes and values from the above example:



Below is an another example with an array containing three integers. The first element is at index 0, and the last at the index equal to the number of elements-1.

1. > var a = [];
2. > typeof a;
3. "object"
4. > var a = [1,2,3];
5. > a
6. [1, 2, 3]
7. > a[0]
8. 1
9. > a[1]
10. 2

#### JavaScript arrays are objects and have some useful properties and methods

Note that in JavaScript, arrays are "objects" (lines 2-3 in the above example), which means that they have properties and methods. You can access/call them using the "." operator. Here are the most common properties and methods.

1. > var a = [1, 3, 2, 5, 7];
2. undefined
4. >**a.length;** // number of elements
5. 5
7. >**a.sort();** // sorts element in a
8. [1, 2, 3, 5, 7]
10. >**a.splice(2, 1);** // remove 1 element starting from index=2 (3rd element)
11. [3]
13. > a; // the '3' has been removed from the array
14. [1, 2, 5, 7]

By default, the sort() method sorts elements alphabetically if they are strings, or from lowest to highest if they are numeric. If you want to sort objects like {firstName:'michel', lastName:'Buffa', age:51}, you will need to use another method passed as an argument to the sort method, for example to indicate the property you want to use for sorting (i.e., sort by age);

Example with an array of persons (each person is an object):

1. var persons = [
2. {givenName: 'Michel', familyName: 'Buffa', age:51},
3. {givenName: 'Pig', familyName: 'Bodine', age:20},
4. {givenName: 'Pirate', familyName: 'Prentice', age:32}
5. ];
7. function compareByAge(a,b) { // comparison function, a and b are persons
8. if (a.age < b.age)         // compare by age
9. return -1;
10. if (a.age > b.age)
11. return 1;
12. return 0;
13. }
15. **persons.sort(compareByAge); // this will call automatically compareByAge**
16. **// passing all persons from the array, compare**
17. **// them by age and sort the array.**

Explanations:

* Line 17 calls persons.sort(function\_that\_compares\_two\_elements), passing as an unique parameter a function that compares two people's ages. This function must return -1 if the first person is younger than the second person. It must return +1 if the first person is older than second person, and 0 if they are the same age.

We will see more methods in the other subsections of this page.

#### Elements can be of different types in a same array:

1. > var a = [1,2,3];
3. > a[2] = 'three';
4. "three"
6. > a
7. [1, 2, "three"]

#### Adding elements to an array

We can add new elements using a new index, if you want to add a new element at the end, use the push method!

1. > var a = [1,2,"three"];
2. undefined
4. > a[3] = 'four';
5. "four"
7. > a;
8. [1, 2, "three", "four"]
10. > a[a.length] = "five"; // adding at the end
11. [1, 2, "three", "four", "five"]
13. >**a.push("six"); // but usually we prefer using the push method for adding**
14. [1, 2, 3, "four", "five", "six"]  **// a new element at the end**

When using indexes, be careful not to leave "holes" in the array:

1. > a[7] = 'height';
2. "height"
4. > a;
5. [1, 2, 3, "four", "five", "six", **undefined × 1**, "height"]

This array is valid, but having a [6] equal to "undefined" is often prone to errors. Be careful when using absolute indexes for adding elements. We recommend using the push method instead.

#### Removing elements from an array

The recommended method is to use the splice method:

1. array.splice(start)
2. array.splice(start, deleteCount)

* **start:** index at which to start changing the array (with origin 0).
* **deleteCount** (Optional): an integer indicating the number of old array elements to remove.  If deleteCount is greater than the number of elements left in the array starting at start, then all of the elements through the end of the array will be deleted. **If deleteCount is omitted**, deleteCount will be equal to (array.length - start), i.e., **all of the elements beginning with start index on through the end of the array will be deleted**.
* **Return value**: an array containing the deleted elements. If only one element is removed, an array of one element is returned. If no elements are removed, an empty array is returned.

Examples:

1. > a;
2. [1, 2, 3, "four", "five", "six", undefined × 1, "height"]
4. >**a.splice(6, 1); // remove element at the sixth index, the undefined one!**
5. [undefined × 1]
7. > a;
8. [1, 2, 3, "four", "five", "six", "height"] // it's no more here :-)
10. >**a.splice(0, 3); // remove the three first elements**
11. [1, 2, 3]
13. > a;
14. ["four", "five", "six", "height"]
16. >**a.splice(a.length-1); // remove the last element**
17. "height"
19. > a;
20. ["four", "five", "six"]

**Recommended method for removing the last element: the pop method!**

1. **> a**
2. ["four", "five", "six"]
4. >**a.pop(); // remember push/pop = add / remove element at last position!**
5. "six"
7. > a
8. ["four", "five"]

Trap: the delete method is not good for removing an element from an array!

1. > **delete a[1];**
2. true
4. > a;
5. ["four", undefined × 1] **// the element became undefined,**
6. **// but it's still in the array!**

#### Arrays of arrays

It is possible for an array to be an element within an array! This example shows an array made of two arrays of three elements each. It's a 2x3 matrix with two rows and three columns!

1. > var a = [[1,2,3], [4,5,6]]; // a is a matrix: 2 rows, 3 columns.
2. undefined
4. > a[0]; // first row
5. [1, 2, 3]
7. > a[1]; // second row
8. [4, 5, 6]
10. > a[0][0]; // top left element
11. 1
13. > a[0][1]; // second element, first line
14. 2
16. > a[0][2]; // third element, first line
17. 3
19. > a[1][0]; // first element, second line
20. 4
22. > a[1][1]; // second element, second line
23. 5
25. > a[1][2]; // third element, second line
26. 6

It is possible to have different arrays with different lengths and different types of element in an array:

1. > var a = [];
2. undefined
4. > a[0] = [1, 2, 3, 4, 5];
5. [1, 2, 3, 4, 5]
7. > a[1] = ['michel', 'henri', 'francois']
8. ["michel", "henri", "francois"]
10. > a
11. [Array(5), Array(3)]

**Knowledge check 3.2.1 (not graded)**

var a = [1, 2, 3, "four", "five", "six", "seven", "height"]

**How would you remove elements "four" and "five" at once?**

1. splice(3, 2);
2. delete(4, 5);
3. remove(3, 2);
4. splice(2, 4);
   1. **String are arrays of characters**

#### Yes, they do look like arrays!

JavaScript strings are "like" arrays of characters, but they have some limitations, and some dedicated properties and methods:

1. > var s = 'Michel';
2. undefined
4. > s[0];
5. "M"
7. > s[1];
8. "i"
10. > s.length;
11. 6

Indeed, the string s behaves like an array, it has the length property like an array, and we can access individual characters using indexes that go from 0 to length-1, like arrays...

However...

#### But they are not quite the same as arrays!

You cannot add elements to strings using a non-existent index, you cannot use the push/pop methods for adding/removing  characters at the end of the string:

1. **s.push(' Buffa');**
3. **ERROR: VM5748:1 Uncaught TypeError: s.push is not a function**
4. **at <anonymous>:1:3**
5. **(anonymous) @ VM5748:1**
7. s[s.length] = 'B'; // add 'B' at the end?
8. "B"
10. s[s.length] = 'u'; // add 'u' at the end?
11. "u"
13. s[s.length] = 'f'; // add 'f' at the end?
14. "f"
16. **s; // s remained UNCHANGED!**
17. **"Michel"**

* You cannot use push/pop as this raises an error "is not a function" (lines 1-5)
* You can try to put elements out of the range of the string: nothing will happen and the string will remain unchanged (lines 7-17)

You can't even modify a character using an index. Strings are "read only" when using brackets to access individual characters!

1. > var s = 'Michel';
2. undefined
4. > s[0] = "R"; // trying to change the 'M' into an 'R'
5. "R"
7. s; // no luck!
8. "Michel"

You also can't remove characters using the array's splice method:

1. > s.splice(0, 3);
3. ERROR: VM716:1 Uncaught TypeError: s.splice is not a function
4. at <anonymous>:1:3

#### So: how do we add characters to a string, how can we modify a string? How can we delete elements in a string ?

Strings come with a whole set of methods, which we'll come to in module 4 when we talk about JavaScript objects (in the section titled "JavaScript predefined objects"). Without going into detail just yet, here are some examples:

**Adding a string to the beginning of a string using the + operator:**

1. > **var s = 'Michel';**
2. undefined
4. >**s = "Hello " + s;**
5. "Hello Michel"
7. >**s = 'O' + s;** // equivalent to push('0') with arrays
8. "OHello Michel"

**Adding a string to the end of another one with the + operator:**

1. **>s = 'Michel';**
2. "Michel"
4. >**s += ' Buffa';**
5. "Michel Buffa"
7. > s;
8. "Michel Buffa"

Adding a string at the end of another one using the concat method:

1. > var s1 = 'Michel';
2. undefined
4. > var s2 = 'Buffa';
5. undefined
7. **> var s3 = s1 + " " + s2; // + can be used to concat more than 2 strings**
8. undefined
10. > s3;
11. "Michel Buffa"
13. **> var s4 = s1.concat(s2);**
14. undefined
16. > s4;
17. "MichelBuffa"
19. **> var s5 = s2.concat(s1);**
20. undefined
22. s5;
23. "BuffaMichel"

**Removing chars from a string using the substring method:**

Removing the last char (equivalent to the pop method from arrays):

1. > var s = 'Michel';
2. undefined
4. >**s = s.substring(0, s.length-1);**
5. "Miche"

Removing a certain number of chars starting from a string, starting at a given index  :

