Rev. 08Aug2019

Exercise 36

IMAGES WITH CSS

CSS is amazing. In fact, if you really want to be shocked by how powerful it is, point your browser to `http://csszengarden.com`. There, you'll see the *same HTML* with dozens of different CSS files applied to them. Each one is totally unique.

In this exercise, we're going to do something much less ambitious but, I think, very eye-catching. Here's the first image:

And here's the same image, but with me hovering over the Poppy:

Using those tags also makes it easier to target them as CSS selectors: no classes needed

Open `index.html` and notice the use of two new elements: `<figure>` and `<figcaption>`. These don't *do* anything special. In fact, I could have replaced `<figure>` with something like `<div class="image">` and `<figcaption>` with `<p class="image-caption">`. The reason I use the new tags is that I want it to be obvious to anyone looking at the code what the *purpose* of the HTML element is. That's known as *semantic HTML* — where we choose tags to express *purpose* and rely on CSS to provide us with *presentation*. This exercise is going to concentrate solely on CSS.

- First thing: I want to center the contents of the `<body>` element horizontally. We can do that with this:

```
body {
  text-align: center;
}
```

- Next, let's change `display` to `inline-block` to get the `<figure>`s to sit next to each other:

```
figure {
  display: inline-block;
}
```

I want the `<figcaption>`s to sit *on top of* the `<img>` element. For that, we need to change the `position` property of both the `<figure>` elements and the `<figcaption>` elements. You can do some really interesting things to your web page by altering the value of the `position` property. Exhaustively exploring all the possibilities is outside the scope of our preschool, but this will give you an example of the kind of thing that's possible.

- Normally, when we place two HTML elements beside each other, they display either next to each other (if `display` is set to `inline-block`) or above/below each other (if `display` is set to `block`). Here's a way to get one element to sit "on top of" another element:

```
figure {
  display: inline-block;
  position: relative;
}
```

- Now for `<figcaption>`:

```
figcaption {
  position: absolute;
}
```

Using those tags also makes it easier to target them as CSS selectors: no classes needed

The reason that I set `<figcaption>`'s `position` to `absolute` is that I can now specify exactly where I want it relative to — yeah, relative to what? To the ***nearest positioned ancestor***.

Think about that: an ancestor has to be a parent (or grandparent or great-grandparent...) element. `<img>` isn't an ancestor; it's a sibling. But since `<figure>` wraps `<figcaption>`, it *is* an ancestor. But! Without explicitly positioning `<figure>`, it still does not qualify as the nearest ***positioned*** ancestor.

- But since we *did* position it (as `relative`), we can now specify where we want `<figcaption>` to appear relative to its nearest positioned ancestor: `<figure>`. So let's add that to figcaption now:

In other words, place <figcaption> 8px up from the bottom — and 20px in from the left — of <figure>

```
figcaption {
  position: absolute;
  bottom: 8px;
  left: 20px;
}
```

This is a very powerful technique.

- What about the nifty grayscale to color effect? That's actually trivially easy. You can assign different *filters* to images. The one I chose causes the image to display in grayscale:

```
img {
  -webkit-filter: grayscale(100%)
  filter: grayscale(100%)
}
```

- When someone hovers over the image, we remove the filter:

```
img:hover {
  filter: none;
}
```

There are a few other small CSS tweaks I made (*e.g.* changing the `font-family` and `color` for `.figcaption`) that you can see in answer.css.

- You can learn about the different image filters at: `https://www.w3schools.com/cssref/css3_pr_filter.asp`.