MAKING YOUR TWINE GAME LOOK AWESOME WITH CSS

The following handout explains how to use CSS to change the appearance of your Twine game. All these instructions are based on the SugarCube story format. Before beginning, make sure that your Twine game is set up for the SugarCube format. To do so, click on the name of your story in its main "story map" view. Select "Change Story Format" and check the box next to "SugarCube." Note: this handout is for my students, and assumes some familiarity with HTML and CSS.

Remembering the basics

One of the things I love about Twine is that it publishes to standard web formats. It uses HTML for content, CSS for presentation, and JavaScript for anything programming-related. Because it uses these standard formats, your game can be played on basically every computer, phone, tablet, etc.

By default, Twine games in SugarCube look pretty awful. That should motivate you to put your own personal visual stamp on your game. Good news: since you already know some CSS, this is pretty easy.

To edit a Twine story's CSS, click on the name your story from its main "story map" screen, then click on "Edit Story Stylesheet." This will load a screen that is just a plain old (empty) CSS file.

Changing default settings

Since Twine games are just regular HTML files, it shouldn't come as much of a surprise that you change the default appearance of most things in your game by styling the `body` element (the highest-level element in the HTML "document tree.")

Adding the following code, for instance, would change the background color to white, make the default color dark grey, change the default font to Futura, and make the default font size a little bigger:

```
body {
    background-color: white;
    color: darkgrey;
    font-family: Futura,Impact,Helvetica,sans-serif;
    font-size: 125%;
}
```

Again unsurprisingly, changing the color and behavior of links is achieved by styling the `a` element, HTML's element for links:

```
a {
    color: red;
}
a:hover {
    color: crimson;
    text-decoration: underline;
}
```

Changing the appearance of individual passages

Okay, so that changes the default settings for your whole game. But what if you want a particular passage to have its own background color, or its own special font?

Again, Twine's the best, and it makes it pretty easy. The first thing you need to do is open the passage you want to do something special with. Right underneath the passage name is an option that says `+tag`. Click on it to add a tag of your choosing. For instance, let's say we want to add a tag called "hooray" for a particularly happy passage. Write in the word `hooray` and click on the checkmark.

Now we need to create a set of instructions for your web browser to follow when it displays this passage. To do so, we go back to to the "Edit Story Stylesheet" page and create a CSS class with a name matching the "tag" we inserted above. In this case, we make a class called `hooray` (remember, in CSS, class names have to be preceded by a period):

```
.hooray {
    background-color: pink;
    color: cornflowerblue;
    font-size: 200%;
}
```

Now when your player gets to this passage, they will be met by a pink background and big blue text.

Hiding the sidebar

If, like me, you're not fond of the default sidebar in SugarCube games, you can hide it pretty easily. Just enter the following code into your story's CSS file.

```
#ui-bar {
    display: none;
}
```

(SugarCube puts the sidebar into an HTML `div` with the `id` of `ui-bar`. This CSS instruction just tells your browser not to display that `div`.)

Since by default your story makes room for the sidebar, you will also want to add the following code to style your story's `body` element:

```
body {
    margin-left: 3.5em;
}
```

Adding your own HTML and styling it with CSS

As you can see, Twine does quite a bit of hand-holding. You don't need to enter your own HTML code for paragraphs, for instance: it puts in `<p>`s for you. But if you're not happy with the way Twine is inserting HTML in your game, you can enter your own HTML code right into a passage.

For instance, you might want to have a spooky effect where text disappears if a user mouses over it. You could do that by sticking your own HTML `div` element into a Twine passage and styling it in CSS.

For instance, you might have a passage that says the following.

```
There is a spooky UFO in the sky.
<div class="aliens">When you look at it, it disappears.</div>
But you're sure it's there.
```

You could then add a few lines to your story's CSS file in which you style this `div`'s `aliens` class to make it disappear when the user's mouse hovers over the element.

```
.aliens:hover {
    opacity: 0;
    transition: 1s all ease;
}
```

This HTML and CSS code would then work together to make the words "When you look at it, it disappears" disappear when the user's cursor hovers over them, by gradually transitioning the opacity of the `aliens div` to zero over a span of one second.

You now know everything you need to know to create a fully customized visual experience in your Twine game. Go nuts!