

Debugging CSS

← Previous

↑ Overview: Introduction to CSS

Next →

In the final article of this module, we take a look to the principles of debugging CSS, including exploring CSS applied to a page, and other tools which can help you to find errors into the CSS code.

Prerequisites:

Basic computer literacy, [basic software installed](#), basic knowledge of [working with files](#), HTML basics (study [Introduction to HTML](#)), and an idea of How CSS works (study the previous articles in [this module](#).)

Objective:

To learn the basics of how CSS can be debugged.

Just like with debugging HTML, debugging CSS isn't really very scary compared to debugging a lot of other types of code. We'd recommend that you (re)read [debugging isn't scary](#) before continuing.

CSS and debugging

Just like HTML, CSS is *permissive* (read [permissive code](#) before continuing.) In CSS's case, if a declaration is invalid (contains a syntax error, or the browser doesn't support that feature), the browser just ignores it completely and moves on to the next one it finds.

If a selector is invalid, then it doesn't select anything, and the whole rule does nothing — again, the browser just moves on to the next rule.


This is great in a lot of ways — in most cases your content will be shown to your users, even if it isn't styled quite right. But this isn't very helpful when you are trying to debug the problem and you don't even get any kind of error message to help you find it. This is even more of a pain when the content isn't viewable by your users — perhaps a critical style isn't being applied, resulting in the layout going badly wrong?

Fortunately there are some tools available to help you. Let's look at these now.

Inspecting the DOM and CSS

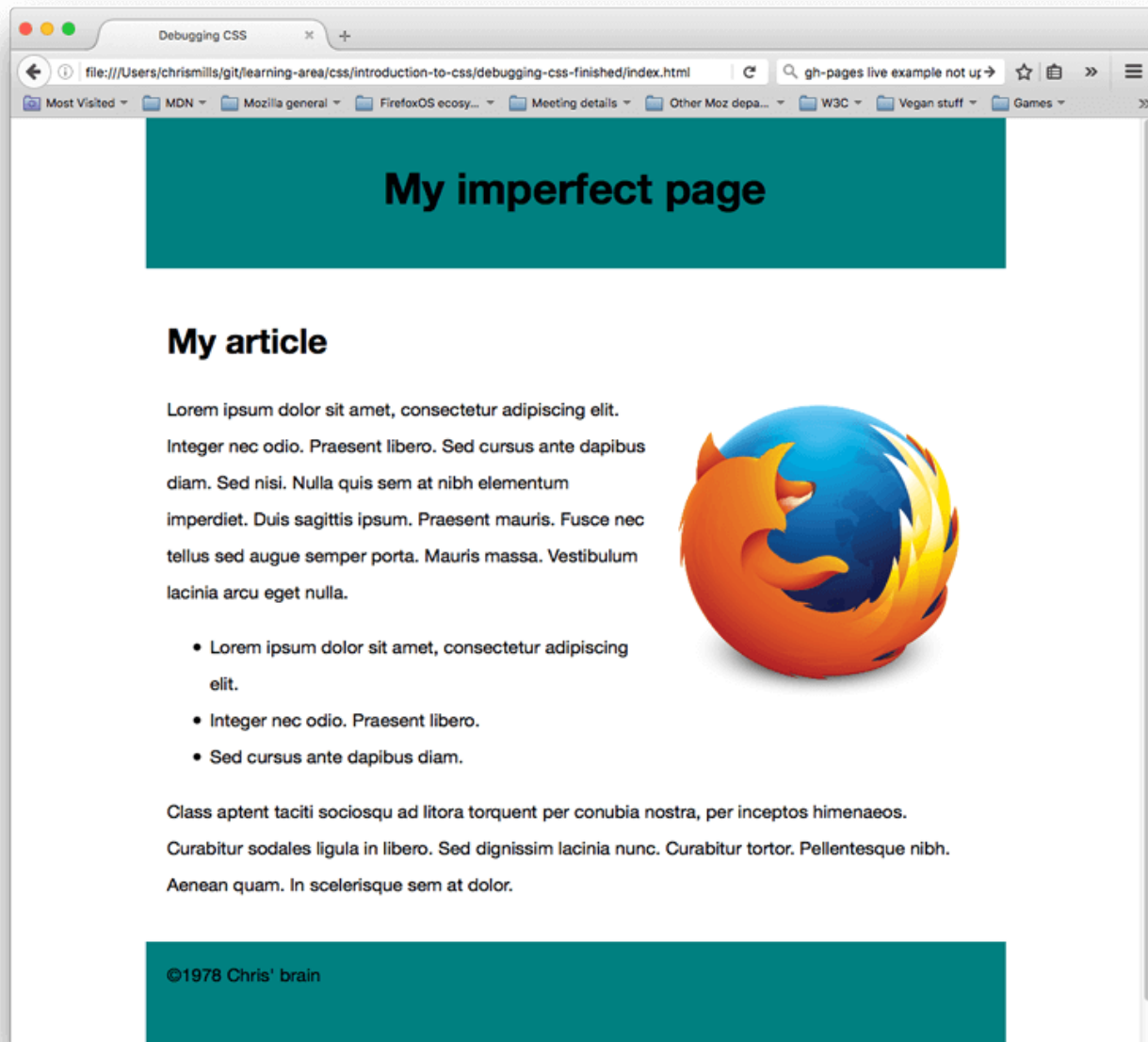
Nowadays, all web browsers provide developer tools made to help you inspect and understand web pages. Among the various tools they provide, there are two that are available in all