1. var s = 'Michel';
3. **function removeChars(s, startIndex, numberOfCharsToRemove) {**
4. **return s.substring(0, startIndex) +**
5. **s.substring(startIndex + numberOfCharsToRemove);**
6. **}**
8. // remove 3 consecutive chars from s, starting at index = 1
9. s = removeChars(s, 1, 3);
11. console.log(s); // will display "Mel" in the console

Replacing a char at a given index:

1. function replaceAt(s, index, character) {
2. return s.substr(0, index) + character + s.substr(index+character.length);
3. }
5. **var s2 = "JavaScript";**
6. **s2 = replaceAt(s2, 1, "o");**
8. console.log(s2); // will display "JovaScript"
10. **// it also works with a string instead of a simple char**
11. **s2 = replaceAt(s2, 0, "Coca");**
12. **console.log(s2); // Will display "CocaScript"**

**Knowledge check 3.2.2 (not graded)**

**What is the name of the function/method for removing a given number of chars from a string, starting at a given index?**

1. There is no built-in method/function for that, we need to build one.
2. splice, with the same syntax we used with JavaScript arrays
3. substring
   1. **Iterating on array elements**

### **Iterating on array elements**

#### 1 - Iterating using the forEach method

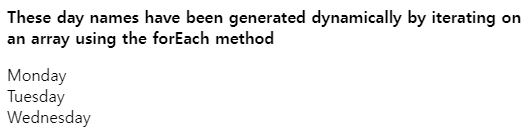
The forEach method takes a single argument that is a function/callback that can have one, two or three parameters:

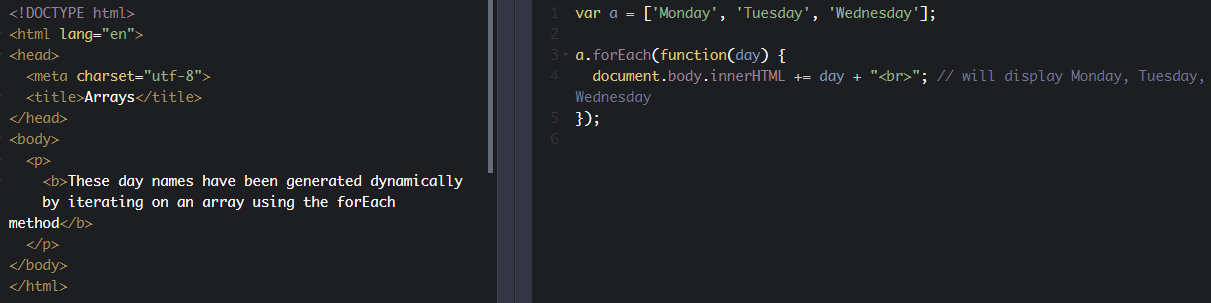
* The first parameter is the current element of the array,
* The second parameter (optional) is the index of the current element in the array,
* The third element is the array itself

**Typical use with only one parameter (the current element):**

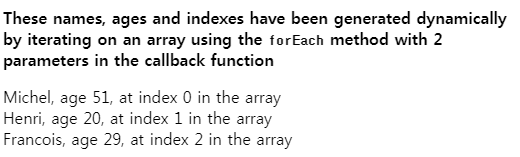
1. **var a = ['Monday', 'Tuesday', 'Wednesday'];**
3. **a.forEach**(function(day) {
4. // day is the current element
5. document.body.innerHTML += day +
6. "<br>"; // will display Monday, Tuesday, Wednesday
7. })

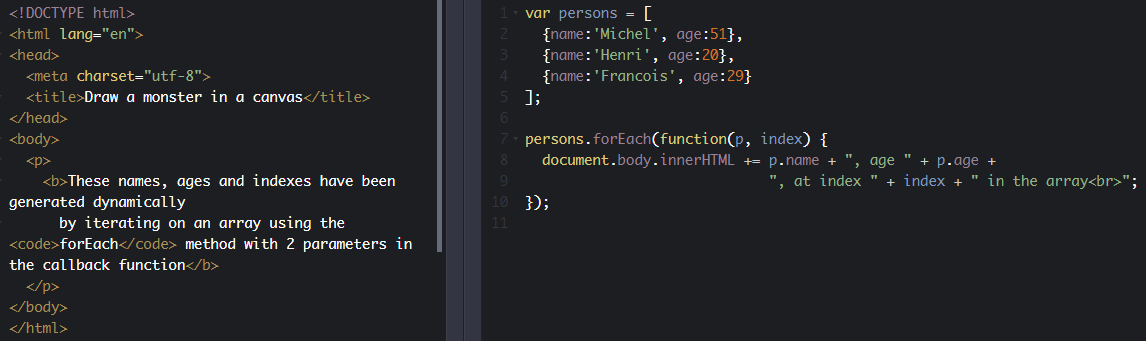
This is the most practical way to iterate on each individual element of a collection (array, string);

****

****

**We iterate on an array of person, and use two parameters in the callback function in order to get the index of the current element**:

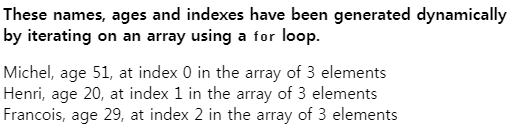
****

****

1. var persons = [
2. {name:'Michel', age:51},
3. {name:'Henri', age:20},
4. {name:'Francois', age:29}
5. ];
7. persons.forEach(function(**p, index**) {
8. document.body.innerHTML += p.name + ", age " + p.age +
9. ", at index " + index + " in the array<br>";
10. });

**Third example using three parameters, the last one being the array itself.**

This can be useful if we need to know the length of the array, or do special things within the array (add/change/move elements during the iteration):

****

****

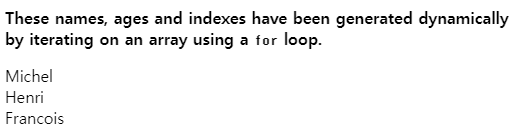
In this example, we used the third parameter (the array) to access its length inside the iteration loop.

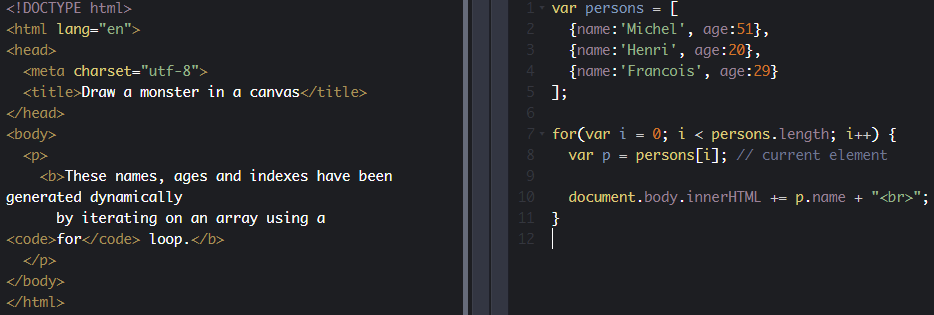
#### 2 - Iterating on an array using regular loop statements

You can use any standard loop statement that we saw during in module 2. The most common way to iterate over an array is to use a for loop from 0 to length-1.

Using this method allows elements to be iterated two by two, or the loop to be broken in the middle using the break instruction, etc.

**Iterating over all elements in an array, using a for loop:**

****

****

**Knowledge check 3.2.3 (not graded)**

**Can you iterate only on elements that have an odd index using the forEach iterator, without adding a test inside the instructions in the body of the loop?**

* No
* Yes
  1. **Discussion topics and projects**

Here is the discussion forum for this part of the course. Please either post your comments/observations/questions or share your creations.

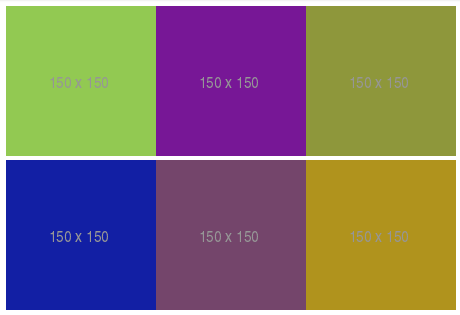
See below for suggested topics of discussion and an optional project.

#### Suggested topics

* Which syntax do you prefer for iterating arrays?
* We decided not to explain all the things you can do with the splice method. It's a very powerful method, but showing all its features could be overwhelming. However, if you're curious, please look on the Web for more details. Do you think we've been right not to tell you all the possibilities of splice?

