About JavaScript and HTML5

HTML5 is composed of new elements, but it also comes with many JavaScript APIs for controlling video and sound, drawing and animating things in the new `<canvas>`element, for offline applications, persistence, geolocation, orientation, etc.

So yes, during this course, in particular during Week 3 and 4, you will have to do a bit of JavaScript. **But, DON'T PANIC!**

Here we provide a basic introduction to JavaScript. If you want to learn more, many resources are available on the Web; this document is simply here to give you a head start. Remember that one great thing about these MOOCs courses is that everybody can help each other. Some students are very good in JavaScript and are usually very happy to help others when they encounter difficulties.

You will learn a lot by looking at examples, tweaking them, cloning and modifying them, etc. Many previous students who were real JavaScript beginners managed to do all the assignments (drawing and animating a monster with keyboard/mouse interaction)! And they did this by just studying the provided examples.

EXTERNAL RESOURCES

- The book I used to learn JavaScript myself: Object Oriented JavaScript by Stoyan Stefanov
- Mozilla Developper Network has a JS guide too.

Extracts from the forum posts:

- *Video tutorials at Treehouse are very slick, this includes a basic JavaScript course.*

- *Codecademy is also very good - very nice JavaScript introduction for beginners. I recommend this for those with no JS knowledge as it starts from scratch.*

WHAT DO YOU NEED? HOW TO DEBUG? HOW TO CATCH ERRORS?

We will not look at the JavaScript syntax here, but more at "JavaScript in the browser", how it works, how to start writing code, etc.

First of all, you need to find a way to debug your code and see errors. If your work does not produce any results, you must know why!

For that you will use **the dev. tools of your browser**. Press *F12*in Windows or *cmd-alt-i* in Mac to open the dev. tools, then go to the console tab: **this is where errors will be displayed**, or messages of your own (use the `console.log(string)`JavaScript function in the JavaScript code embedded in your html page). In the console, you will be able to type any JavaScript command.

Let's look at this example on JS Bin:

```
<!DOCTYPE html>
<html>
  <head>
  <meta charset=utf-8 />
  <title>Web Audio API</title>
  <script>
   console.log("Some JavaScript code has been executed");
  </script>
  </head>
  <body>
    <h1>JavaScript debugging using the dev tool console</h1>
  </body>
</html>
```

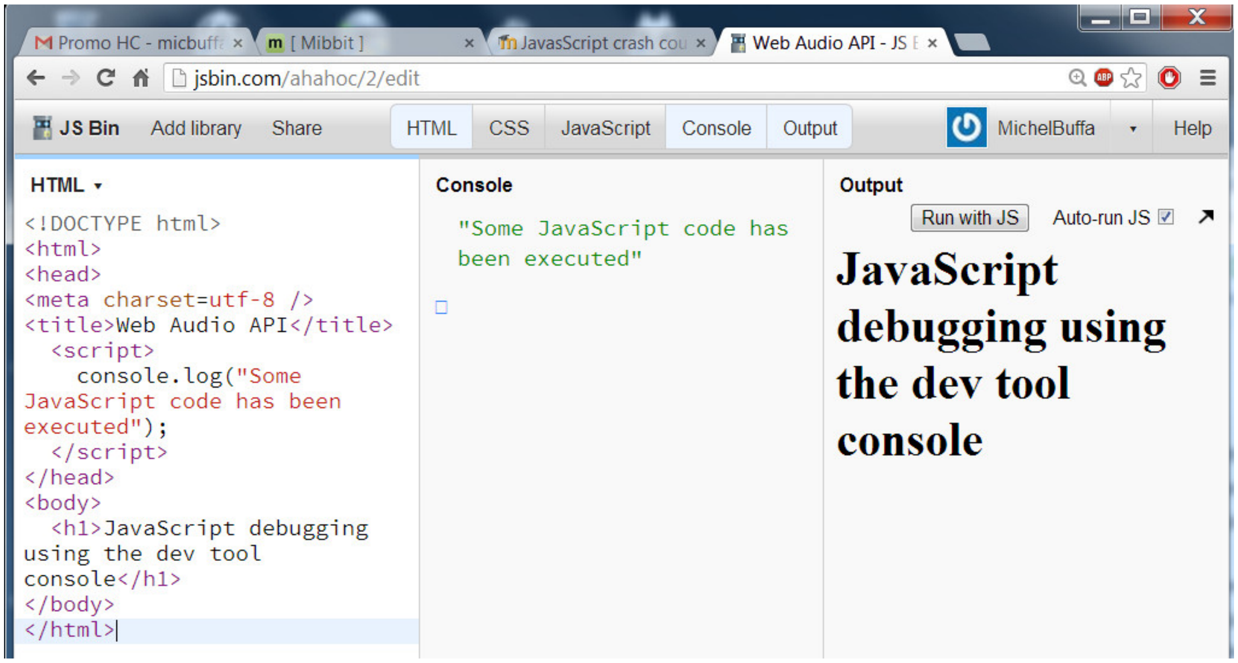
The simplest way to add JavaScript code in an HTML page, is by using the `<script>...</script>` element.

