The head of an HTML document is the part that is not displayed in the web browser when the page is loaded. It contains information such as the page `<title>`, links to CSS (if you choose to style your HTML content with CSS), links to custom favicons, and other metadata (data about the HTML, such as the author, and important keywords that describe the document.) In this article we'll cover all of the above and more, in order to give you a good basis for working with markup.

Prerequisites:	Basic HTML familiarity, as covered in Getting started with HTML.
Objective:	To learn about the HTML head, its purpose, the most important items it can contain, and what effect it can have on the HTML document.

What is the HTML head?

Let's revisit the simple HTML document we covered in the previous article:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My test page</title>
  </head>
  <body>
    <p>This is my page</p>
  </body>
</html>
```

The HTML head is the contents of the `<head>` element — unlike the contents of the `<body>` element (which are displayed on the page when loaded in a browser), the head's content is not displayed on the page. Instead, the head's job is to contain metadata about the document. In the above example, the head is quite small:

```
<head>
  <meta charset="utf-8">
  <title>My test page</title>
</head>
```

In larger pages however, the head can get quite full. Try going to some of your favorite websites and use the developer tools to check out their head contents. Our aim here is not to show you how to use everything that can possibly be put in the head, but rather to teach you how to use the major elements that you'll want to include in the head, and give you some familiarity. Let's get started.

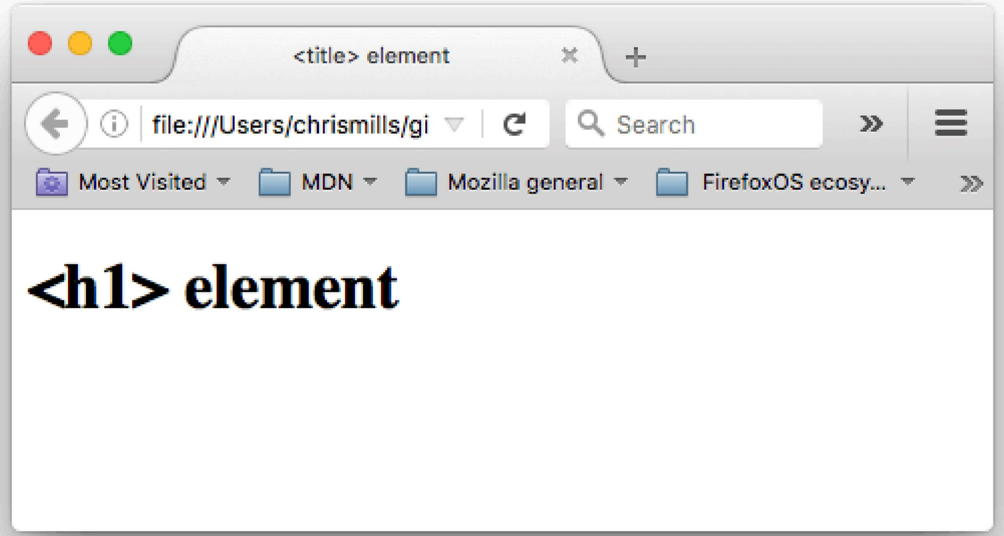
Adding a title

We've already seen the `<title>` element in action — this can be used to add a title to the document. This however can get confused with the `<h1>` element, which is used to add a top level heading to your body content — this is also sometimes referred to as the page title. But they are different things!

- The `<h1>` element appears on the page when loaded in the browser — generally this should be used once per page, to mark up the title of your page content (the story title, or news headline, or whatever is appropriate to your usage.)
- The `<title>` element is metadata that represents the title of the overall HTML document (not the document's content.)

Active learning: Inspecting a simple example

1. To start off this active learning, we'd like you to go to our GitHub repo and download a copy of our title-example.html page. To do this, either
    1. Copy and paste the code out of the page and into a new text file in your code editor, then save it in a sensible place.
    2. Press the "Raw" button on the GitHub page, which causes the raw code to appear (possibly in a new browser tab). Next, choose your browser's *File > Save Page As...*menu and choose a sensible place to save the file.
