The new HTML5 `DOWNLOAD` attribute for `<A HREF>`

External resources:

- The W3C specification about "downloading resources"
- Eric Bilderman's presentation (he is a Chrome developer)

OLD WAY TO DOWNLOAD FILES USING HTML AND HTTP

Everyone knows the classic way to make hyperlinks, using `<a href="...">`some text`</a>`. What happens when you click on the hyperlink depends on the MIME type received by the browser. If you link to a file the browser knows how to render (an html page, a gif, jpg, or png image, etc.) there are good chances that the MIME type received by the browser will be something like this:

```
Content-type: text/html, text/plain, image/gif, image/jpg, etc.
```

For example, HTML code such as this:

```
<a href="toto.jpg">
   please right click this link to download
   the toto.jpg picture</a>
```

...will ask the remote HTTP server to send back the `toto.jpg` file. The browser will receive in the response HTTP header from the server (and by default the browser will display the image in a new tab):

```
...
```

```
Content-type: image/jpg
...
```

However, if the link points to some PHP code, Java servlet code, or any kind of script/application on the server side, this remote server code can send in its HTTP response a `Content-type` that may force the browser to download the image instead of rendering it.

It may also propose a name for the file to be downloaded that may be different from the one that appears in the URL of the href attribute. This can be done by generating, in addition to the Content-type line in the response HTTP header, a `Content-Disposition`line that looks like this:

```
Content-Disposition: attachment; filename="MyImage.png";
```

Here are some extracts from a Java Servlet that generate a zip file and forces the browser to propose downloading it using a specified name:

```
protected void doGet(HttpServletRequest request, HttpServletResponseresponse)
 throws ServletException, IOException {
  try {
   // Build the zip file
    String path = getServletContext().getRealPath("data");
    File directory = new File(path);
    String[] files = directory.list();
    if (files != null && files.length > 0) {
     byte[] zip = zipFiles(directory, files);
     ServletOutputStream sos = response.getOutputStream();
     // generate a HTTP response that forces the download
     response.setContentType("application/zip");
     response.setHeader("Content-Disposition",
                 "attachment; filename=\"DATA.ZIP\"");
     sos.write(zip); sos.flush();
    }
```

```
    } catch (Exception e) {
      e.printStackTrace();
20.   }
    }
```

The above example will cause the browser that invoked this server-side code to start the download of a file named "DATA.ZIP".

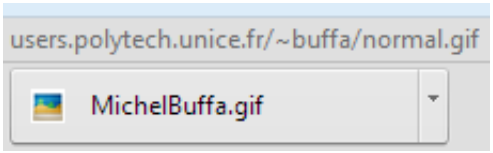
NEW WAY TO DOWNLOAD A FILE USING AN ARBITRARY NAME: THE DOWNLOAD ATTRIBUTE

HTML5 proposes using a new attribute named `download` to download resources rather than navigating to them. The example below shows how to trigger the download of an image by the browser (instead of rendering it, which is the default behavior) with a name different from the name of the resource.

```
<a href="/asset-v1:W3Cx+W3C-HTML5+2015T3+type@asset+block/normal.gif"
  download="MichelBuffa.gif">
   download a picture of Michel Buffa
</a>
```

Here is the result of this link (just click on it): download a picture of Michel Buffa.

This will indeed force the download of an image with a filename different from its original filename on the server side. Here is a screen capture of the Web browser while downloading the picture. We can see in the status bar the name of the link (the image is "`normal.gif`") and the downloaded file is "`MichelBuffa.gif`":



Note: this attribute is not supported by IE or Safari yet, as of February 2015.

See http://caniuse.com/#search=download, for an up-to-date status.

WARNING: since 2015, and for security reasons, **the image should be located on the same domain as the HTML page that contains the link** (use a relative URL works well, for example, but linking a page on another domain will not work, it willkeep its original name).