The code in this example is executed sequentially when the page is loaded: the JavaScript code is executed before the browser could see the rest of the page (as the `<script></script>` is located before the `<body>`).

The H1 element, for example, does not exist in the Document Object Model, and has not yet been displayed when the JavaScript code is executed. If we move the `<script></script>` at the end of the document, then the H1 would have been built before the JavaScript code is executed.

The only line of code we have is `console.log("Some JavaScript code has been executed");`

This means "display in the JavaScript console the message...". If we open the console tab provided by jsbin.com in a dedicated tab (that redirects all `console.log()` messages), and re-execute the page (just type a space at the end of a line, this will re-render the page and display the message in the console), we see the message in the console tab, as well as in the dev. tools console. This is illustrated by the image below:

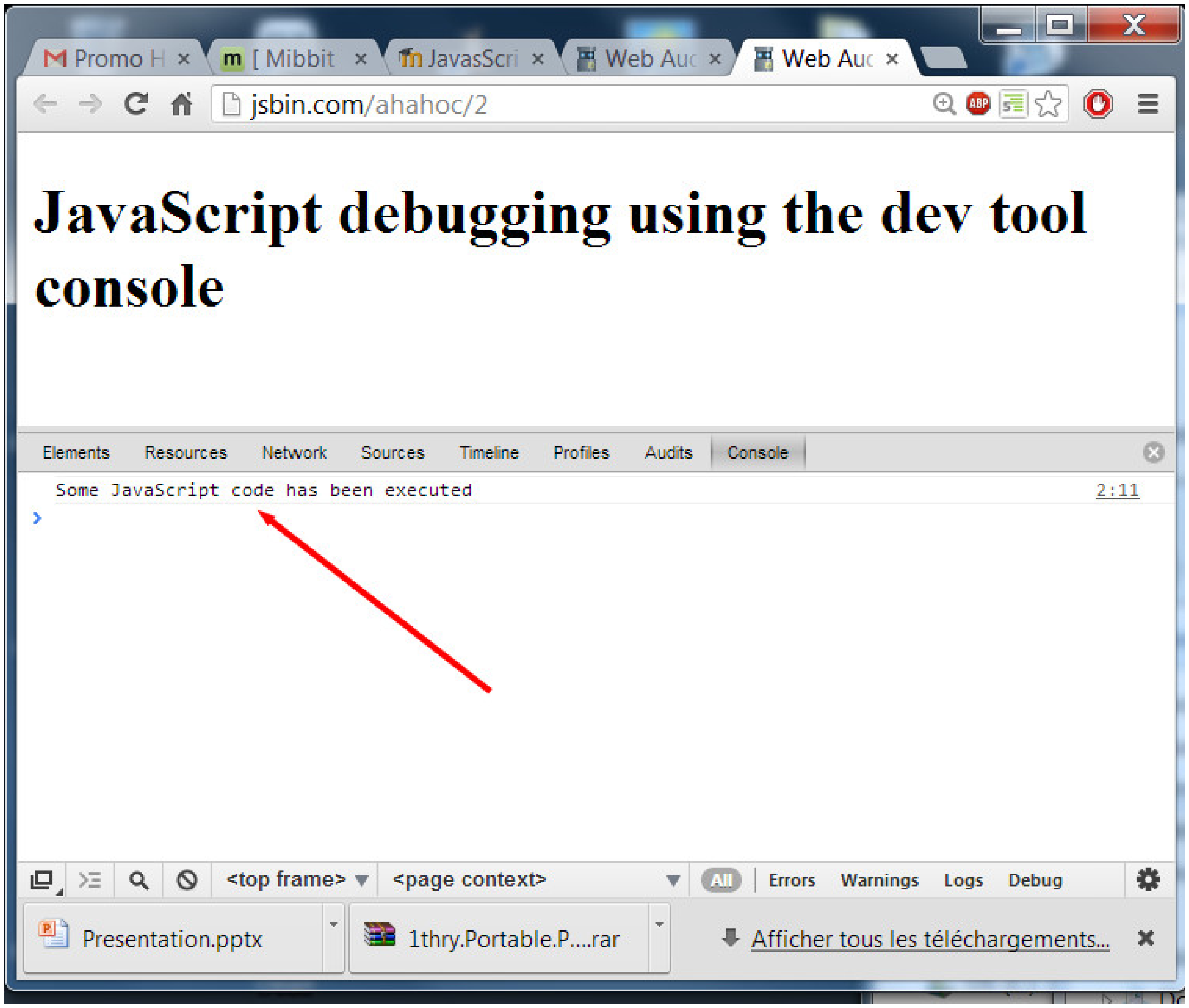


```
<!DOCTYPE html>
<html>
<head>
<meta charset=utf-8 />
<title>Web Audio API</title>
  <script>
    console.log("Some
JavaScript code has been
executed");
  </script>
</head>
<body>
  <h1>JavaScript debugging
using the dev tool
console</h1>
</body>
</html>
```

```
"Some JavaScript code has
been executed"
```