browsers: The DOM Inspector and the CSS Editor, which are available in Firefox in the [page inspector tool](#). We already looked at the DOM Inspector in [Debugging HTML](#) and how it can be used to inspect HTML. Here we'll look at this *and* the CSS Editor, and how to use them together to debug CSS problems.

 **Note:** In the following example, we are using [Firefox](#), but all browsers provide the same kind of tools — they might just be available in slightly different places. Read [What are browser developer tools?](#) to find more about accessing them in different browsers.

When going through this example, we'd first like you to open our [CSS debugging example](#) in a new browser tab. If you want to work through and fix the code problems to create a finished version of the example, we'd advise you to make a copy of the [HTML and CSS files](#), and implement the fixes locally.

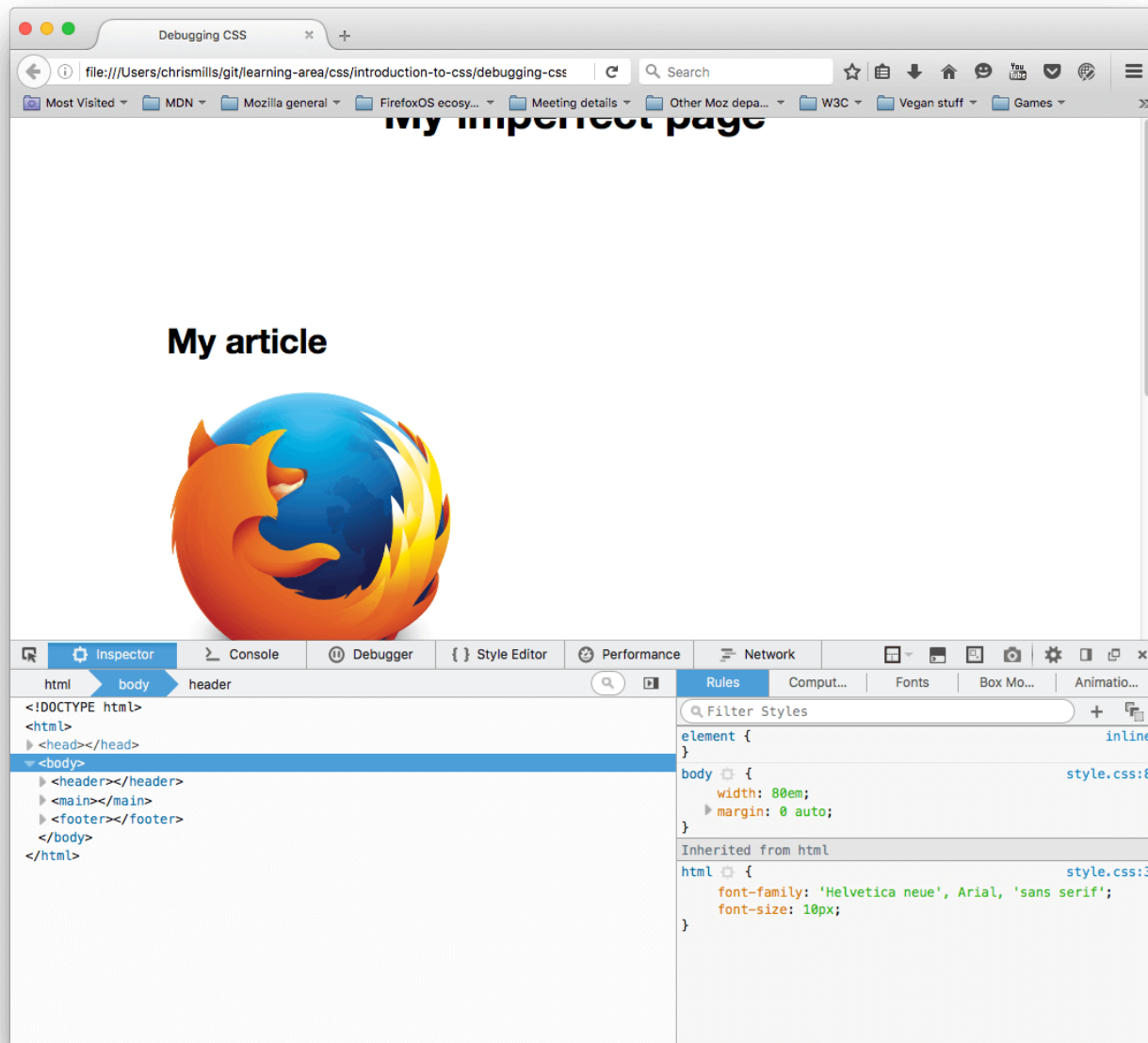
It is meant to be a simple, clear one column web page containing a simple article:



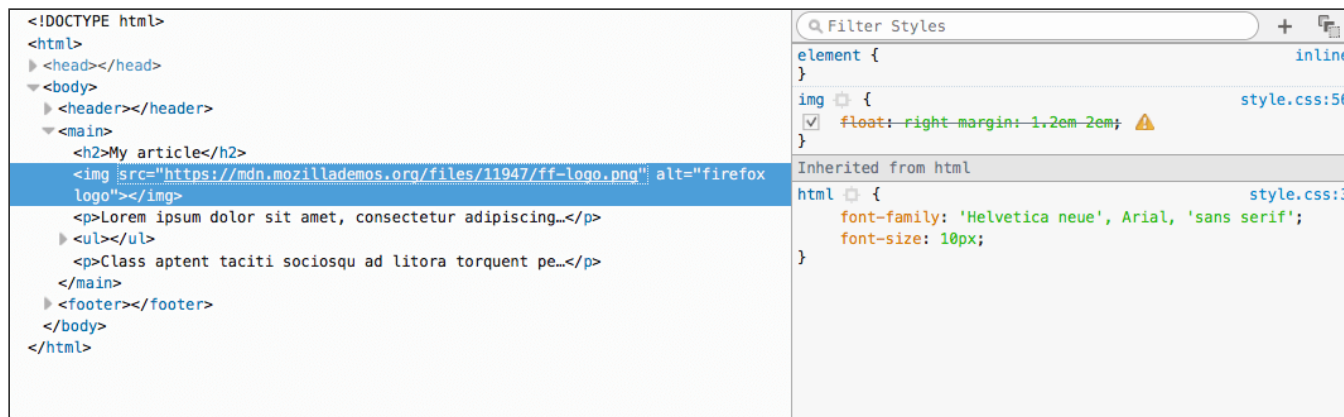
At the moment however it is a bit of a mess:



Let's start investigating the page with the page inspector's features. In Firefox you can open the page inspector using **Cmd / Ctrl + Shift + I** (or by choosing *Tools > Web Developer > Inspector* from the menu), which will give you a set of tools open in a window on the bottom of the browser like so:



If you click on an element inside the DOM Inspector on the left, the CSS editor on the right will update to show all the CSS rules applied to that element. This is really useful, especially as any invalid properties appear with a line through them and a little warning symbol next to them. This will come in handy below!