#### Optional project: an interactive picture album browser

* We will start with an array variable like that:
  1. let myPicturesArray = [
  2. {
  3. "albumId": 1,
  4. "id": 1,
  5. "title": "accusamus beatae ad facilis cum similique qui sunt",
  6. "url": "http://placehold.it/600/92c952",
  7. "thumbnailUrl": "http://placehold.it/150/92c952"
  8. },
  9. {
  10. "albumId": 1,
  11. "id": 2,
  12. "title": "reprehenderit est deserunt velit ipsam",
  13. "url": "http://placehold.it/600/771796",
  14. "thumbnailUrl": "http://placehold.it/150/771796"
  15. },
  16. {
  17. "albumId": 2,
  18. "id": 51,
  19. "title": "non sunt voluptatem placeat consequuntur rem incidunt",
  20. "url": "http://placehold.it/600/8e973b",
  21. "thumbnailUrl": "http://placehold.it/150/8e973b"
  22. },
  23. {
  24. "albumId": 2,
  25. "id": 52,
  26. "title": "eveniet pariatur quia nobis reiciendis laboriosam ea",
  27. "url": "http://placehold.it/600/121fa4",
  28. "thumbnailUrl": "http://placehold.it/150/121fa4"
  29. },
  30. {
  31. "albumId": 3,
  32. "id": 127,
  33. "title": "magnam quia sed aspernatur",
  34. "url": "http://placehold.it/600/74456b",
  35. "thumbnailUrl": "http://placehold.it/150/74456b"
  36. },
  37. {
  38. "albumId": 3,
  39. "id": 128,
  40. "title": "est facere ut nam repellat numquam quia quia eos",
  41. "url": "http://placehold.it/600/b0931d",
  42. "thumbnailUrl": "http://placehold.it/150/b0931d"
  43. }
  44. ];
  45. // Display picures
  46. myPicturesArray.forEach(function(currentImage) {
  47. let image = document.createElement("img");
  48. image.src = currentImage.thumbnailUrl;
  49. image.alt = currentImage.title;
  50. document.body.append(image);
  51. });
* It's an array of pictures, each picture having a URL, a URL for a tiny version of the picture, called a thumbnail, a title, and the name of the album it belongs to, that can be used as a picture description (HTML alt attribute) but also for displaying it next to the picture.
* Here is a small example that iterates on the pictures and create <img> elements on the fly:



* What you will have to do:

1. **Improve the display, by adding margins (CSS), shadows, border, and changing the URLs for real pictures**. Remember that you need to have a smaller versions of the pictures, i.e., thumbnails. You can use existing images (images.google.com is your friend) or images you upload somewhere.
2. **Use JavaScript for adding a click event listener on each image**, then when clicked, you will show a bigger version of each picture. For the moment, just change the value of the src attribute of the clicked image (set it to the URL of the full size image from the array).
3. **Try to make something nicer: reserve a <div> on the right of the document so to display the clicked image with a bigger size**. In that case, you will need to create an image (only once, after the first click), to set it to the size of the div (use the width and height attributes of the img element), and to append it to the div.
4. **Try to add more images, and find a way to display them per album.** Create buttons entitled "album1", "album2", etc., and when clicked, you will only display images from the selected album.
5. **[Advanced] Add an option for deleting a picture.** It should be removed from the document and from the array too...
6. **Feel free to add any interesting feature you think about ;-)**
7. **HTML5 multimedia and JavaScript API**
   1. **Audio and video stream intro**

### **Play audio and video streams**

These examples are adapted from the ones in the W3Cx [HTML5 Coding Essentials and Best Practices](https://www.edx.org/course/html5-coding-essentials-and-best-practices) course, which covers multimedia in depth. The HTML5 advanced course also gives further examples covering topics such as making a video player with chapter navigation, clickable transcript, audio EQ, etc.

#### The <video> element

##### INTRODUCTION

The <video> element of HTML5 is one of the two "Flash killers" (the other being the <canvas> element). It was designed to replace horrible things like embedded Flash objects that we used to encounter not so long ago.

The new way of doing things is a lot better... The source code of this example shows the typical usage of the <video> element:

1. <video width="320" height="240" controls="controls">
2. <source src="movie.mp4" type="video/mp4" />
3. <source src="movie.ogg" type="video/ogg" />
4. Your browser does not support the <video> element.
5. </video>

Please note that:

* The controls attribute indicates that a control panel with play/stop/volume/progress widgets should be displayed;
* Usually the browser will use the first format it recognizes  (in this case, the browser checks whether mp4 is supported, and if not, it will check for the ogg format, and so on). Some browsers may use a different heuristic and choose a "preferred" format.
* The <video> element is a DOM member, so  CSS styling can be applied, as well as manipulation using the DOM API.

You will learn more about the different attributes of the <video> element later on in the course.

##### RESTRICTION: You cannot embed a YouTube or a Daily Motion video using the <video> element

Help! <video src="my youtube video URL"></video> does not work!

**BEWARE**: you cannot directly embed videos from most of the popular social Web sites such as YouTube, Dailymotion, Vimeo, etc. For commercial reasons, and because advertising is automatically  added to the videos, these Web sites do not allow "regular" embedding of their videos.

While they use HTML5 to render their videos, these hosting sites (YouTube, etc.) use rather complex techniques in order to prevent you from using them with the <video> element. Instead, you often need to embed an <iframe> that will render the HTML5 videos in your Web site, and of course, the advertising that comes along with them.

Usually you have an "embed" button close to the videos that prompts you with some HTML code that you can copy and paste for embedding.

**An example using YouTube:**

Here is the HTML code you need to copy and paste in order to embed a video (in this case, a tutorial that tells you how to embed a YouTube video):

1. <iframe width="560" height="315" src="https://www.youtube.com/embed/ZH1XOsv8Oyo" frameborder="0" allowfullscreen></iframe>

Below is the YouTube video embedded in this page using the above code: it's HTML5 but it's not a <video> element directly inserted in the HTML of this page - it's an <iframe>.

#### The <audio> element

##### INTRODUCTION

HTML5 audio is composed of several layers:

* [**The <audio> element**](http://www.w3.org/wiki/HTML/Elements/audio)is useful for embedding an audio player into a Web page. It is dedicated for streamed audio. It is very similar to the <video> element, both in its use and in its API.
* [**The "Web Audio API"**](http://www.w3.org/TR/webaudio/) is designed for musical applications and for adding sound effects to games. This pure JavaScript API supports manipulation of sound samples (loops, etc.), music synthesis and sound generation (oscillators, etc.). It also comes with a set of predefined sound processing modules (reverb, delay, etc.).

This course will focus on the <audio> element. Check for the advanced HTML5 course ([HTML5 Apps and Games](https://www.edx.org/course/html5-apps-games-advanced-techniques-w3cx-html5-2x)), available on W3Cx, which covers [the Web Audio API](http://www.w3.org/TR/webaudio/) and other advanced parts of HTML5.

The attributes, event set and JavaScript API of the <audio> element are just a "reduced" version of the ones from the <video> element, and here we will only address their differences and peculiarities.

#### The <audio> element - basic usage

##### The most simple basic example

Press play to stream the neigh of a horse:

As you can see, the code is very similar to the basic <video> element usage.

1. <!DOCTYPE html>
2. <html lang="en">
3. <head>
4. <meta charset="utf-8">
5. <title>Draw a monster in a canvas</title>
6. </head>
7. <body>
8. <audio controls="controls" crossorigin="anonymous">
9. <source src="https://mainline.i3s.unice.fr/mooc/week2p1/horse.ogg"
10. type="audio/ogg" />
11. <source src="https://mainline.i3s.unice.fr/mooc/week2p1/horse.mp3"
12. type="audio/mp3" />
13. Your browser does not support the audio element.
14. Download the audio/video in
15. <a href=”https://mainline.i3s.unice.fr/mooc/week2p1/horse.ogg”>OGG</a>
16. or<a href=”https://mainline.i3s.unice.fr/mooc/week2p1/horse.mp3”>MP3</a>
17. format.
18. </audio>
19. </body>
20. </html>

In this example, just as for the <video> element, we used the controls attribute in order to render the play/stop, time, volume and progress widgets.

Notice the other similarities: between the <audio>...</audio> tags, we added a text message that is displayed if the Web browser doesn't support the <audio> element, and we used several <source>...</source> elements that link to different audio formats for the same file. The browser will use the first format it recognizes.

Lines 8-12:  we suggest downloading the audio files if the browser does not support the <audio> element. This is also a best practice!

**Knowledge check 3.3.1 (not graded)**

**The <video> element is like any other HTML element: I can style it using CSS and interact programmatically with it using the JavaScript DOM API.**

* True
* False
  1. **Audio and video player JavaScript API**

**[ -- Live coding video: the video element JavaScript API -- ]**

#### Control <audio> and <video> elements from JavaScript

The <video> element has methods, properties/attributes and events that can be manipulated with JavaScript. Using the DOM API it's possible to manipulate an audio or video element as a JavaScript object that has:

* **Methods** for controlling its behavior, such as play(), pause(), etc.;
* **Properties** (duration, current position, etc.), either in read/write mode (such as volume), or in read-only mode (such as encoding, duration, etc.);
* **Events** generated during the life cycle of the element that can be processed using JavaScript callbacks. It is also possible to send events to control the video player.

Like any HTML element, the <video> element can be manipulated/created using the DOM JavaScript API. Here is an example of programmatically creating a <video> element:

1. var video = document.createElement('video');
2. video.src = 'video.mp4';
3. video.controls = true;
4. document.body.appendChild(video);

This will create a complete video player for the file "video.mp4", with control buttons, and will add it to the <body> element of the page.

#### JavaScript API of the <audio> and <video> elements

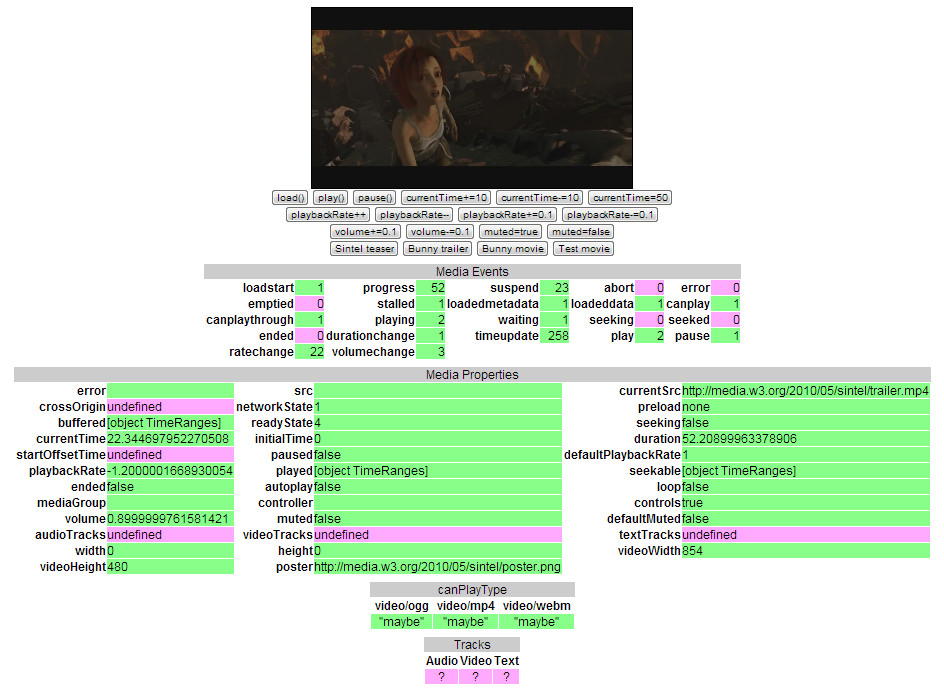
##### Methods, properties and events

The JavaScript API gives you powerful tools to manipulate the <video> element, as the video object provides many properties, methods and events.