2. Now open the file in your browser. You should see something like this:

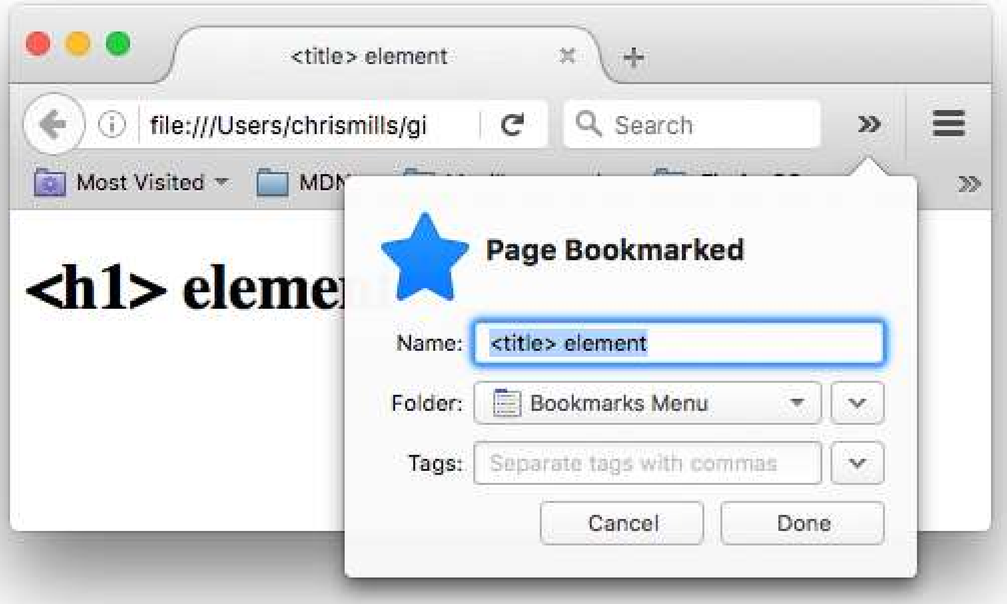


<h1> element

It should now be completely obvious where the `<h1>` content appears, and where the `<title>` content appears!

3. You should also try opening the code up in your code editor, editing the contents of these elements, then refreshing the page in your browser. Have some fun with it.

The `<title>` element contents are also used in other ways. For example, if you try bookmarking the page (*Bookmarks > Bookmark This Page* or the star icon in the URL bar in Firefox), you will see the `<title>` contents filled in as the suggested bookmark name.



The `<title>` contents are also used in search results, as you'll see below.

Metadata: the <meta> element

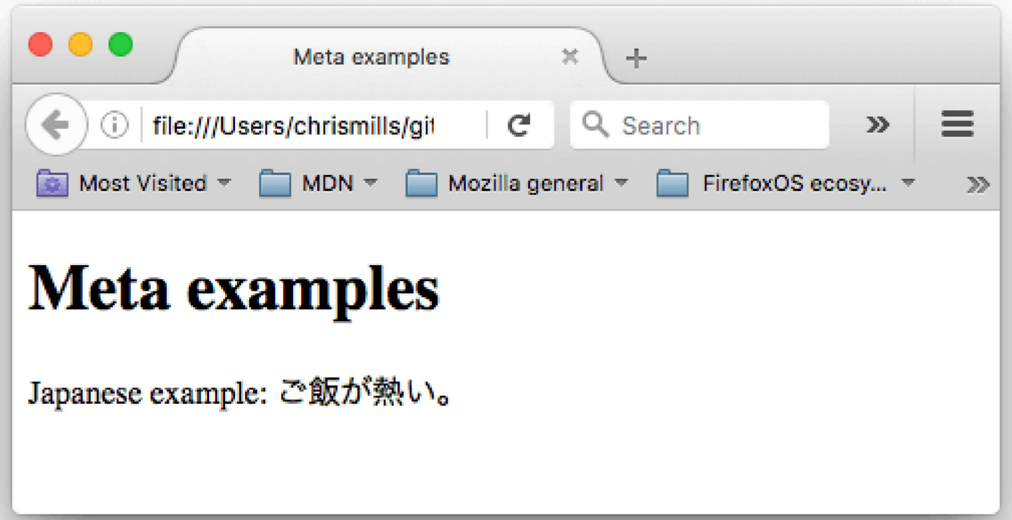
Metadata is data that describes data, and HTML has an "official" way of adding metadata to a document — the `<meta>` element. Of course, the other stuff we are talking about in this article could also be thought of as metadata too. There are a lot of different types of `<meta>` elements that can be included in your page's <head>, but we won't try to explain them all at this stage, as it would just get too confusing. Instead, we'll explain a few things that you might commonly see, just to give you an idea.

