Painless CSS

Build Beautiful, Intuitive Websites

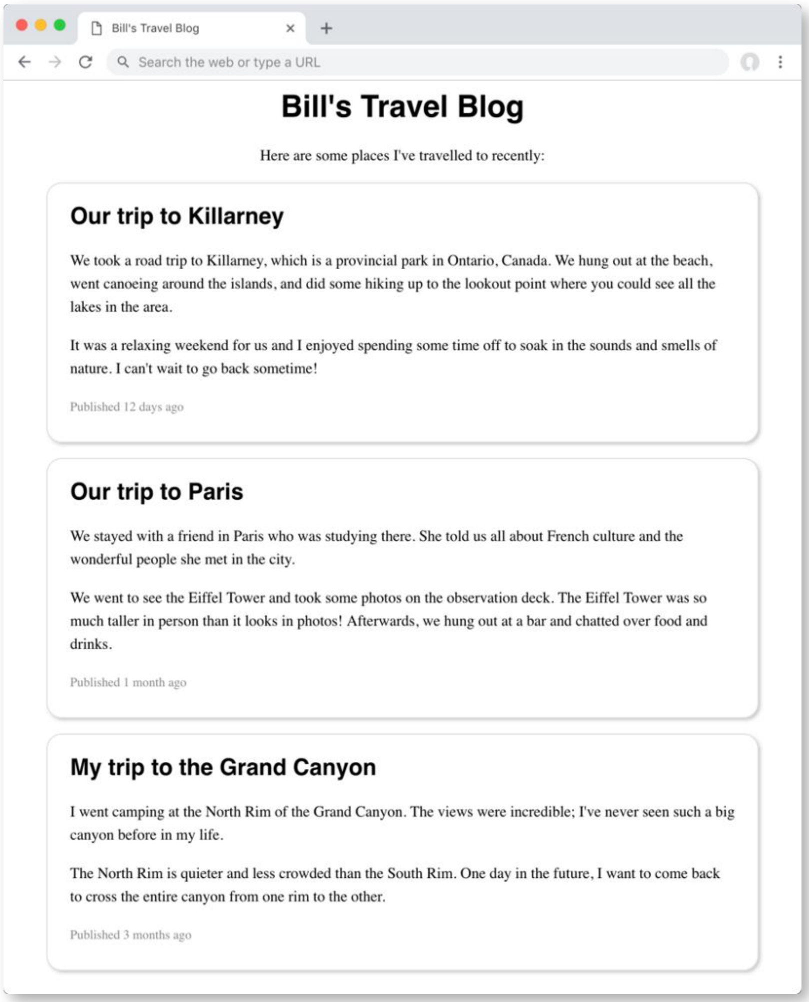
BILL MEI

Chapter 6:
CSS Parsing

In *Chapter 2: CSS Styles*, we talked about how the `style` attribute can change the design of any HTML tag. But what if we want to style multiple tags at once?

CSS Stylesheets

Let's start a travel blog. Here's what it looks like:



Bill's Travel Blog

Here are some places I've travelled to recently:

Our trip to Killarney

We took a road trip to Killarney, which is a provincial park in Ontario, Canada. We hung out at the beach, went canoeing around the islands, and did some hiking up to the lookout point where you could see all the lakes in the area.

It was a relaxing weekend for us and I enjoyed spending some time off to soak in the sounds and smells of nature. I can't wait to go back sometime!

Published 12 days ago

Our trip to Paris

We stayed with a friend in Paris who was studying there. She told us all about French culture and the wonderful people she met in the city.

We went to see the Eiffel Tower and took some photos on the observation deck. The Eiffel Tower was so much taller in person than it looks in photos! Afterwards, we hung out at a bar and chatted over food and drinks.

Published 1 month ago

My trip to the Grand Canyon

I went camping at the North Rim of the Grand Canyon. The views were incredible; I've never seen such a big canyon before in my life.

The North Rim is quieter and less crowded than the South Rim. One day in the future, I want to come back to cross the entire canyon from one rim to the other.