The complete list of events can be found at [the W3C specification page](http://www.w3.org/TR/html5/embedded-content-0.html#event-definitions), and numerous examples of each event can be found on many Web sites [such as this one](http://www.htmlgoodies.com/html5/tutorials/HTML5-Development-Class-Media-Events-3883356.htm#fbid=rRDjiexm8vR).

The complete list of properties can be found at [the W3C HTML5 Video Events and API page](http://www.w3.org/2010/05/video/mediaevents.html). This page is interesting for Web developers because it shows an interactive view of the different values and events changing over time while the video is playing within the page.

**Click the picture to see it running online (or try the**[**direct link**](http://www.w3.org/2010/05/video/mediaevents.html)**), then play with the different buttons and look at the table of events and properties that will change in real time. The displayed names show the properties, events, and methods from the API.**

[](http://www.w3.org/2010/05/video/mediaevents.html)

#### Here is a table that shows ****the most interesting methods, properties and events**** provided by the <video> element API

We provide this as a quick reminder - keep in mind that the complete list is much longer!

|  |  |  |
| --- | --- | --- |
| **Methods** | **Properties** | **Events** |
| **play()** | **currentSrc** | **play** |
| **pause()** | **currentTime** | **pause** |
| **load()** | **startTime (readonly)** | **progress** |
| **canPlayType()** | **videoWidth** | **error** |
|  | **videoHeight** | **timeupdate** |
|  | **duration (readonly)** | **ended** |
|  | **ended (readonly)** | **abort** |
|  | **error** | **empty** |
|  | **paused (readonly)** | **emptied** |
|  | **muted** | **waiting** |
|  | **seeking** | **loadedmetadata** |
|  | **volume** |  |
|  | **height** |  |
|  | **width** |  |
|  | **seekable (readonly)** |  |
|  | **played (readonly)** |  |

Now let's take a look at a set of examples demonstrating how to use the most important of these properties, methods, and events...

**Knowledge check 3.3.2 (not graded)**

**The W3C specification about the JavaScript API associated to <audio> and <video> elements, proposes an interactive demonstration of the different properties/methods/events; it's a must see for all Web developers interested in multimedia. Try it and guess what properties indicate the length of the video in seconds and the name of a valid event that is sent while the video is being played...**

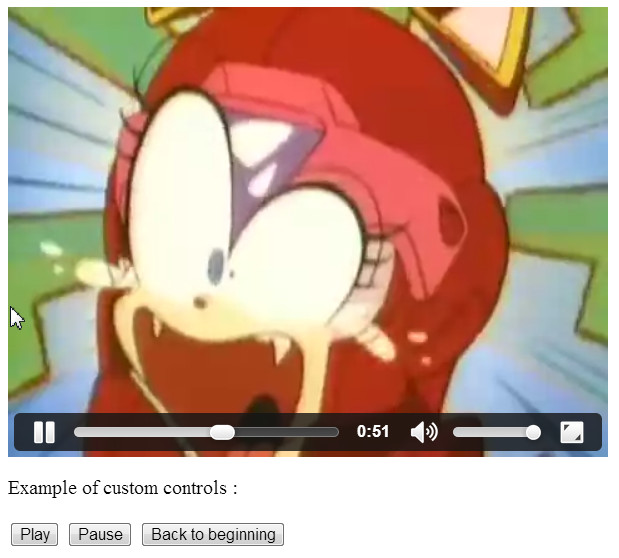
* duration and timeupdate
* currentTime and play
  1. **Examples using the JavaScript API**

### **Examples using the JavaScript API**

The JavaScript API is useful for implementing playlists, making custom user interfaces and many other interesting things. The "enhanced HTML5 multimedia players" presented later on in the course rely heavily on this API.

#### EXAMPLE 1 - HOW TO USE EXTERNAL BUTTONS TO CONTROL A PLAYER'S BEHAVIOR

This example shows the first steps towards writing a custom video player. It shows basic usage of the JavaScript API for adding custom buttons to play/pause the video or to go back to the beginning by setting the currentTime property to zero.



Source code extract:

1. <video id="vid" controls>
2. <source src=http://html5doctor.com/demos/video-canvas-magic/video.webm
3. type=video/webm>
4. ...
5. </video>
6. <p>Example of custom controls:</p>
7. <button onclick="playVideo();" style="cursor: pointer;">Play</button>
9. <button onclick="pauseVideo();" style="cursor: pointer;">Pause</button>
11. <button onclick="rewindVideo();" style="cursor: pointer;">
12. Back to beginning</button>
13. <script>
14. var vid = document.querySelector("#vid");
15. function playVideo() {
16. vid.play();
17. }
18. function pauseVideo() {
19. vid.pause();
20. }
21. function rewindVideo() {
22. vid.currentTime = 0;
23. }
24. </script>

* Lines 7, 9 and 11: we add a click listener to each button, in order to call a JavaScript function when the button is clicked.
* Line 14: using the DOM API we get the JavaScript object that corresponds to the video element we inserted in the HTML document. This line is outside a function, it will be executed when the page loads.
* Lines 17 and 20: we call methods from the API for playing/pausing the video.
* Line 24: we modify the currentTime property in order to rewind the video. Note that vid.load() also rewinds the video, shows the poster image again, but also pauses the video. By using currentTime=0 the playback does not stop.

#### EXAMPLE 2 - HOW TO DETECT THE END OF A VIDEO AND START ANOTHER ONE

This example listens for the ended event, and calls a callback function when the video has finished.

1. <video src="video.ogv" id="myVideo">
2. video not supported
3. </video>
4. <script type='text/javascript'>
5. var vid = document.querySelector('#myVideo');
6. vid.addEventListener('ended', playNextVideo, false);
7. function playNextVideo(e) {
8. // Whatever you want to do after the event (play another video,
9. // for example), change the src attribute, of the video element, etc.
10. }
11. </script>

#### ****EXAMPLE 3, APPLICATION OF THE ABOVE TECHNIQUE - HOW TO MANAGE PLAYLISTS****

This example detects the end of a video then loads the next video, changes the src attribute of the video element and plays the video (see the online [example](https://jsbin.com/ridujix/1/edit?html,output)).

To try this example: use the progress cursor to go near the end of the first video that is being played and see how it continues with the next video.

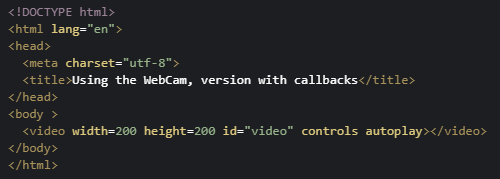
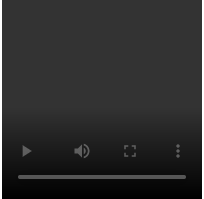
1. <!doctype html>
2. <html lang="en">
3. <head>
4. <title>Sequential Movies</title>
5. <script>
6. var myVideo;
7. var currentVideo = 0;
8. var sources = [
9. "http://html5doctor.com/demos/video-canvas-magic/video.mp4",
10. "http://www.archive.org/download/AnimatedMechanicalArtPiecesAtMit/P1120973\_512kb.mp4"
11. ];
12. // Set the src of the video to the next URL in the playlist
13. // If at the end, we start again from beginning (the modulo
14. // source.length does this)
15. function loadNextVideo() {
16. myVideo.src = sources[currentVideo % sources.length]
17. myVideo.load();
18. currentVideo++;
19. }
20. // callback that loads and plays the next video
21. function loadAndplayNextVideo() {
22. console.log("playing " + sources[currentVideo % sources.length])
23. loadNextVideo();
24. myVideo.play();
25. }
26. // Called when the page is loaded
27. function init(){
28. // get the video element using the DOM api
29. myVideo = document.querySelector("#myVideo");
30. // Define a callback function called each time a video ends
31. myVideo.addEventListener('ended', loadAndplayNextVideo, false);
32. // Load the first video when the page is loaded.
33. loadNextVideo();
34. }
35. </script>
36. </head>
37. <body onload="init()">
38. <video id="myVideo" controls></video>
39. </body>
40. </html>

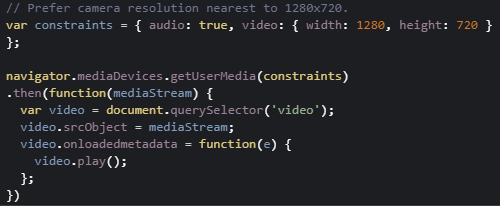
* Line 8: the JavaScript array that contains the URLs of the videos in the playlist. In this example, we've only got two of them, but if the array is larger the example will still work.
* Line 42: When the page is loaded, an init() function is called.
* Lines 32 - 38: we used the DOM to get the JavaScript object corresponding to the video element, then defined a listener for the ended event. Each time a video ends, the loadAndPlayNextVideo() callback will be called. As the video element has no src attribute by default, we also preload the first video (call to loadNextVideo() at line 38).
* Lines 16 - 20: the loadNextVideo() function uses a variable called currentVideo that corresponds to the index of the current video. By setting myVideo.src = sources [currentVideo % sources.length], we set the src of the video element to sources[0], then to sources[1], and as we increment the currentVideo index each time (line 19). If it becomes greater than 1, the modulo (the "%" symbol is the modulo in JavaScript) will make it "loop" between 0 and the number of videos in the playlist. In other words, when the last video ends, it starts back at the first one.
  1. **Using the Webcam**

It's very easy to use the getUserMedia API for accessing the WebCam.

