Week 6 Worksheet

Task 1: Debugging HTML

Let's begin by looking at the kind of errors that can creep into our HTML.

Download the zip file and find `broken.html`. Open it in Visual Studio Code and save it as `fixed.html`.

Inside this file are examples of all of the different types of errors we have seen in the class test over the last few years. You might be able to spot a few immediately.

These are the kind of errors you will find:

- Incorrect attributes
- Spelling errors in tags
- Incorrectly nested tags
- Missing tags
- Content without tags, or content incorrectly placed in relation to tags.

So, to train your eyes to start spotting these:

1. Use the HTML validator at https://validator.w3.org/ to validate the current state of `fixed.html`, and use the errors that are returned as the starting point to continue removing all the errors from this page.
2. Keep validating until all the errors are removed. Save as you make the changes.
3. Once you've validated the page, check that all the links in the HTML work correctly. There should be two links within the text.
4. Once you believe you have removed all the errors, add some relevant content to the `<title>` tag and add a `<meta name="description" content="">` tag. Add some relevant information about the page to the `content` attribute of the description meta tag.
5. Add any tags you think are missing.

Compare and contrast the before and after

Finally, compare the before and after versions - go to http://www.mergely.com/editor and click **file -> import...**. For 'left file' choose `broken.html`, for 'right file' choose `fixed.html`, then click **import**.

You should then see the changes you have made as you fixed the initial version of the file you downloaded.

Task 2: removing inline styles and creating reusable CSS

We are going to start with some badly written code and gradually improve it: our aim will be to make it as easy to read, easy to modify, and easy to reuse, as possible.

1. Open the file `bad.html` from the folder you have downloaded in your editor.
2. Save it as `good.html` and link the HTML file to the `boxes.css` file in the same folder using the `<link href="" rel="stylesheet">` tag in the `<head>` of your html document.
3. View `good.html` in a browser. You should see the red, blue and green boxes.

In the `good.html` file you will find a series of what are called inline styles. It is possible to style HTML elements using these `style` tags but as you can see it makes the code difficult to read and change.

What we would like you to do:

1. Remove the styling code in the inline styles from `good.html` and paste them into the `boxes.css` stylesheet. You should have code that looks like this in the HTML: `<div style="">A red box</div>`. Clean it up by removing the in-line style attribute - `<div>A red box</div>`, etc. Save.
2. Reload `good.html` in your browser. The styling for the boxes should be gone.
3. Rewrite the styling code you've removed from `good.html` into CSS so that you create three classes: `.red-box`, `.green-box` and `.blue-box`.
4. Link the classes in your `boxes.css` CSS file to the correct boxes in the HTML by giving each div in the HTML the correct class attribute.
5. Refresh the HTML page to see if you've rewritten the code correctly. The boxes have returned, now from the external styles in `boxes.css`. If things aren't working, check through your code: do you have the right class names? Are they attached to the HTML `div` tags correctly?

Chaining classes

So we are now at the point where we have three very similar rules for our three boxes. Ideally we want to be able to edit and refine our CSS code in as few places as possible, so we need to 'refactor' our CSS to make it easier to read and shorter.

To do this:

1. Look at the CSS code you currently have. Which properties are set the same for all three boxes? Create a new class called `.box` and copy those repeated properties into it.
2. Delete the code that you don't need from the other classes you've already created (`.red-box`, `.green-box` and `.blue-box`), leaving behind the code which is specific to those classes (hint: it will definitely include the `background-color`).
3. Remove the '`box`' from the class names, so you should now have a stylesheet with a `.box` class, a `.green` class, a `.red` class and a `.blue` class.
4. Edit your HTML so that each of the boxes is styled with the `.box` class and the relevant colour. Do this by changing the `<div class="green-box">` to something like `<div class="green box">`.
5. Save and reload in your browser.
6. Right click on a box and choose 'inspect element'. Try editing the width or height rules for the `.box` CSS class - notice that all three boxes change as they are all using the same class to control their layout.

Our CSS is now more economical. We have avoided having a lot of repeated attributes with the same values in each class. Writing good CSS means being economical, making the most of the cascade, inheritance and specificity. It's not easy.

Why does this work?

HTML allows you to add multiple classes to the same element: all you need to do is separate them with a space. You can have many classes attached to the same element, e.g. `<p class="story-content extra-styling fancy-typeface animated-in">` - they just need to be separated by a single space. Any relevant CSS class will then be added to the list of styles applied to that piece of HTML.

Task 3: Researching and previewing CSS

Continue to work with the `boxes.css` file and the `good.html` file.

Using a new CSS property

1. Using the explanation at https://css-tricks.com/almanac/properties/b/border-radius/ write code to restyle just the red box as a circle.
2. Open the web developer tools in your browser (Chrome or Firefox is best) by right clicking on the red box and choosing 'inspect element'.
3. Find the rule you have just added and try changing the value of the `border-radius` CSS rule. The browser should update as you make changes.
4. Try adding two values separated by a space to your `border-radius` rule - what happens? Why is this?

Another CSS property

1. Using the explanation at https://cssreference.io/property/box-shadow/ add a drop shadow to the green box. Be sure to set a colour for your box shadow, as the browser will default to showing a white box shadow on a white background, which isn't very helpful.
2. Again, use the web developer tools to view and modify the display of the box shadow.

Going even further with CSS properties

Using the transition https://css-tricks.com/almanac/properties/t/transition/ property on the initial styling of the `.blue` box (set your transition to work on `all`), and the transform https://css-tricks.com/almanac/properties/t/transform/ property as part of a `.blue:hover` selector (https://css-tricks.com/almanac/selectors/h/hover/), can you make the blue box spin 180 degrees in 1 second when it is hovered over?

Task 4: Organising CSS

Organising a stylesheet

Have a look at `complex.html` and `complex.css`. This code creates a complex layout using some code you might not be totally familiar with, but that's fine at the moment. The biggest problem with this code is that it's a mess: there is CSS code styling the footer before the code dealing with the header!

This makes the code difficult to read, follow or relate to the HTML it is styling.

We need you to:

1. Reorganise the stylesheet to make it more readable. This may include adding comments to show when we move from one part of the page to another, e.g. from the header to the main page content.
2. We have had some feedback from the client that needs doing:

- The 'test nav link' links in the header should have white text without underlines (You can use `text-decoration:none` in your CSS to do this).
- The links in the text in the main content should be blue.
- The social media links in the footer should be red.

Add the relevant code to the stylesheet to create these styles. Use a combination of a class and element selector (e.g. `.page-footer ul`), related to the correct part of the page, to do this.

You may also want to use the `:hover` pseudo class to add hover styles for the links in the text and in the footer.