Note: Each property also has a checkbox next to it, which can be checked/unchecked to allow you to enable/disable your CSS property by property. This is also very useful for figuring out what might be causing an error.

Active learning: find the errors! [🔗](#)

So, with the tool basics outlined, let's try to find our errors. In each case, you should try clicking on the element where the fault is, to see what the applied CSS looks like.

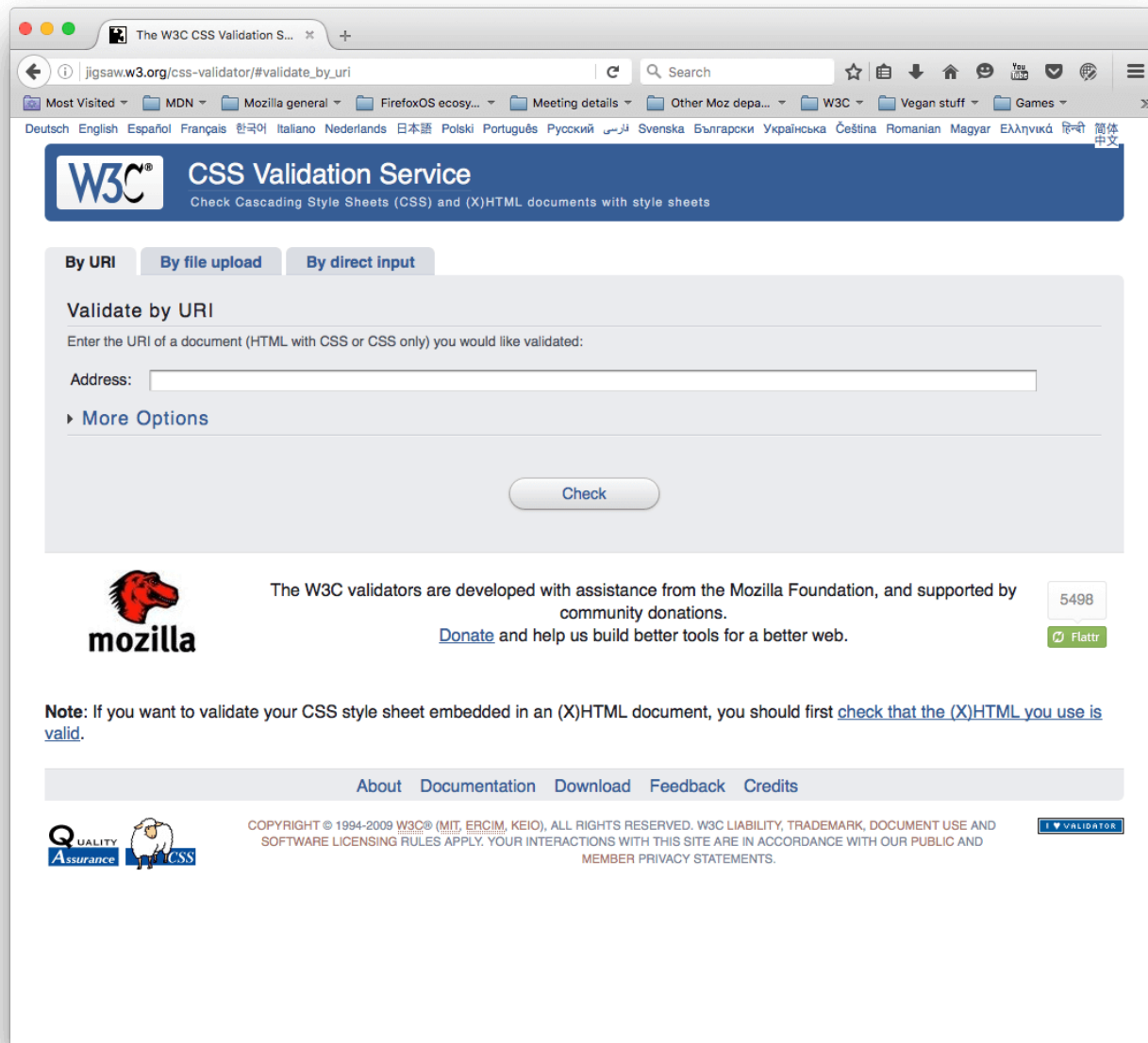
1. The `<header>` and `<footer>` elements are supposed have a background color, but no color appears.
2. The `<h1>` in the header and the `<p>` in the footer are both too high up on the page — this is especially apparent on the `<h1>`, which is nearly off the screen! You could try clicking on the `<h1>` in the developer tools inspector and then unchecking the checkboxes next to the CSS declarations shown in the rules panel to find out which one is causing the problem.
3. The `` is supposed to be floated down and to the right so it sits to the right of some of the text, but it isn't — it's directly above all of the text.