Here is a version that should work on any recent browser except Apple Safari (which still does not support this API). Note that for security reasons you must host your HTML/CSS/JS page on an HTTPS server for getUserMedia to work.

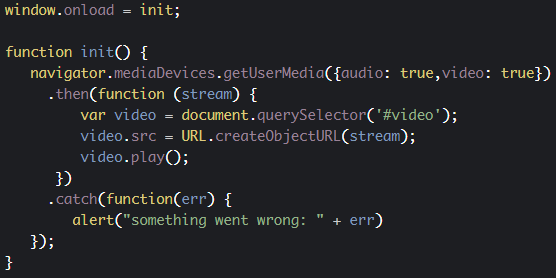
**First version that uses callbacks (success/error, see the JS code):**

** **

****

**Second version that uses a new JavaScript syntax called "promises"**:

This is another way of saying, "Please, browser, try to give me access to the WebCam, **THEN** when the Webcam is ready, please tell me so that I can display its stream in a <video> element".

****

**Knowledge check 3.3.3 (not graded)**

**What is getUserMedia?**

1. A JavaScript API that can be used to redirect the webcam video stream to a video element
2. An API which only works with WebRTC for audio conferencing
3. An upcoming API that is not available yet on browsers, but can be emulated by the video element
   1. **Extended examples**

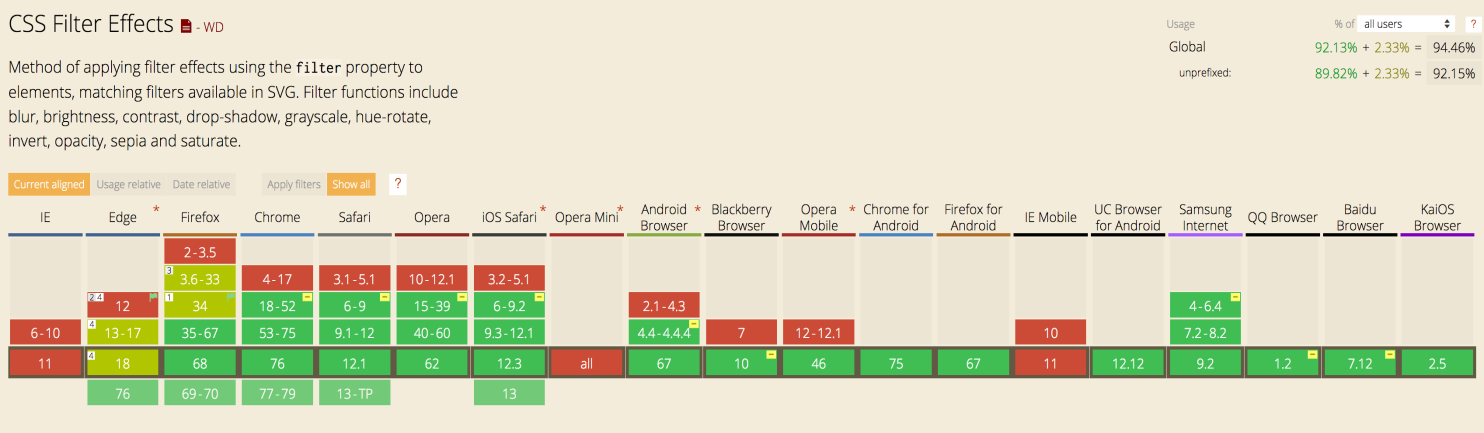
In this section, we provide some extended examples that use more JavaScript and more complex CSS manipulation.

#### Example 1: applying CSS3 filters to a video in real time

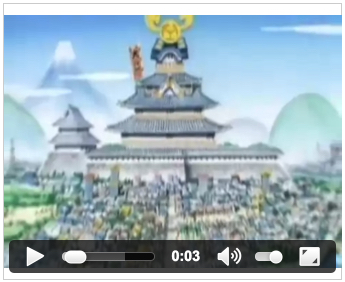
Play the video and then click on the video while it's playing. This will change in real-time the CSS class of the video element. Each class uses the filter property with different values.

Note that CSS filters are not yet 100% supported by the major browsers. You still need to use prefixed versions of the CSS properties, as shown below (this table is taken from [caniuse.com](http://caniuse.com/)).

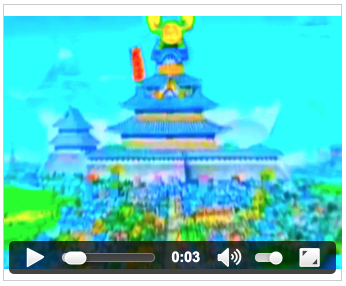
CSS filter support (green squares with a small yellow part in the top right corner) means that a prefix is needed, like -webkit-filter, or -moz-filter or -o-filter) - see August 2019 snapshot below:

[](https://caniuse.com/#feat=css-filters)

Below, you will find images obtained with different filter values:

Use <video class="blur"> for example, to obtain a blurry video. This [complete example](http://jsbin.com/duluyej/1/edit?html,output) changes the CSS class associated to the video element, on the fly in a mouseclick listener callback.

Here, we define the CSS classes used in the example:

1. .blur {
2. filter: blur(3px);
3. }
4. .brightness {
5. filter: brightness(5);
6. }
7. .contrast {
8. filter: contrast(8);
9. }
10. .hue-rotate {
11. filter: hue-rotate(90deg);
12. }
13. .hue-rotate2 {
14. filter: hue-rotate(180deg);
15. }
16. .hue-rotate3 {
17. filter: hue-rotate(270deg);
18. }
19. .saturate {
20. filter: saturate(10);
21. }
22. .grayscale {
23. filter: grayscale(1);
24. }
25. .sepia {
26. filter: sepia(1);
27. }
28. .invert {
29. filter: invert(1)
30. }

This extract from the source code explains how to set a mouseclick listener and how to change the value of a CSS class attribute on the fly:

1. <video id="output" controls autoplay>
2. <source src=http://html5doctor.com/demos/video-canvas-magic/video.webm
3. type=video/webm>
4. <source src=http://html5doctor.com/demos/video-canvas-magic/video.ogg
5. type=video/ogg>
6. <source src=http://html5doctor.com/demos/video-canvas-magic/video.mp4
7. type=video/mp4>
8. </video>
9. <script>
10. var output = document.getElementById('output');
11. var idx = 0;
12. var filters = [
13. 'grayscale',
14. 'sepia',
15. 'blur',
16. 'brightness',
17. 'contrast',
18. 'hue-rotate', 'hue-rotate2', 'hue-rotate3',
19. 'saturate',
20. 'invert'];
21. function changeFilter(e) {
22. var el = e.target;
23. var effect = filters[idx++ % filters.length];
24. el.classname = effect;
25. // Do not propagate the event, prevent default behavior.
26. // By default, a click on a video element pauses/unpauses the video
27. // By stopping the propagation and canceling the default behavior,
28. // we stop the pause/unpause behavior when the video is clicked.
29. // Now a click just changes the CSS filter we apply on the video.
30. e.stopPropagation();
31. e.preventDefault();
32. }
33. output.addEventListener('click', changeFilter, false);
34. </script>
35. <style>
36. #output {
37. width: 307px;
38. height: 250px;
39. background: rgba(255,255,255,0.5);
40. border: 1px solid #ccc;
41. }
42. .blur {
43. filter: blur(3px);
44. }
45. .brightness {
46. filter: brightness(5);
47. }
48. ...
49. </style>

#### Example 2: how to track all possible events and manipulate many properties

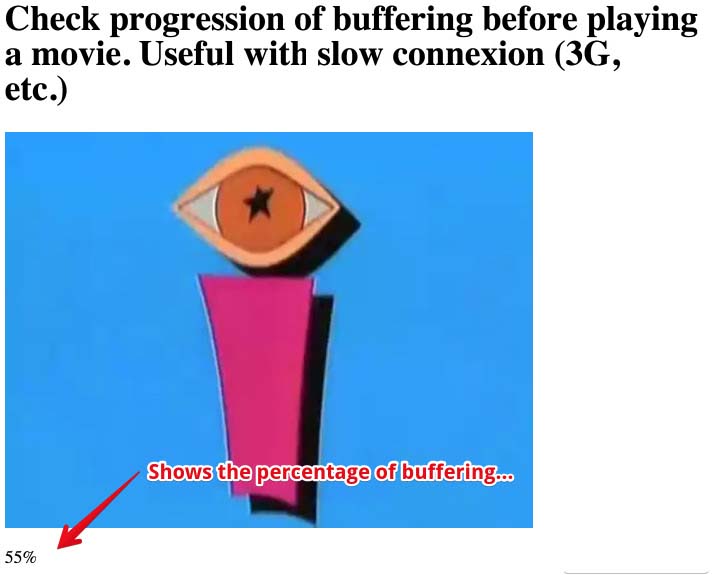
This example also shows how to handle failures..