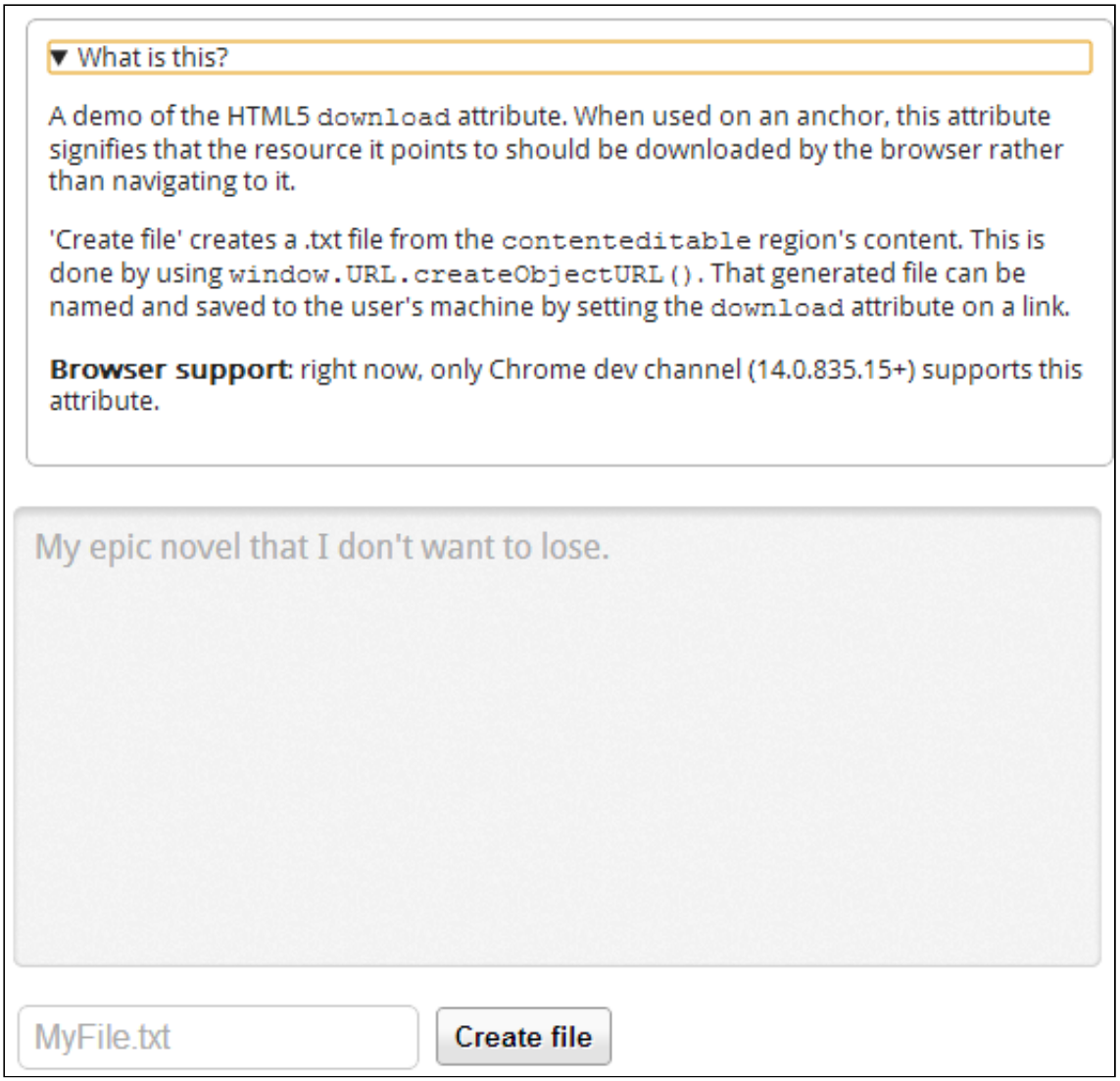
INTERESTING APPLICATIONS: SERVERLESS DOWNLOAD

Serverless download demo (by E.Bilderman)

This demo shows the use of the `download` attribute together with the HTML5 File, FileSystem and FileWriter APIs (to be studied later in this course) for generating on-the-fly content from JavaScript code, and proposing to download it to a file.

We won't detail this demo here, it's just for any of you curious to see what can be done with this new `download` attribute. As the FileWriter and FileSystem APIs are still supported only by Google Chrome (other browsers need polyfills), you will need Google Chrome to try it today.

We have also put the simplified source code of this demo on jsbin.com for you to play with: http://jsbin.com/amoxah/2/edit, the original demo by E.Bilderman is at http://html5-demos.appspot.com/static/a.download.html.



HTML5 TRANSLATE ATTRIBUTE

EXTERNAL RESOURCES:

- The W3C specification about the translate attribute,
- Interesting blog post about Google translate, Microsoft translator and the translate attribute
- Using HTML's `translate` attribute

INTRODUCTION

HTML5 brings us a new `translate` attribute. This attribute is used to limit the impact of translation tools such as Google Translate by prohibiting the translation of certain content. In many cases some parts of a document should not be translated.

Use cases include:

- HTML pages that contain source code: you will certainly not like to see the Java or PHP or whatever programming language parts of your page translated into another spoken language!
- Video game Web sites that propose cheat codes; the codes do not have to be translated,
- Street names, author names in an "about" page must not be translated,
- etc.

Both Google translate and Microsoft online translation services already provide the ability to prevent translation of content by adding markup to your content, although they do it in (multiple) different ways. Hopefully, the new attribute will help significantly by providing a standard approach.

PRINCIPLE: GIVE HINTS TO TRANSLATING TOOLS

The W3C specification about the translate attribute tells us that

"The `translate` attribute is an enumerated attribute that is used to specify whether an element's attribute values and the values of its Text node children are to be translated when the page is localized, or whether to leave them unchanged.

The attribute's keywords are the empty string, `yes`, and `no`. The empty string and the yes keyword map to the `yes` state. The `no` keyword maps to the no state. In addition, there is a third state, the inherit state, which is the missing value default (and the invalid value default)."

Example illustrating how to specify parts of an HTML element that should not be translated:

```
<span translate="no" class="author">Michel Ham</span>
```

In the above example, a `<span>` element defines an author (of a blog, for example) who is named Michel Ham. However, his family name is the same as pork and would be translated to "Michel Jambon" in French, or Michel Jamón in Spanish...

Using the `translate="no"` attribute should prevent this behavior...

```
<span translate="no" class="author">Michel Ham</span> is a professor from the University of Nice,France.
```

Will be correctly translated into French by:

```
"Michel Ham est un professeur de l'Université de Nice, France."
```

...where all of the end of the sentence has been translated except the author's name.

INHERITANCE BETWEEN ELEMENTS

When you define an element as not being translatable, its children inherit this behavior and are themselves not translatable. The reverse is also true.

```
<p translate="no">This is a text in a paragraph element, that should not be translated: the p element has a translate="no" attribute.<span> This part that is in a span element embedded within the paragraph. It does not have a translate attribute but inherits the translation-mode of the p and will not be translated too</span>. This is the end of the paragraph...</ p>
```