JavaScript debugging using the dev tool console

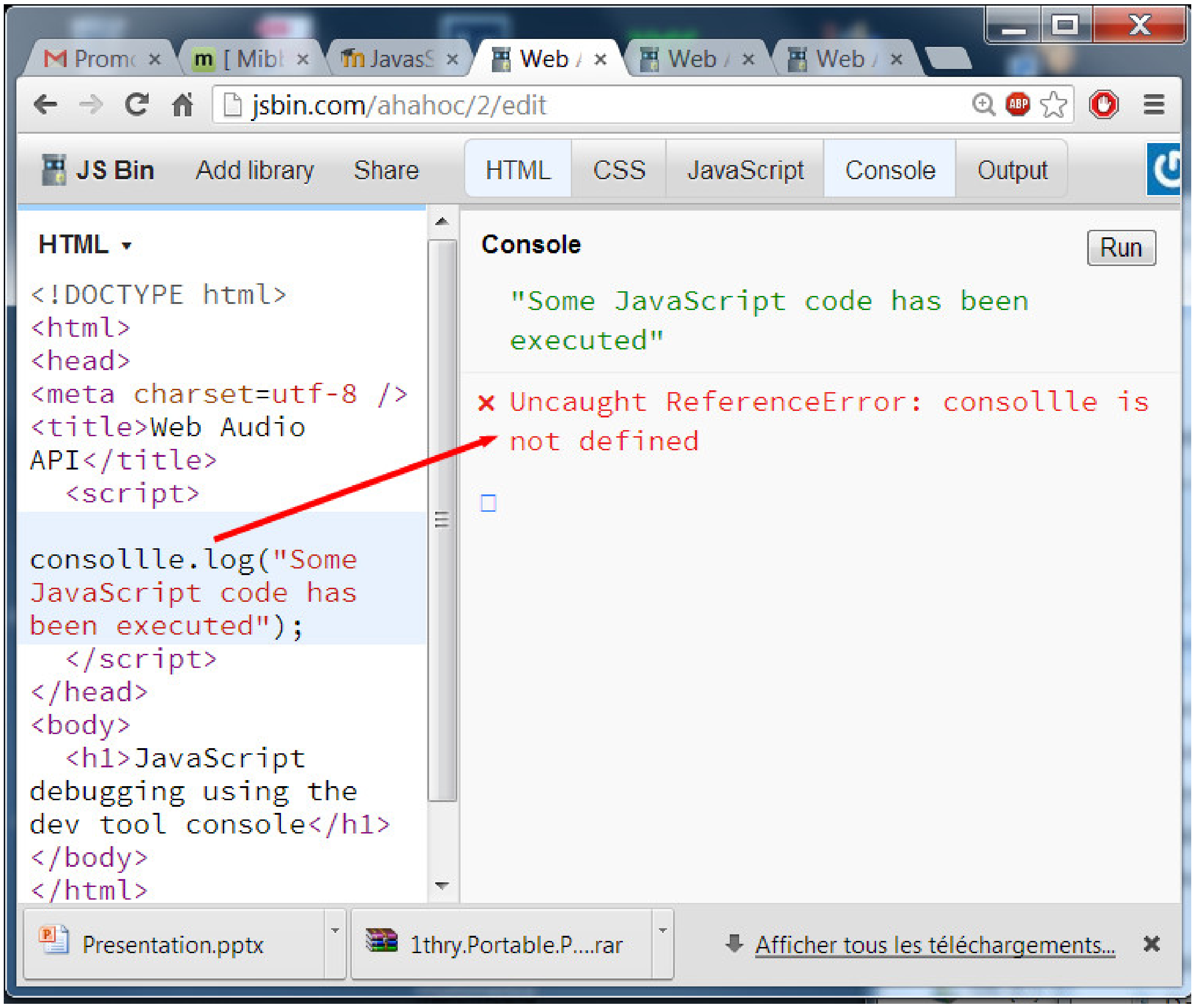
It is also possible to use the "real dev. tool console", and for this I recommend running the application in a single window, not in the JS Bin editor. Press the black arrow on the top right of the output window - this will render the page as a standalone Web page, then press *F12*. You should see:



JavaScript debugging using the dev tool console

```
Some JavaScript code has been executed                                    2:11
>
```

Ok, now, let's make an error: change `console.log()` into`consollle.log()`. Let's see what happens:

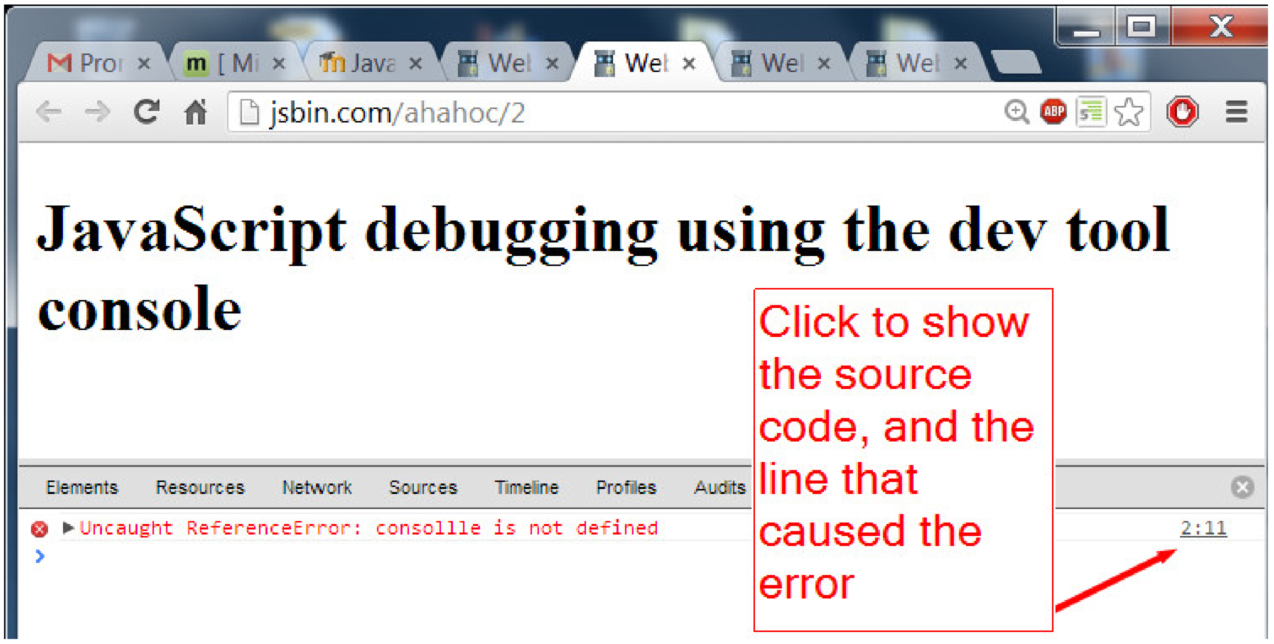


```
<!DOCTYPE html>
<html>
<head>
<meta charset=utf-8 />
<title>Web Audio API</title>
  <script>

consollle.log("Some JavaScript code has been executed");
  </script>
</head>
<body>
  <h1>JavaScript debugging using the dev tool console</h1>
</body>
</html>
```

```
"Some JavaScript code has been executed"
× Uncaught ReferenceError: consollle is not defined
```