Here is an example of a piece of code for handling errors during video playback:

1. ...
2. vid.addEventListener('error', function(evt) {
3. logEvent(evt,'red');
4. }, false);
5. ...
6. function logEvent(evt, color) {
7. switch (evt.type) {
8. ...
9. case 'error':
10. var error = document.querySelector('video').error;
11. switch (error.code) {
12. case error.MEDIA\_ERR\_ABORTED:
13. note.innerHTML = "fetching aborted at the user's request";
14. break;
15. case error.MEDIA\_ERR\_NETWORK:
16. note.innerHTML = "a network error caused the browser to stop fetching the media";
17. break;
18. case error.MEDIA\_ERR\_DECODE:
19. note.innerHTML = "an error occurred while decoding the media";
20. break;
21. case error.MEDIA\_ERR\_SRC\_NOT\_SUPPORTED:
22. note.innerHTML = "the media indicated by the src
23. attribute was not suitable";
24. break;
25. default:
26. note.innerHTML = "an error occurred";
27. break;
28. }
29. break;
30. }
31. ...
32. }

#### Example 3: how to display a percentage of buffering when using a slow connection



Note that on mobile phones, the video does not start until the user presses the play control or clicks on the video picture. Using the "canplaythrough" event is a trick to call a function that starts the video player as soon as the page is loaded on desktop. This event is not supported by mobile devices, so if you try this example on a mobile, the video will not start automatically.

As [the Apple Developer](http://developer.apple.com/) Web site explains:  "The buffered property is a TimeRanges object: an array of start and stop times, not a single value. Consider what happens if the person watching the media uses the time scrubber to jump forward to a point in the movie that hasn’t loaded yet—the movie stops loading and jumps forward to the new point in time, then starts buffering again from there. So the buffered property can contain an array of discontinuous ranges. The example simply seeks the end of the array and reads the last value, so it actually shows the percentage into the movie duration for which there is data. "

1. <!DOCRYPE html>
2. <html lang="en">
3. <head>
4. <title>JavaScript Progress Monitor</title>
5. <script>
6. function getPercentProg() {
7. var myVideo = document.getElementsByTagName('video')[0];
8. var endBuf = myVideo.buffered.end(0);
9. var soFar = parseInt(((endBuf / myVideo.duration) \* 100));
10. document.getElementById("loadStatus").innerHTML = soFar + '%';
11. }
12. // Will be called as soon as the page is ready on desktop computer,
13. // Only when a user clicks on play control or image on mobile
14. function myAutoPlay() {
15. var myVideo = document.getElementsByTagName('video')[0];
16. myVideo.play();
17. }
18. function addMyListeners(){
19. var myVideo = document.getElementsByTagName('video')[0];
20. myVideo.addEventListener('progress', getPercentProg, false);
21. // Calls autoplay only if the device is adapted
22. myVideo.addEventListener('canplaythrough', myAutoPlay, false);
23. }
24. </script>
25. </head>
26. <body onload="addMyListeners()">
27. <h1>Check progression of buffering before playing a movie. Useful withy
28. slow connexion (3G, etc.)</h1>
29. <div>
30. <video controls>
31. <source src=http://html5doctor.com/demos/video-canvas-magic/video.webm
32. type=video/webm>
33. <source src=http://html5doctor.com/demos/video-canvas-magic/video.ogg
34. type=video/ogg>
35. <source src=http://html5doctor.com/demos/video-canvas-magic/video.mp4
36. type=video/mp4>
37. </video>
38. <p id="loadStatus">Buffering...</p>
39. </div>
40. </body>
41. </html>

* 1. **Discussion topics and projects**

Here is the discussion forum for this part of the course. Please either post your comments/observations/questions or share your creations.

See below for suggested topics of discussion and optional projects.

#### Suggested topics

* This might be useful: free videos at <https://download.blender.org/peach/bigbuckbunny_movies/>
* Hosting videos is complicated when you want to use them with CodePen or JsBin, do you have some tips to share with others? Here at W3C we run our own private HTTP server...We are speaking about files that can be used with the <video> element directly, not on YouTube, DailyMotion, etc.
* What tool do you use for encoding your audio and video files?

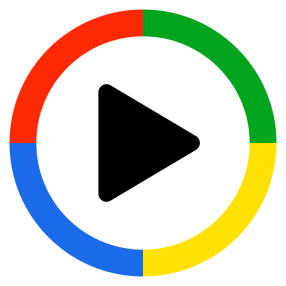
#### Optional projects:

#### 1. A custom video player

Here are a few ideas to play with the material learned in this section. Your classmates and the team who prepared the course will be happy to look at them and give feedback. Please post URLs of your work in this discussion forum. These projects are optional, meaning that they won't be graded.

Try to write a video  player with a few custom buttons for play/stop/etc. When your custom player is done, please add a way to play several videos one after another (what we call a playlist), etc.

Examples that can help you, created by students of earlier versions of this MOOC:

* [A custom player with nice CSS and buttons](http://jsbin.com/mayugik/1/edit?html,css,output)
* [Another simple one with custom buttons for play/stop/volume up/volume down](http://jsbin.com/neluwes/edit?html,css,output)...
* [Custom players with a small playlist composed of three songs by Queen](https://jsbin.com/vefiniq/7/edit?html,output)
* [AWESOME custom player created by GeorgianaB, with playlist, progress bar, CSS3 animations, etc. Check this out!](https://codepen.io/w3devcampus/pen/reQbow)

#### 2. A video quiz!

Create a quiz based on videos - here is a proposed story telling:

* a video is playing, then it stops at a given time, and you display a question such as: "who is this actor?" followed by some radio buttons + a proposal (see what we do with quizzes in this course): "Leonardo Di Caprio" or "Harisson Ford"?
* Once the question is answered, you display "Correct" or "Incorrect"
* Then the video continues....
* When the video ends, please show the final score.

A few hints:

1. Use an array with stop times, for example let stopTimes = [5, 10, 20]. This will mean "the video should stop at currentTime = 5, currentTime = 10, currentTime = 20".
2. You will use a timeupdate event listener on the video, like in the example from the live coding video, and the pause, play and stop methods from the video element JavaScript API. And also an ended event listener for detecting the end of the video.
3. Start from one of the example in the course (the one from the live coding video): try to make the video stop at 5s ,for example, and then display a question, and a "continue" button. When the button is pressed, the video goes on and stops a bit further, etc.
4. When this works (the video plays, then stops, you click, it continues, etc.), try to turn the displayed sentence into a quiz: add HTML radio buttons, and when you click the continue button, you will validate the answer, show "correct" or "incorrect", and maybe increment the score.
5. Feel free to add any feature(s) you'd like.

As always, do not forget to post the URL of your work in the forum so that we can enjoy your creation. Michel will inevitably give you his advice(s), and also tell you that you are the best ;))

1. **Displaying a map with the geolocation API**
   1. **Introduction to the geolocation API**

**The Geolocation API**

Let's start with an example!

Click the button to get your position and display a map as a picture. This may take some time, or fail, if a geolocation is not available with the device and connection you are using (e.g. at work, beyond a proxy). Try it anyway!

This example will be explained later on in the course...

## Introduction

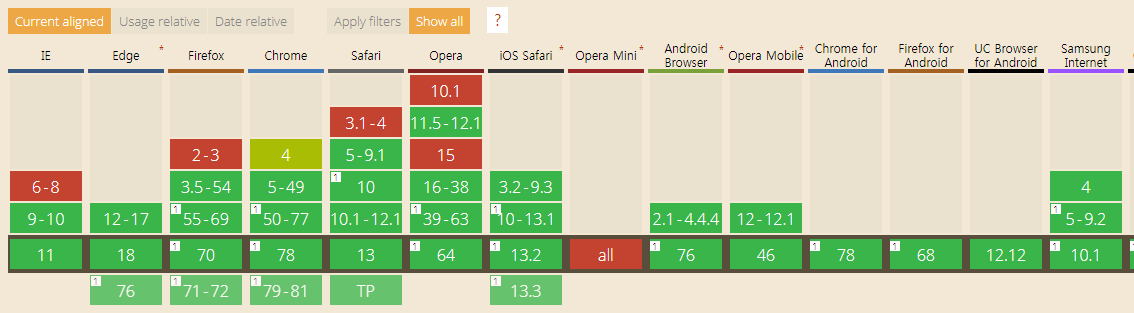
This chapter presents the new Geolocation API and illustrates its use with several examples.

The Geolocation HTML5 JavaScript API is implemented by most modern Web browsers, and uses different means to get the current location: GPS, GSM/3G triangulation, Wifi, IP address, etc.

It is possible to prompt the user to activate the GPS (this is what most GPS navigation software does on mobile phones), or ask for a particular mean among those available. It is also possible to track the current position when it changes. This is useful for writing a navigation application or for tracking in real time the position of different participants in the case of an application that involves several persons at the same time (using WebSockets, for example).

## Current support is excellent

As at February 2019, support for this API is excellent, both on mobile and on desktop devices.