Specifying your document's character encoding

In the example we saw above, this line was included:

```
<meta charset="utf-8">
```

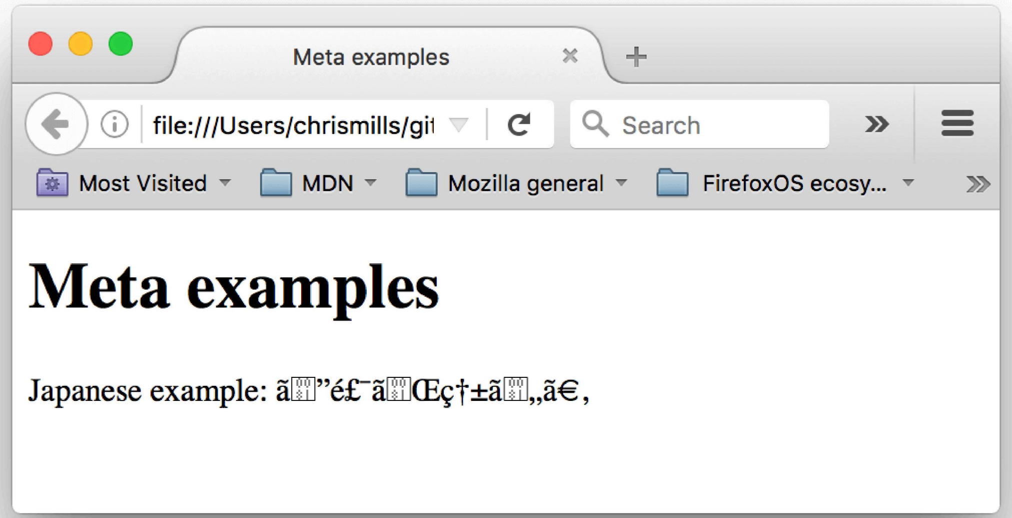
This element simply specifies the document's character encoding — the character set that the document is permitted to use. `utf-8` is a universal character set that includes pretty much any character from any human language. This means that your web page will be able to handle displaying any language; it's therefore a good idea to set this on every web page you create! For example, your page could handle English and Japanese just fine:



Meta examples

Japanese example: ご飯が熱い。

If you set your character encoding to `ISO-8859-1`, for example (the character set for the Latin alphabet), your page rendering may appear all messed up:



Meta examples

Japanese example: ã□”é£¯ã□Œç†±ã□„ã€,

> **Note**: Some browsers (e.g. Chrome) automatically fix incorrect encodings, so depending on what browser you use, you may not see this problem anyway. You should still set an encoding of `utf-8` on your page anyway, to avoid any potential problems in other browsers.

Active learning: Experiment with character encoding

To try this out, revisit the simple HTML template you obtained in the previous section on `<title>` (the title-example.html page), try changing the meta charset value to `ISO-8859-1`, and add the Japanese to your page. This is the code we used:

```
<p>Japanese example: ご飯が熱い。</p>
```

Adding an author and description

Many `<meta>` elements include `name` and `content` attributes:

- `name` specifies the type of meta element it is; what type of information it contains.
- `content` specifies the actual meta content.

Two such meta elements that are useful to include on your page define the author of the page, and provide a concise description of the page. Let's look at an example:

```
<meta name="author" content="Chris Mills">
<meta name="description" content="The MDN Web Docs Learning Area aims t
complete beginners to the Web with all they need to know to get
started with developing web sites and applications.">
```

Specifying an author is beneficial in many ways: it is useful to be able to understand who wrote the page, if you have any questions about the content and you would like to contact them. Some content management systems have facilities to automatically extract page author information and make it available for such purposes.

Specifying a description that includes keywords relating to the content of your page is useful as it has the potential to make your page appear higher in relevant searches performed in search engines (such activities are termed Search Engine Optimization, or SEO.)