4. The text in the `<main>` content area is far too small — the paragraphs and list items should have a larger font-size applied to it, but it doesn't seem to have been applied to either. Hint: This one is a bit harder, as it is a problem with the selector rather than a property. You may have to study the source code to find this — you can find it in the [Style Editor](#) in Firefox's developer tools.

If you are stuck, you can look at the [fixed source code on Github](#).

CSS validation [🔗](#)

We already looked at the [W3C HTML Validator](#), but they have a [CSS Validator](#) available too. This works in the same way, allowing you to validate CSS at a [particular URL](#), by [uploading a local file](#), or by [direct CSS input](#).



Active learning: Validating our CSS

Let's try feeding our CSS into the validator to see what it spits out.

1. Go to the CSS Validation Service [↗ Validate by direct input](#) view.
2. Copy our error-filled CSS into the text area on the validation service.
3. Press the *Check* button.


You will now be presented with a list of errors. Only two are returned:

Sorry! We found the following errors (2)		
URI : TextArea		
32	header, footer	Property background-colour doesn't exist : teal
58	img	Value Error : float attempt to find a semi-colon before the property name. add it

These are useful messages, especially as they include line numbers and what selector is in action in each case. Let's see how we can interpret these.

- *Property background-colour doesn't exist : teal* — simple; this is telling us that a property doesn't exist, which makes it clear what needs to be done.
- *Value Error : float attempt to find a semi-colon before the property name. add it* — this is telling us that a semi-colon is missing. Looking at the line number of where this is occurring makes this easy to find.

You could argue that this is less useful than the browser developer tools — it doesn't allow you to look at the rendered code while referencing what is wrong, and it is no good for finding instances where the selector is incorrect, or the syntax is correct and you've simply chosen an incorrect value for your design. But we would argue that for a large stylesheet, it is worth running it through this service first to get rid of any basic syntax mistakes, before then relying on the browser developer tools to pinpoint other problems.

 **Note:** [CSS Lint](#) is a similar tool for validating CSS and uncovering errors, which can also present some useful hints and give interesting warnings.

Summary [↗](#)

Well done for completing the last article of the first CSS module! Now that you've come this far, you can have a go at completing our [CSS assessment](#), and then start exploring some other CSS and HTML modules.

[← Previous](#)

[↑ Overview: Introduction to CSS](#)

[Next →](#)

In this module [↗](#)

- [How CSS works](#)
- [CSS syntax](#)
- [Selectors](#)
- [Simple selectors](#)
- [Attribute selectors](#)

- Pseudo-classes and pseudo-elements
 - Combinators and multiple selectors
 - CSS values and units
 - Cascade and inheritance
 - The box model
 - Debugging CSS
 - Fundamental CSS comprehension
-