Published 3 months ago

And here is the code (extremely truncated, I encourage you to read the full source in the companion repo):

`Source Code: chapter06/travel-blog-inline.html`

```
<html>
  <body style="max-width: 700px; margin-left: auto; margin-right:
auto; padding: 1em;">
    <header>
      <h1 style="font-family: sans-serif; text-align: center;">Bil
l's Travel Blog</h1>
      <p style="line-height: 1.5em; text-align: center;">Here are
some places I've travelled to recently:</p>
```

That's a lot of styles! To change the `line-height` on every paragraph, we must write `style="line-height: 1.5em;"` on every `<p>` tag. We want the headers to be sans-serif instead of serif, so we need to write `style="font-family: sans-serif;"` on every `<h1>` and `<h2>` tag.

Writing everything out this way is also known as writing ***inline styles*** (nothing to do with `display: inline`, by the way). This is very tedious and repetitive, and you usually don't do this in a real website.[35] This also makes the HTML hard to read, hard to debug, and hard to change.

Is there a simple way to say "apply `line-height: 1.5em` to every paragraph" once, and have the browser do it automatically for us? Yes! This can be achieved using ***CSS Stylesheets*** instead of writing each style inline.

CSS Stylesheets are a collection of ***CSS Rules***, and each CSS Rule is made up of ***CSS Selectors*** and ***CSS Declarations***.

CSS Selectors are called "selectors" because we want to say, "apply our style to the *selected* elements on the page." To select an element, we need to describe what that element looks like. In our case, we want to select all paragraphs, so we use the `p` selector:

```
p
```

[35] Surprisingly, many websites today actually do just make everything use inline styles and forgo CSS stylesheets entirely! I have a discussion about this in *Beyond Cascading*.

Yup. That's it. To use this `p` selector, we use curly braces `{}` to define a block, and put our CSS declarations inside the block:

```
p {line-height: 1.5em;}
```

This is saying: "apply `line-height: 1.5em` to the *selected* paragraphs". Similarly, for our `h2` tags:

```
h2 {font-family: sans-serif;}
```

We also have some styling on each `<article>` tag to add a border, some padding, and some shadows. To style all these `<article>` tags at once, we can do:

```
article {border: 1px solid #dddddd; border-radius: 1em; margin-bot
tom: 1em; padding-left: 1.5em; padding-right: 1.5em; padding-botto
m: 0.5em; box-shadow: 2px 2px 4px #cccccc;}
```

To make this more readable, we can add newlines and indentation:

```
article {
  border: 1px solid #dddddd;
  border-radius: 1em;
  margin-bottom: 1em;
  padding-left: 1.5em;
  padding-right: 1.5em;
  padding-bottom: 0.5em;
  box-shadow: 2px 2px 4px #cccccc;
}
```

This entire chunk of code is called a ***CSS Ruleset*** or ***CSS Rule***.

How do we apply our CSS rules to the document? We do this using the `<style>` tag.

```
<style>
  article {
    border: 1px solid #dddddd;
    border-radius: 1em;
    margin-bottom: 1em;
    padding-left: 1.5em;
    padding-right: 1.5em;
    padding-bottom: 0.5em;
    box-shadow: 2px 2px 4px #cccccc;
  }

  h2 {
    font-family: sans-serif;
  }
  p {
    line-height: 1.5em;
  }
</style>
```

The collection of all these rules inside this `<style>` tag is the ***CSS Stylesheet***.

The `<style>` tag is usually included in the `<head>` section of the document, which means our document now looks like this:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      article {
        border: 1px solid #dddddd;
        border-radius: 1em;
        margin-bottom: 1em;
        padding-left: 1.5em;
        padding-right: 1.5em;
        padding-bottom: 0.5em;
        box-shadow: 2px 2px 4px #cccccc;
      }