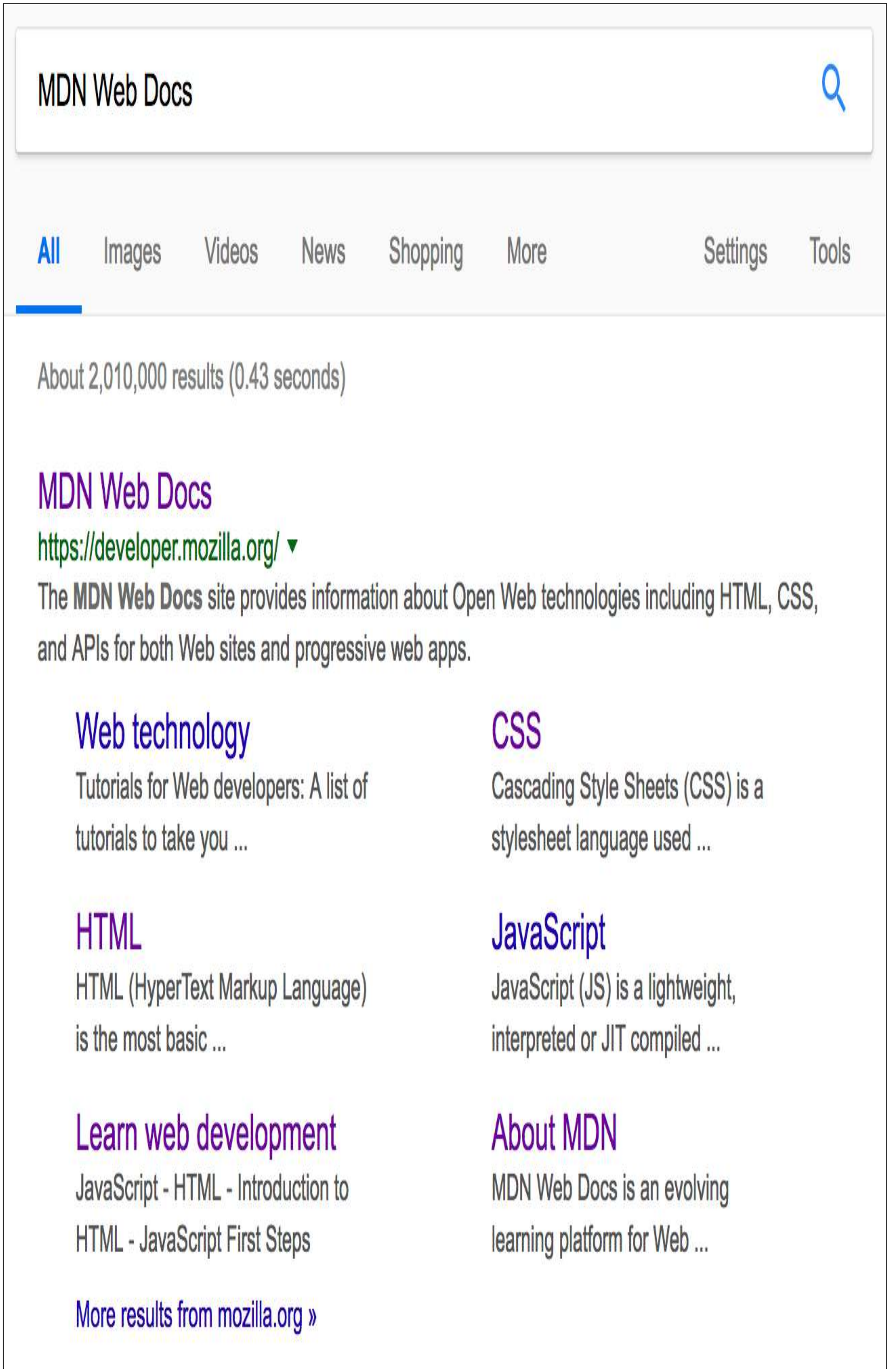
Active learning: The description's use in search engines

The description is also used on search engine result pages. Let's go through an exercise to explore this

1. Go to the front page of The Mozilla Developer Network.
2. View the page's source (Right/`Ctrl` + click on the page, choose *View Page Source* from the context menu.)
3. Find the description meta tag. It will look something like this (although it may change over time):

   ```
   <meta name="description" content="The MDN Web Docs site
     provides information about Open Web technologies
     including HTML, CSS, and APIs for both Web sites and
     progressive web apps.">
   ```

4. Now search for "MDN Web Docs" in your favorite search engine (We used Google.) You'll notice the description `<meta>` and `<title>` element content used in the search result — definitely worth having!