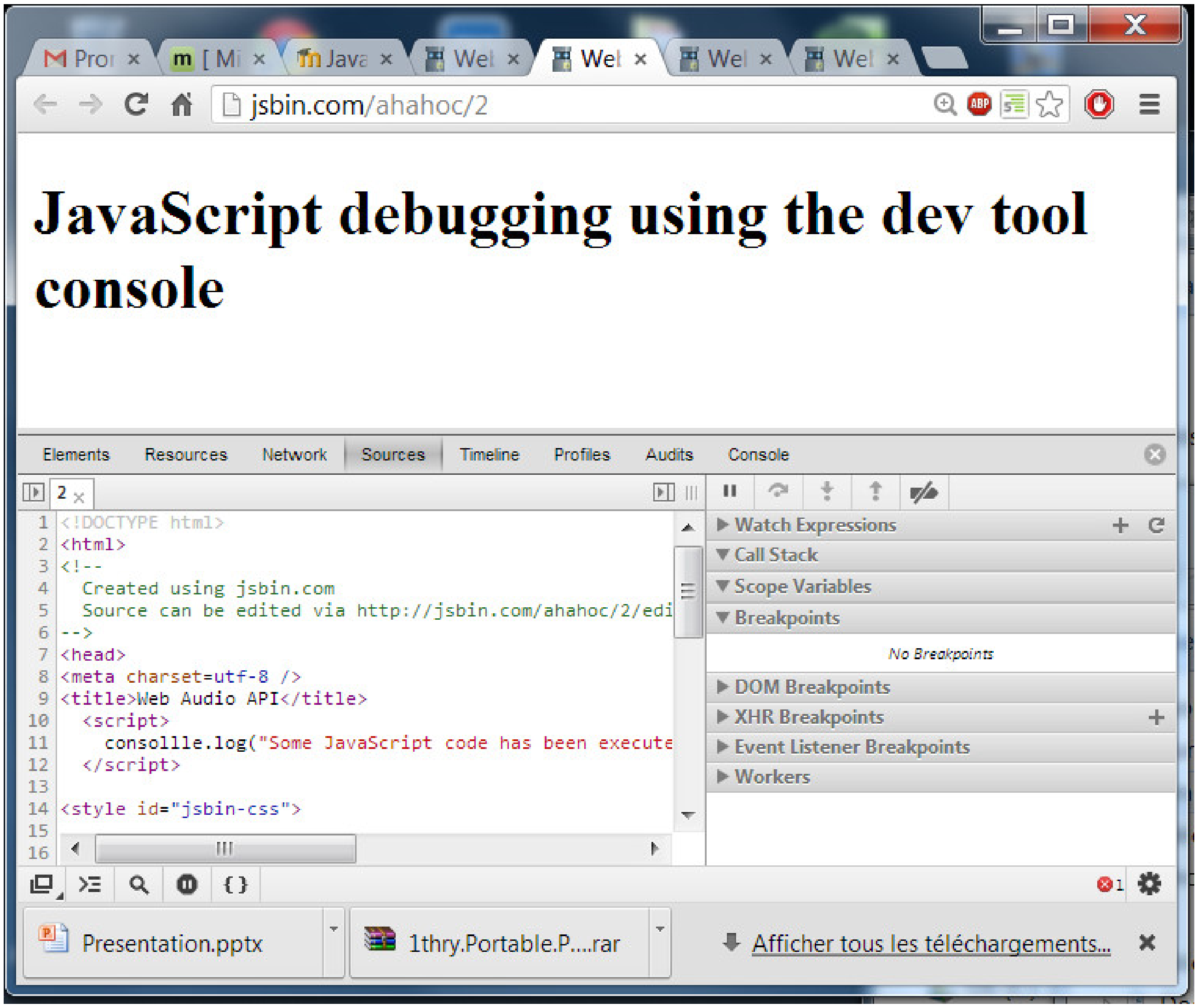
And if we run it standalone and use the dev. tool console:



jsbin.com/ahahoc/2
JavaScript debugging using the dev tool console
Elements
Resources
Network
Sources
Timeline
Profiles
Audits
Click to show the source code, and the line that caused the error

```
▶Uncaught ReferenceError: consollle is not defined                2:11
>
```

And if we click on the line number in the right, the dev. tool shows the source code centered on the line that caused the error:



JavaScript debugging using the dev tool console

```
<!DOCTYPE html>
<html>
<!--
  Created using jsbin.com
  Source can be edited via http://jsbin.com/ahahoc/2/edi
-->
<head>
<meta charset=utf-8 />
<title>Web Audio API</title>
  <script>
    consollle.log("Some JavaScript code has been execute
  </script>

<style id="jsbin-css">

```

Without such tools, debugging JavaScript code is impossible. So you need to look at some basic tutorials on how to use the dev. tools of your browsers, since they differ from one another in the way they work - although the principles remain the same.