****

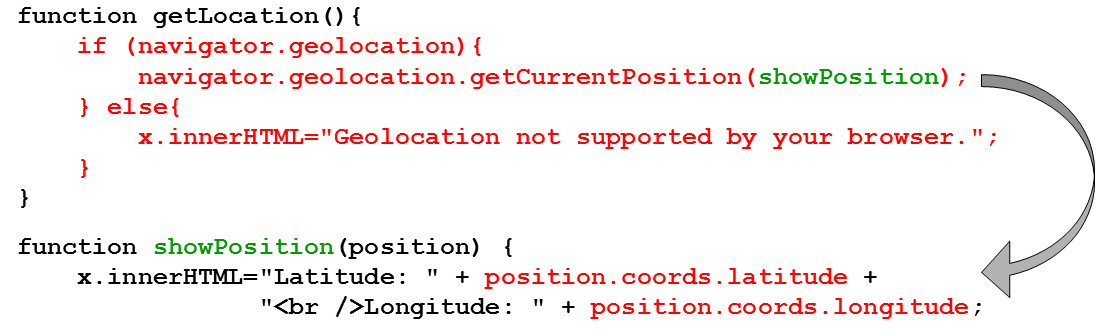
## Typical use

1. **navigator.geolocation.getCurrentPosition(showPosition, onError);**
3. function showPosition(position) {
4. console.log("latitude is: " + position.coords.latitude);
5. console.log("longitude is: " + position.coords.longitude);
6. }
8. function onError(err) {
9. console.log("Could not get the position");
10. }

Note that the first time you execute this example, for privacy reasons, the browser will ask if you agree to share your position with the application.

Source code of this typical example:

1. <!DOCTYPE html>
2. <html lang="en">
3. <head>
4. <meta charset="utf-8">
5. <title>Basic example of use of the geolocation API</title>
6. </head>
7. <body>
8. <p id="msg">Click the button to get your coordinates:</p>
9. <button onclick="getLocation()">Where am I ?</button>
10. <script>
11. var displayCoords=document.getElementById("msg");
12. function getLocation() {
13. if (navigator.geolocation) {
14. **navigator.geolocation.getCurrentPosition(showPosition);**
15. } else {
16. displayCoords.innerHTML="Geolocation API not supported by your browser.";
17. }
18. }
19. **function showPosition(position)** {
20. displayCoords.innerHTML="Latitude: " +**position.coords.latitude**+
21. "<br />Longitude: " +**position.coords.longitude**;
22. }
23. </script>
24. </body>
25. </html>

**Explanations:**

* Line 14 checks if the Web browser supports the geolocation API by testing the variable navigator.geolocation. If not null, then the geolocation API is supported.
* Line 15 calls navigator.geolocation.getCurrentPosition(showPosition) passing a callback function as a parameter (in this example we did not specify a callback in case of error). **When a current position is available, the callback function will be called asynchronously, and the input parameter of this callback function will be the current position,** like in the function showPosition(position) of the example.
* Line 22: the position objects has a coords property that is the object that holds the longitude and the latitude.

## External resource:

* [The W3C specification about the Geolocation API](http://www.w3.org/TR/geolocation-API/)(in Recommendation status, aka, Web standard) - 8 November 2016
  1. **Practical examples using the Google Map API**

# Three practical examples: use the geolocation API together with Google Maps

This section presents some examples of how to get a static map (a picture), using [the Google Static Map API](https://developers.google.com/maps/documentation/staticmaps/), how to display an interactive map using [the Google Map JavaScript API](https://developers.google.com/maps/tutorials/) and even how to get an estimation of a physical address from the longitude and latitude, using [the Google Reverse Geocoding JavaScript API](https://developers.google.com/maps/documentation/javascript/examples/geocoding-reverse).

The following three examples increase in complexity, but most of the code is reused and adapted without even reading the Google documentation about the different APIs.

**WARNING : since 2018, Google requires an API key in order to use interactive maps in HTML pages. In most situations, using such a key is free of charge.**

**Please follow**[**this YouTube tutorial**](https://www.youtube.com/watch?v=C3znRXBMYZo)**on how to get a Google Map API key.**

**In the examples of this JavaScript Introduction MOOC, we use our own key that is restricted to jsbin.com. It means that you can clone our examples or write your own ones in jsbin.com and use our key to make them work. However, for applications hosted somewhere else, you will need a personal API key.**

**You can find equivalent examples using the free, open source, Open Street Map and Leaflet APIs.**

**Here are a few that we have selected for you:**

* **Basic Map with leaflet and OSM :**[**https://jsfiddle.net/user2314737/tvwkgo34/**](https://jsfiddle.net/user2314737/tvwkgo34/)
* **Get your location using the geolocation API and display it on an OSM:**[**https://codepen.io/leemark/pen/vcbuf**](https://codepen.io/leemark/pen/vcbuf)
* **Another example that shows a map centered on your location:**[**https://codepen.io/dangvanthanh/pen/tlpLG**](https://codepen.io/dangvanthanh/pen/tlpLG)
* **Google map (using iframe, no key is necessary for that) v.s. OSM/Leaflet :**[**https://codepen.io/chris0stein/pen/qFKhg**](https://codepen.io/chris0stein/pen/qFKhg)
* **Reverse geocoding with OSM:**[**https://services.gisgraphy.com/static/leaflet/index.html**](https://services.gisgraphy.com/static/leaflet/index.html)

## Example 1 (easy):  how to get a static image map centered on your longitude and latitude

It also illustrates the use of the error callback from the previous section. The Google Map API is used to get an image centered at the longitude and latitude collected with the HTML5 Geolocation API.

Source code extract:

1. <!DOCTYPE html>
2. <html lang="en">
3. <head>
4. <meta charset="utf-8">
5. <title>Basic example of use of the geolocation API</title>
6. </head>
7. <body>
8. <p id="demo">Click the button to get your position:</p>
9. <button onclick="getLocation()">Try It</button>
10. <div id="mapholder"></div>
11. <script>
12. var x=document.getElementById("demo");
13. function getLocation()
14. {
15. if (navigator.geolocation)
16. {
17. navigator.geolocation.getCurrentPosition(showPosition,showError);
18. }
19. else{x.innerHTML="Geolocation is not supported by this browser.";}
20. }
22. function showPosition(position)
23. {
24. var latlon=position.coords.latitude+","+position.coords.longitude;
26. var img\_url="https://maps.googleapis.com/maps/api/staticmap?key=PUT\_HERE\_YOUR\_API\_KEY&center="
27. +latlon+"&zoom=14&size=400x300&sensor=false";
28. document.getElementById("mapholder").innerHTML="<img src='"+img\_url+"' />";
29. }
31. function showError(error)
32. {
33. switch(error.code)
34. {
35. case error.PERMISSION\_DENIED:
36. x.innerHTML="User denied the request for Geolocation."
37. break;
38. case error.POSITION\_UNAVAILABLE:
39. x.innerHTML="Location information is unavailable."
40. break;
41. case error.TIMEOUT:
42. x.innerHTML="The request to get user location timed out."
43. break;
44. case error.UNKNOWN\_ERROR:
45. x.innerHTML="An unknown error occurred."
46. break;
47. }
48. }
49. </script>
50. </body>
51. </html>

The magic occurs at line 23, where we use [the Google Static Map API](https://developers.google.com/maps/documentation/staticmaps/).

## Example 2 (a bit more complicated...) that shows how to display an interactive Google map centered on the current position

This example is just given "as is", as there are so many possibilities for rendering a map with [the Google Map API](https://developers.google.com/maps/tutorials/). However, we think having such a basic example might be useful.



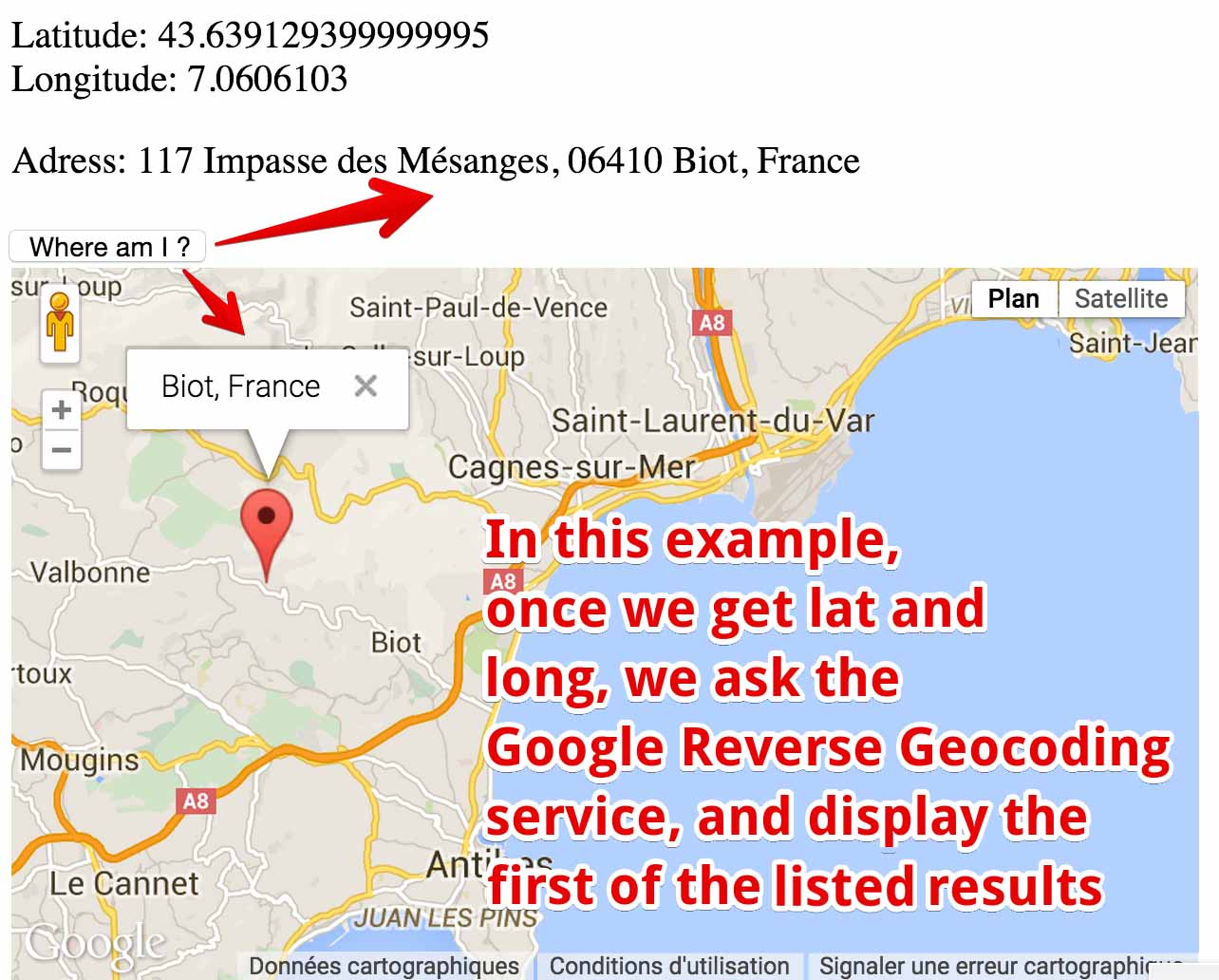
Source code of the example:

1. <!DOCTYPE html>
2. <html lang="en">
3. <head>
4. <meta charset="utf-8">
5. <title>Basic example of use of the geolocation API</title>
6. </head>
7. <body>
8. <!-- for position display -->
9. <div id="myposition"></div>
11. <!-- for gmap display -->
12. <div id="map" style="width:640px;height:480px"></div>
14. <!-- get gmap API -->
15. <script src="https://maps.google.com/maps/api/js?key=PUT\_HERE\_YOUR\_API\_KEY&sensor=false"></script>
17. <script>
18. // Default position
19. var centerpos = new google.maps.LatLng(48.579400,7.7519);
21. // default options for the google map
22. var optionsGmaps = {
23. center:centerpos,
24. navigationControlOptions: {style: google.maps.NavigationControlStyle.SMALL},
25. mapTypeId: google.maps.MapTypeId.ROADMAP,
26. zoom: 15
27. };
29. // Init map object
30. var map = new google.maps.Map(document.getElementById("map"), optionsGmaps);
32. if(navigator.geolocation) {
34. // callback function, called by getCurrentPosition() in case of success
35. function drawPosition(position) {
36. var infopos = "Got position : <br>";
37. infopos += "Latitude : "+position.coords.latitude +"<br>";
38. infopos += "Longitude: "+position.coords.longitude+"<br>";
39. infopos += "Altitude : "+position.coords.altitude +"<br>";
40. document.getElementById("myposition").innerHTML = infopos;
42. // Make new object LatLng for Google Maps
43. var latlng = new google.maps.LatLng(position.coords.latitude, position.coords.longitude);
45. // Add a marker at position
46. var marker = new google.maps.Marker({
47. position: latlng,
48. map: map,
49. title:"You are here"
50. });
51. // center map on longitude and latitude
52. map.panTo(latlng);
53. }
55. // callback function, called by getCurrentPosition() in case of error
56. function errorPosition(error) {
57. var info = "Error during geolocation : ";
58. switch(error.code) {
59. case error.TIMEOUT:
60. info += "Timeout !";
61. break;
62. case error.PERMISSION\_DENIED:
63. info += "You did not access to the geolocation API";
64. break;
65. case error.POSITION\_UNAVAILABLE:
66. info += "Position could not be determined";
67. break;
68. case error.UNKNOWN\_ERROR:
69. info += "Unknown error";
70. break;
71. }
72. document.getElementById("myposition").innerHTML = info;
73. }
74. navigator.geolocation.getCurrentPosition(drawPosition,errorPosition);
75. } else {
76. alert("Geolocation API not supported by your browser");
77. }
79. </script>
80. </body>
81. </html>

## Example 3 (advanced): how to get a physical address from the longitude and latitude

This is another example that obtains an address from longitude and latitude. It uses [the Google Reverse Geocoding JavaScript API.](https://developers.google.com/maps/documentation/javascript/examples/geocoding-reverse) For those of you who are really interested to know how this API works, please read the Google documentation and tutorials.

Without going into detail, the below example might be useful to copy/paste/adapt for trying to pre-fill a form where one is asked for an address. Geolocation is useful for guessing the country, city, zip code, street, etc. Some examples that use this feature will be given in the next section of the course.



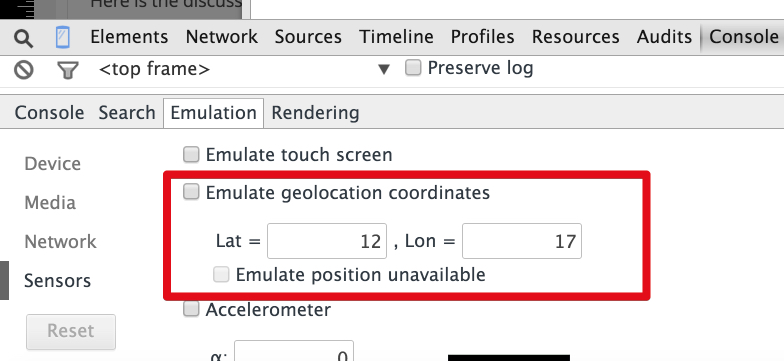
Source code of the example:

1. <!DOCTYPE html>
2. <html lang="en">
3. <head>
4. <meta charset="utf-8">
5. <title>Js bin </title>
6. <script src="https://maps.googleapis.com/maps/api/js?key=PUT\_HERE\_YOUR\_API\_KEY&v=3.exp&sensor=false"></script>
7. <script>
8. // p elements for displaying lat / long and address
9. var displayCoords, myAddress;
10. // used with the google apis
11. var geocoder;
12. var map;
13. var infowindow = new google.maps.InfoWindow();
14. var marker;
15. // Called when the page is loaded
16. function init() {
17. displayCoords=document.getElementById("msg");
18. myAddress = document.getElementById("address");
19. geocoder = new google.maps.Geocoder();
20. // In order to show something even before a user clicks on the button
21. var latlng = new google.maps.LatLng(34.0144, -6.83);
22. var mapOptions = {
23. zoom: 8,
24. center: latlng,
25. mapTypeId: 'roadmap'
26. }
27. map = new google.maps.Map(document.getElementById('map\_canvas'), mapOptions);
28. } // end of init()
29. // Called when the button is clicked
30. function getLocation() {
31. if (navigator.geolocation) {
32. navigator.geolocation.getCurrentPosition(showPosition);
33. } else {
34. displayCoords.innerHTML="Geolocation API not supported by your browser.";
35. }
36. }
37. // Called when a position is available
38. function showPosition(position) {
39. displayCoords.innerHTML="Latitude: " + position.coords.latitude +
40. "<br />Longitude: " + position.coords.longitude;
41. // Display the map
42. showOnGoogleMap(new google.maps.LatLng(position.coords.latitude,
43. position.coords.longitude));
44. }
45. function showOnGoogleMap(latlng) {
46. // Ask google geocoder for an address once we get a longitude and
47. // a latitude. In fact, the reverse geocoder sends back an array of "guesses"
48. // i.e. not just one address object, but several. Each entry in this array
49. // has several properties such as street, city, etc. We use the "formatted\_address"
50. // one here, but it might be interesting to get the detailed properties in other
51. // applications like a form with street, city, zip code etc.
52. geocoder.geocode({'latLng': latlng},reverseGeocoderSuccess);
53. function reverseGeocoderSuccess(results, status) {
54. if (status == google.maps.GeocoderStatus.OK) {
55. if (results[1]) {
56. map.setZoom(11);
57. marker = new google.maps.Marker({
58. position: latlng,
59. map: map
60. });
61. infowindow.setContent(results[1].formatted\_address);
62. infowindow.open(map, marker);
63. // Display address as text in the page
64. myAddress.innerHTML="Adress: " + results[0].formatted\_address;
65. } else {
66. alert('No surface address found');
67. }
68. } else {
69. alert('Geocoder failed due to: ' + status);
70. }
71. } // end of reverseGeocoderSuccess
72. } // end of showOnGoogleMap
73. </script>
74. </head>
75. <body onload="init()">
76. <title>HTML5 + Geolocalisation + Google Maps API Reverse Geocoding</title>
77. <p id="msg">Click the button to get your coordinates:</p>
78. <p id="address"></p>
79. <button onclick="getLocation()">Where am I ?</button>
80. <div id="map\_canvas" style="width: 500px; height: 300px"></div>
81. </body>
82. </html>
    1. **Discussion topics and projects**

Here is the discussion forum for this part of the course. Please either post your comments/observations/questions or share your creations.

See below for suggested topics of discussion and optional projects.

#### Suggested topics

* What features of a Web application do you think could benefit from geolocation?
* Do you know that you can simulate a position using the dev. tools of some browsers? Try exploring the dev. tools of Google Chrome. Also, there are browser extensions and applications that can help develop interactive maps. Please look for some of them and share your findings in the forum.  
  
* Can you recommend good tutorials about Google Map and about OpenStreetMap, the two main free services that propose maps on the fly?

#### Optional projects

* Here are a few project ideas. Your classmates and the team who prepared the course will be glad to try them and offer feedback. Please post URLs in this discussion forum. These projects are optional, meaning that they won't be graded.
* **Project 1 (easy)**: Add a map showing your location to one of your Web pages. Start with a simple, static map, then try with an interactive map. Reuse the examples from the course.
* **Project 2 (a bit harder)**: The examples provided in the course used Google maps, but why don't you try to do the same with Open Street Map?