About 2,010,000 results (0.43 seconds)

MDN Web Docs
https://developer.mozilla.org/ ▾
The **MDN Web Docs** site provides information about Open Web technologies including HTML, CSS, and APIs for both Web sites and progressive web apps.

Web technology
Tutorials for Web developers: A list of tutorials to take you ...

CSS
Cascading Style Sheets (CSS) is a stylesheet language used ...

HTML
HTML (HyperText Markup Language) is the most basic ...

JavaScript
JavaScript (JS) is a lightweight, interpreted or JIT compiled ...

Learn web development
JavaScript - HTML - Introduction to HTML - JavaScript First Steps

About MDN
MDN Web Docs is an evolving learning platform for Web ...

More results from mozilla.org »

Note: In Google, you will see some relevant subpages of MDN Web Docs listed below the main homepage link — these are called sitelinks, and are configurable in Google's webmaster tools — a way to make your site's search results better in the Google search engine.

Note: Many `<meta>` features just aren't used any more. For example, the keyword `<meta>` element (`<meta name="keywords" content="fill, in, your, keywords, here">`) — which is supposed to provide keywords for search engines to determine relevance of that page for different search terms — is ignored by search engines, because spammers were just filling the keyword list with hundreds of keywords, biasing results.

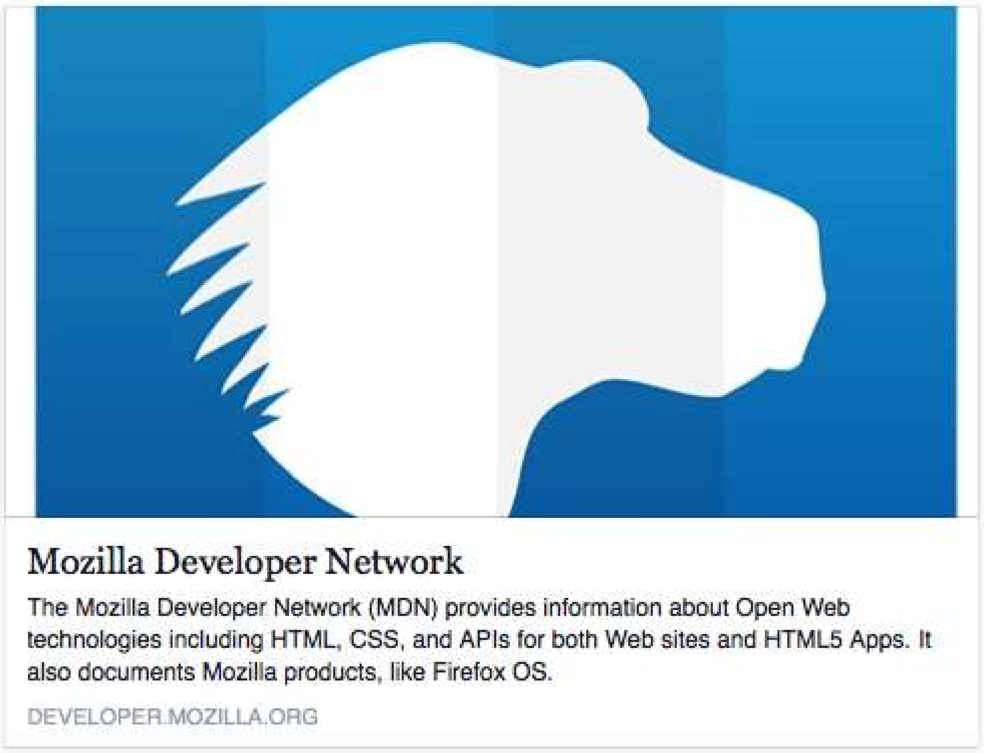
Other types of metadata

As you travel around the web, you'll find other types of metadata, too. A lot of the features you'll see on websites are proprietary creations, designed to provide certain sites (such as social networking sites) with specific pieces of information they can use.

