HTML & CSS Handbook

Shahzeb Asif

June 17, 2017

References

codecademy.com

I made this while following the codecademy tutorials. Most of the information is at least based on their tutorial. This is not meant to replace their tutorial but instead, function as my own notes.

HTML

Boilerplate

There is some boiler plate in starting an HTML file.

```
<!DOCTYPE html>
<html>
...
</html>
```

The `<!DOCTYPE html>` line tells the browser to treat the file as an HTML file. The `<html> </html>` tags surround the useful data.

Note: The stuff in between `<>` are called tags.

We indicate **comments** by `<!-- comments go here -->`

Tags can be self-closing if you write them as `<tag property="blah" />`.

Head/Body

There are two sections in the HTML files. `<head>` and `<body>`. The `<title>` goes in the `<head>`.

The `<body>` contains all the text and, well, body content.

Content

Paragraphs

We can specify **paragraphs** in the body by using `<p>`.

```
<p> This would be a paragraph. </p>
```

Headings

We can specify **headings** with `<h*>`.

There are six levels of headings: `<h1>` to `<h6>`

Links

We can write **links** by using `<a>` and the `href` value.

```
<a href="shahzebasif.github.io"> My Website </a>
```

The actually location the link points to is contained within the `<a href=...>`. The stuff between the tag is just what the link will be called. The `href` is a **tag attribute**. It gives more information about the tag.

Images

We can embed **images** using the `<img>` tag. The `src` is a tag attribute like `href`.

```
<img src="website.com/img.jpg"/>
```

Putting an image inside a `<a>` will make the image clickable.

Always indent code so it looks clean.

Lists

We can create **numbered lists** with the `<ol>` tag. Each item inside the list is created by the `<li>` tag.

```
<ol>
    <li> Item one. </li>
    <li> Item two. </li>
</ol>
```

We can create **unordered lists** with the `<ul>` tag and the items are still created by the `<li>` tag.

Tables

We can create **tables** with the `<table>` tag. But we can only add rows at first by using the `<tr>` tag. We never really add columns but the table will automatically create columns once we add data with the `<td>` tag.

An example will help (codecademy):

```
<table border="1px">
    <tr>
        <td> One </td>
    <tr>
</table>
```

Tables have their own `<thead>` and `<tbody>` tags as well. These can be used to give extra information about the table.

`<tbody>` just contains the actual content. But `<thead>` can be used to show headings for the table.

```
<thead>
    <tr> Table Caption </tr>
    <tr>
        <th> Column One Heading </th>
        <th> Column Two Heading </th>
    </tr>
</thead>
```

Div

The `<div>` tag is used an awful lot. They are just used to create containers on screen similar to most GUI tools.

Span

`<span>` is used to make small containers for things like change in font for a single word.

CSS

CSS and HTML are typically used together. HTML for the content and CSS to make it look good.

Link from HTML

We need to link the actual CSS file from the HTML file to make things work.

To do this, we place this line below between the HTML file's `<head>` tag.

```
<link type="text/css" rel="stylesheet" href="cssfile.css"/>
```

Comments are indicated by a `/*` and `*/` like C.

HTML Elements

We link content by using the tags in the CSS file. Let's say we want paragraphs to be red instead of the default black. To do this we change `p` in the CSS file.

```
/*file: somecssfile.css*/
p {
    color: red;
}
```

Syntax

Generally, CSS syntax is as follows (example borrowed from codecademy):

```
selector {
    property: value;
}
```

Font

Color

We can select `color` with defined values or hex values such as `color: #FFFFFF`.

Some defined values include `blue`, `red`, `yellow`.

Font Size

Size can be denoted in many forms.

We can use `px` for pixels. Use `pt` if we want. But we can also use a relative measurement called `em`.

`1 em` is equal to the default font size on your computer. It scales linearly.

Font Family

We can select the font family we would like. But if the fonts are not available on someone's computer, they will not be displayed properly.

CSS does have built-in fonts though. `serif`, `sans-serif`, and `cursive` are three built-in fonts.

We can also specify multiple fonts as well and CSS will choose the first available font.

```
p {
    font-family: Verdana, serif
}
```

That will try Verdana and if Verdana cannot be used, then serif will be used.

Font Weight

The font weight decides the bold properties.

```
font-weigth: normal;
```

Element Properties

Those containers and elements we got from HTML can do some cool things with CSS.

Background

We can specify the `background-color` for any element.

Size

We can control the size of any element using their `width` and `height` in CSS.

Borders

We can specify the border around an element as well.

These work on a bunch of elements.

Borrowed from `codecademy.com`

```
border: 2px solid red
```

Note how we specify the color and the style of the border.

Text Decoration

We can also do text decoration.

By setting `text-decoration: none` we get rid of the underlines on links.

Selectors

We've talked about properties but not what can have properties. Pretty much any HTML element can be used as a CSS selector.

You can even do stuff like only select paragraphs inside `div` by using `div p` as a selector. The `div p` selector will overwrite the regular `p` selector in its context.

The wildcard character `*` can be used to apply properties to **all** content.

There are some cool selectors called `class` and `id` which are a little more special.

Display

Display is where we position elements.

The default display elements are **blocks**. The are stacked vertically.

```
selector {
    ...
    display: block;
}
```

Use **inline blocks** to put things horizontally. Set the display to `inline-block`.

Note that both **blocks** keep their dimensions.

Using **inline** puts things horziontally but does not maintain their dimensions. It will shrink them as much as possible.

To turn an element off, use **none**. By using `none`, the CSS will just toss the entire thing until you change the `none`.

Margin

There are three things we can use to create space around a block.

The space around the block is **margin**.

```
margin: auto
```

Padding

The **padding** is around the content.

```
padding: 10px 10px 10px 10px
```

Going out from the content, the order is

```
padding > border > marginv+.

### Floating
We can have floating elements as well. These
are elements that just move to where you
specify but without getting in the way of
other elements.

+v
div {
    ...
    float: right;
}
```

This will make the element float to the right.

`clear` is also useful a lot. It is used to tell elements to stay away from other floating elements.

Position

We use the `position` property to make things `absolute`, `static`.

There's also `relative` positioning and `fixed` positioning.

Class

Classes are just another way to group elements.

If we place a whole bunch of elements into a class. Then they will all be treated with the class properties instead of the regular properties.

```
<p class="temp"> ... </p>

.temp {
    ...
}
```

Pseudo Classes

Pseudo classes are things that aren't explicityly in our HTML document. They work behind the scenes.

An example woul be the color of a link upon hovering. The format for changing pseudo-classes' properties is by using the `:`.

```
a:hover {
    color: blue;
}
```

In that example, `hover` is the pseudo class and `a` is the selector.

As a note, the pseudo classes for links are `link`, `hover`, `visited`.

A useful pseudo-class is `first-child`. It will apply to the first item at the same level. `nth-child(n)` is the same thing but with the n^{th} level.

Note how we use `.temp` in the CSS selector. This is how we refer to class in CSS.

ID

Ids are very similar to classes but whereas classes are used for groups of objects, ids are used for singular objects.

```
<p id="blah"> ... </p>
```

```
#blah {
    ...
}
```

Font Face

To get custom fonts, we can use `@font-face`. This lets use store fonts in our website and use them everywhere.

Here is an example:

```
@font-face {
    font-family: "Example";
    src: url("/fonts/example.eot") format("eot"), ...
}
```

If you want to add a bold font then do the same as above but add this between the brackets:

```
font-weight: bold;
```

There are a whole bunch of formats you can import but the best ones seem to be `.eot` and `.ttf`. These two types cover pretty much all popular browsers.