Intro To CSS

Cascading Style Sheets (CSS)

CSS is a style sheet language used for describing the look and formatting of a document written in a markup language.

It is used to manipulate the way elements appear on a web page, and CSS can interact with both HTML and JavaScript.

Objectives

In this section, we'll accomplish the following:

- Select elements using CSS selectors
- Apply rules to selected elements
- Use rules to arrange elements on the page
- Use rules to style text and images

So far, we put content on our page, but nowhere did we say "use Times New Roman", or "list items should have little black circles next to them". These are the browsers default styles. Terrible, ugly default styles. Let's spiff up this page a bit.

- Copy your `index.html` page with the `cp` command, and call it `bears.html`.
- Delete the contents of the `body` tag.
- Add an `h1` tag, give it the text "Bears - the cuddliest natural predator of humans" .
- Add 3 image tags, () and let's give it a src attribute of "http://placebear.com/500/500".
- Change the "500" to different values for each tag until different pictures appear for each `img` tag.

> There are lots of placeholder image providers, but my jokes will bear-ly make sense if you pick a different one, so just go with me here.

Writing CSS

Go to the `<head>` section of the page, we're going to tell the browser some rules to apply to the text and layout of the page. Open a new tag within `<head>` - call it `<style>`.

What we put between the opening `<style>` and the closing `</style>` tags is called CSS. Usually this ends up in a different file, but we'll get to that later. First, we're going to pick a *selector* to apply a *style* to. It looks something like this:

```
selector {
    rule: value;
    rule: value;
}
```

Styling Text

We're going to add a rule to the `h1` tag, which will apply to *all* `h1`s. Check it:

```
h1 {
    font-family: arial;
}
```

Let's go ahead and add that rule to our `<style>` tag, and reload the page. Looks much better, right?

Probably want to add some other styles, like color. Colors in CSS are expressed in several ways, either by using a hex RGB value - `#ff00ff`, or like this - `rgba(200,200,200,0.5)`. The "a" in "rgba" refers to *alpha*, which is nerd for "transparency". `css h1 { font-family: arial; color: #000088; background-color: rgba(200,255,200,0.5); }` Play with those color values, see if you can make the text brown (for bears.. you know.)

The `h1` in the above code is called the `selector`. This part is what tells CSS how to go and find all of the elements it needs to apply the rules inside of the `{}` braces to. Think of the selector as the search term when you're googling something- CSS can be thought of as *querying* the HTML for matching tags. Sometimes there is only one `h1` tag per document, but sometimes there are many- this rule would apply to *all* of them. That means that using the tag's name as the selector is the *lowest specificity* we can use.

Now that we've done some basic text styling, we can start arranging those bears. First, let's start by giving all of these bears a name. Like names, each tag should be*uniquely identifiable* by it's id, so every id should be unique.

- Add an `id` to each `img` with a name- `id="doctor_von_cuddles"`
- Put a paragraph tag under each image so that it's not just the browser that knows the name of each bear.

Now we can refer to our bears individually within our style tag. The way we do this is to use a `#`.
The `id` selector is the *highest level* of specificity we can use in CSS. We use it when we want to have rules apply to a single element only.

```
#doctor_von_cuddles {
    border: solid #3300ff 10px
}
```

- Give each bear image it's own border.

Arranging Boxes with Floats

Let's see if we can get these bears to line up nicely, one next to the other. Unfortunately bears can be rather territorial, so let's put them in some containers to keep them separated. We're going to wrap these bears in a `div` tag.

```
<div>
```

```
    <img src="http://placebear.com/300/300" id="doctor_cuddles">
    <p>Dr. Cuddles BD.</p>
</div>
```

- Put each bear in a div, and don't forget to close it (or it will escape).

Now that each bear is in it's box, we're going to line up the boxes next to each other. Rather than doing that for each box individually, we're going to treat all the boxes the same, since they're all just boxes with bears in them (even though the bears inside are all unique individuals with their own hopes and dreams).

- Add a `class` attribute, give it the value `bearbox`

You'll note that simply adding the class doesn't change how the page looks. Nor did adding the `div` tag. Here's where the magic happens- now we can refer to all of the boxes at once, and line them all up. Put the following in the `style` tag:

```
.bearbox {
    float:left;
}
```

Holy bears Batman! Each one is in their own little box, all lined up. Beautiful. These simple rules allow you to create complex layouts, by stacking boxes either on top of, or next to each other.

The dot in `.bearbox` is what tells CSS that it's targeting *a class*. We're saying here that we want all of the elements containing bears (`.bearbox`) to behave one way. This class wouldn't affect any other `divs` or anything else on the page, just elements with the class `bearbox` .

The Box Model

Now, bears don't like to be all crushed up against each other- let's give them some room, shall we?

```
.bearbox {
    float:left;
    padding: 10px 10px 10px 10px;
}
```

There, now they've got some breathing room. The property `padding` refers to the space between the content (the bear and it's name) and the outside of the box (the border). Think of the padding as exactly how it's named- like stuffing padding in a box for shipping things safely.

The `border` property we defined earlier can be thought of as the walls of the box. To visualize this:

- Add a `border` property like we have for each individual bear to the `.bearbox` class.

You can only add one of each property to a selector, so just pick your favorite border to apply to all the bears. Now you can see the walls of the boxes the bears are contained within. The `border` is outside of the padding.

We can keep some space between each box's border and the next box by adding a property called `margin` . It works the same as `padding` , except that it refers to the space on the outside of the `border` .

- Add a `margin` to the `.bearbox` selector.

These tools may seem simple, but they are very powerful. With them we can create complex layouts and reusable code.

Reading

Getting Started Guide to CSS The Box Model
Floats

Exercises

Work through these exercises, then do the Floats Exercise

Challenge Creating Layouts From Mockups

Create a page for the following wireframes:

- https://wireframe.cc/GuRoUr
- https://wireframe.cc/0ftEEJ

Don't worry about them being pixel perfect, just try to produce a page that achieves the same layout.

If you finish with all of the above exercises, try these stretch goals:

- CSS Nav Challenge
- CSS Card Flip

Practice

- Basic HTML/CSS Practice
- CSS Reading