For example, Open Graph Data is a metadata protocol that Facebook invented to provide richer metadata for websites. In the MDN Web Docs sourcecode, you'll find this:

```
<meta property="og:image" content="https://developer.cdn.mozilla.net/st
<meta property="og:description" content="The Mozilla Developer Network
information about Open Web technologies including HTML, CSS, and APIs f
and HTML5 Apps. It also documents Mozilla products, like Firefox OS.">
<meta property="og:title" content="Mozilla Developer Network">
```

One effect of this is that when you link to MDN Web Docs on facebook, the link appears along with an image and description: a richer experience for users.



Mozilla Developer Network

The Mozilla Developer Network (MDN) provides information about Open Web technologies including HTML, CSS, and APIs for both Web sites and HTML5 Apps. It also documents Mozilla products, like Firefox OS.

DEVELOPER.MOZILLA.ORG

Twitter also has its own similar proprietary metadata called Twitter Cards, which has a similar effect when the site's URL is displayed on twitter.com. For example:

```
<meta name="twitter:title" content="Mozilla Developer Network">
```

Adding custom icons to your site

To further enrich your site design, you can add references to custom icons in your metadata, and these will be displayed in certain contexts. The most commonly used of these is the **favicon** (short for "favorites icon", referring to its use in the "favorites" or "bookmarks" lists in browsers).

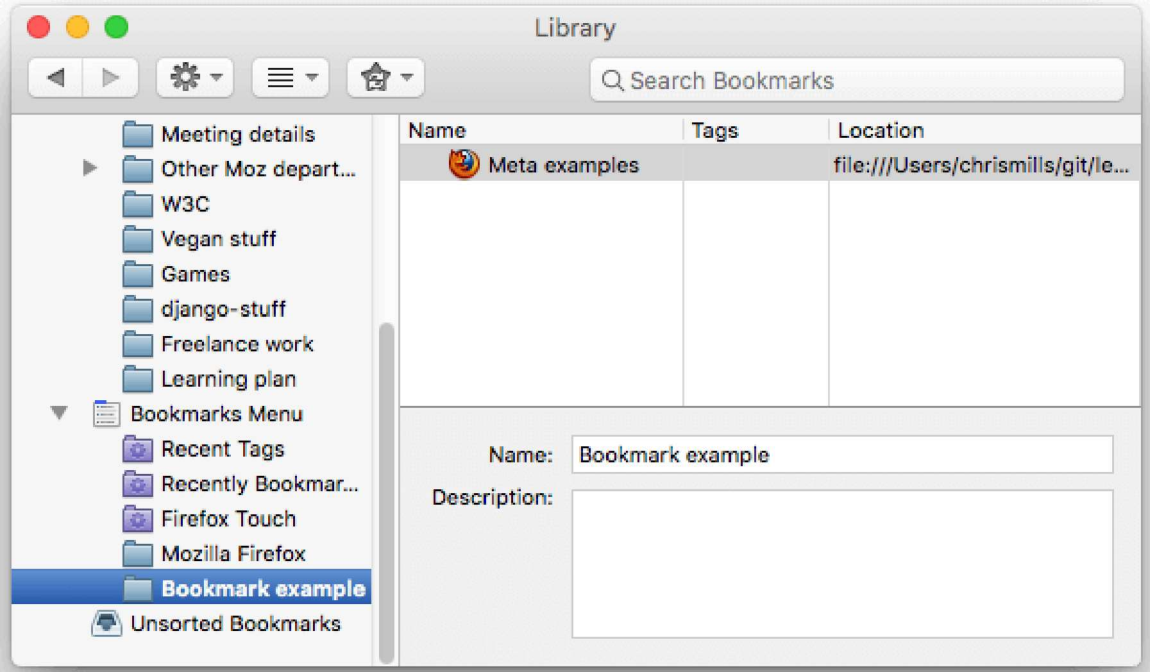
The humble favicon has been around for many years. It is the first icon of this type: a 16-pixel square icon used in multiple places. You may see (depending on the browser) favicons displayed in the browser tab containing each open page, and next to bookmarked pages in the bookmarks panel.

A favicon can be added to your page by:

1. Saving it in the same directory as the site's index page, saved in `.ico` format (most browsers will support favicons in more common formats like `.gif` or `.png`, but using the ICO format will ensure it works as far back as Internet Explorer 6.)
2. Adding the following line into your HTML's `<head>` block to reference it:

```
    <link rel="shortcut icon" href="favicon.ico" type="image/x-icon
```

Here is an example of a favicon in a bookmarks panel:



There are lots of other icon types to consider these days as well. For example, you'll find this in the source code of the MDN Web Docs homepage:

```
    <!-- third-generation iPad with high-resolution Retina display: -->
    <link rel="apple-touch-icon-precomposed" sizes="144x144" href="https://
    <!-- iPhone with high-resolution Retina display: -->
    <link rel="apple-touch-icon-precomposed" sizes="114x114" href="https://
    <!-- first- and second-generation iPad: -->
    <link rel="apple-touch-icon-precomposed" sizes="72x72" href="https://de
    <!-- non-Retina iPhone, iPod Touch, and Android 2.1+ devices: -->
    <link rel="apple-touch-icon-precomposed" href="https://developer.cdn.mo
    <!-- basic favicon -->
    <link rel="shortcut icon" href="https://developer.cdn.mozilla.net/stati
```

The comments explain what each icon is used for — these elements cover things like providing a nice high resolution icon to use when the website is saved to an iPad's home screen.

Don't worry too much about implementing all these types of icon right now — this is a fairly advanced feature, and you won't be expected to have knowledge of this to progress through the course. The main purpose here is to let you know what such things are, in case you come across them while browsing other websites' source code.

> **Note:** If your site uses a Content Security Policy (CSP) to enhance its security, the policy applies to the favicon. If you encounter problems with the favicon not loading, verify that the `Content-Security-Policy` header's `img-src` directive is not preventing access to it.

Applying CSS and JavaScript to HTML

Just about all websites you'll use in the modern day will employ CSS to make them look cool, and JavaScript to power interactive functionality, such as video players, maps, games, and more. These are most commonly applied to a web page using the `<link>` element and the `<script>` element, respectively.

- The `<link>` element should always go inside the head of your document. This takes two attributes, `rel="stylesheet"`, which indicates that it is the document's stylesheet, and `href`, which contains the path to the stylesheet file:

  ```
  <link rel="stylesheet" href="my-css-file.css">
  ```

- The `<script>` element should also go into the head, and should include a `src` attribute containing the path to the JavaScript you want to load, and `defer`, which basically instructs the browser to load the JavaScript at the same time as the page's HTML. This is useful as it makes sure that the HTML is all loaded before the JavaScript runs, so that you don't get errors resulting from JavaScript trying to access an HTML element that doesn't exist on the page yet. There are actually a number of ways to handle loading JavaScript on your page, but this is the most foolproof one to use for modern browsers (for others, read Script loading strategies).

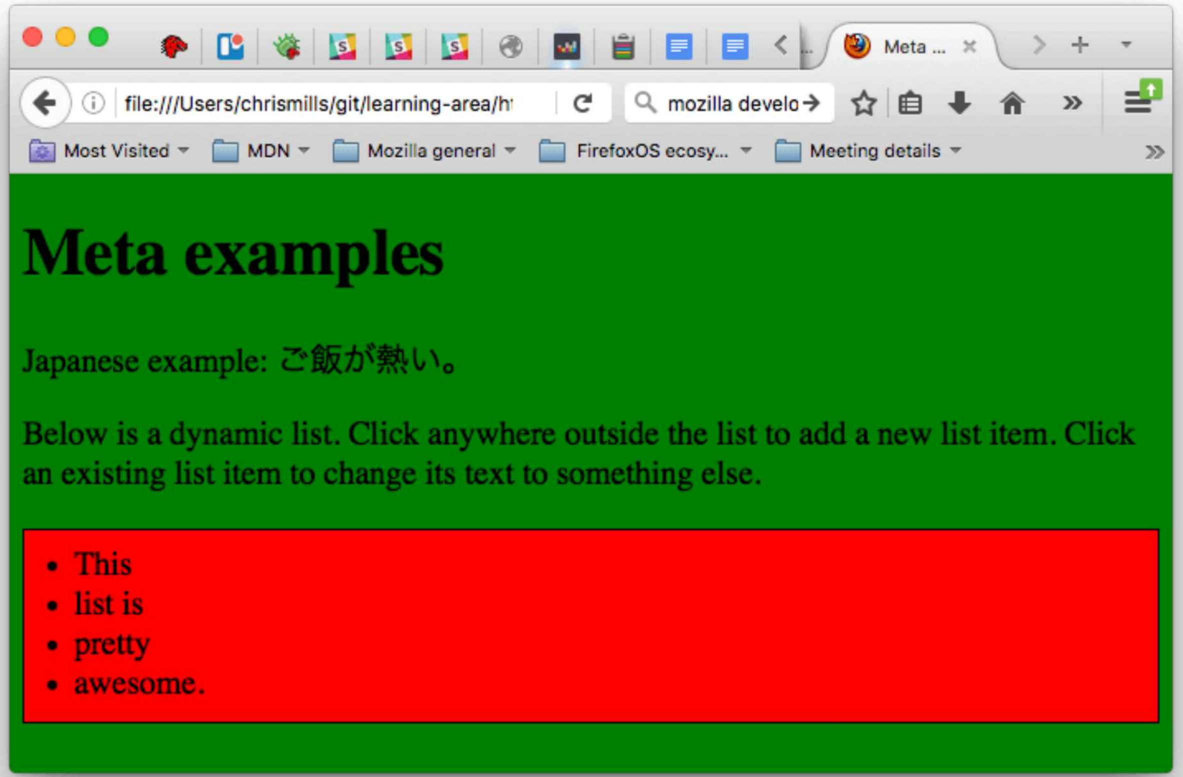
  ```
  <script src="my-js-file.js" defer></script>
  ```

  > **Note**: The `<script>` element may look like an empty element, but it's not, and so needs a closing tag. Instead of pointing to an external script file, you can also choose to put your script inside the `<script>` element.

Active learning: applying CSS and JavaScript to a page

1. To start this active learning, grab a copy of our meta-example.html, script.js and style.cssfiles, and save them on your local computer in the same directory. Make sure they are saved with the correct names and file extensions.
2. Open the HTML file in both your browser, and your text editor.
3. By following the information given above, add `<link>` and `<script>` elements to your HTML, so that your CSS and JavaScript are applied to your HTML.

If done correctly, when you save your HTML and refresh your browser you should be able to see that things have changed:



Meta examples

Japanese example: ご飯が熱い。

Below is a dynamic list. Click anywhere outside the list to add a new list item. Click an existing list item to change its text to something else.

- This
- list is
- pretty
- awesome.

- The JavaScript has added an empty list to the page. Now when you click anywhere on the list, a dialog box will pop up asking you to enter some text for a new list item. When you press the OK button, a new list item will be added to the list containing the text. When you click on an existing list item, a dialog box will pop up allowing you to change the item's text.
- The CSS has caused the background to go green, and the text to become bigger. It has also styled some of the content that the JavaScript has added to the page (the red bar with the black border is the styling the CSS has added to the JS-generated list.)

> **Note**: If you get stuck in this exercise and can't get the CSS/JS to apply, try checking out our css-and-js.html example page.

Setting the primary language of the document

Finally, it's worth mentioning that you can (and really should) set the language of your page. This can be done by adding the lang attribute to the opening HTML tag (as seen in the meta-example.html and shown below.)

```
<html lang="en-US">
```

This is useful in many ways. Your HTML document will be indexed more effectively by search engines if its language is set (allowing it to appear correctly in language-specific results, for example), and it is useful to people with visual impairments using screen readers (for example, the word "six" exists in both French and English, but is pronounced differently.)

You can also set subsections of your document to be recognised as different languages. For example, we could set our Japanese language section to be recognised as Japanese, like so:

```
<p>Japanese example: <span lang="ja">ご飯が熱い。</span>.</p>
```

These codes are defined by the ISO 639-1 standard. You can find more about them in Language tags in HTML and XML.

Summary

That marks the end of our quickfire tour of the HTML head — there's a lot more you can do in here, but an exhaustive tour would be boring and confusing at this stage, and we just wanted to give you an idea of the most common things you'll find in there for now! In the next article we'll be looking at HTML text fundamentals.