      h2 {
        font-family: sans-serif;
      }

      p {
        line-height: 1.5em;
      }
    </style>
  </head>
  <body>
    <header>
      <h1 style="text-align: center;">Bill's Travel Blog</h1>
      <p style="text-align: center;">Here are some places I've tra
velled to recently:</p>
    </header>
    <article>
      <h2>Our trip to Killarney</h2>
...etc.
```

What's going on here? We are writing CSS inside an HTML file?

Loading CSS

CSS and JS are just tags themselves: `<style>` for CSS and `<script>` for JS.

During the parsing step (what we talked about in *Chapter 3: Rendering*), once the browser sees an opening `<style>` tag, it knows that this means "Everything from now on must be CSS. I'll assume it's CSS until I see a closing `</style>` tag, and then I'll go back to normal HTML parsing."

Comments work the same way. In HTML, a comment starts with `<!--` and end with `-->`. Comments are ignored by the browser when it renders HTML, so you can use them to annotate your code without having your notes visible to the user:

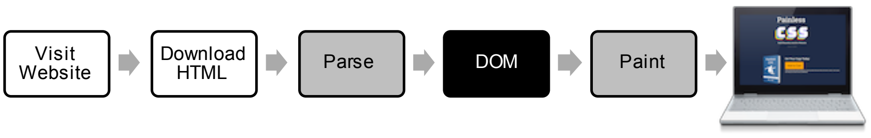
```
<!-- everything in here is ignored -->
```

CSS Comments start with `/*` and end with `*/`. Anything between `/*` and `*/` is ignored by the browser. Here is a recap of all the terminology:

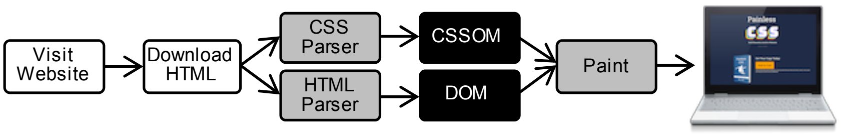
```
/* This is a comment. */

/* The following chunk of code is called a "CSS Ruleset" */
selector {
  /* Everything inside the curly braces is called a "block" */
  property: value; /* This line is called a "declaration" */
}
```

If the CSS is not parsed by the HTML parser, where does it go? To the CSS parser! After the CSS is parsed, it is transformed into the ***CSS Object Model***, or CSSOM for short. Just as the DOM is an internal browser representation of your HTML content, the CSSOM is an internal browser representation of your CSS content. Recall the steps needed to render a webpage:



We can update this diagram to reflect what happens when we parse CSS in addition to HTML:



Like the DOM, the CSSOM is also a hierarchical tree that is constructed from text inside your `<style>` tag. Also like the DOM, once the CSSOM is constructed, the original CSS text file is thrown away because it's not needed anymore.

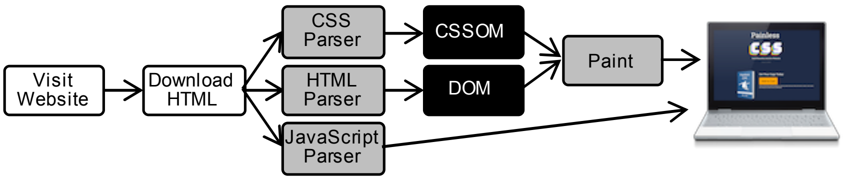
The CSSOM is determined not by a family tree, but instead by the ***CSS Cascading Rules***, which has nothing to do order of the elements. In a later chapter we'll investigate in more detail how the CSSOM is constructed using the cascading rules.

Loading JS

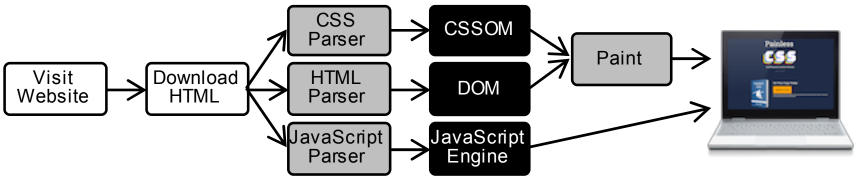
JavaScript is used for handling user interaction, and more complex things like animations and dynamic content.

JavaScript lives inside `<script>` tags, and when the browser sees an opening `<script>` tag, it assumes "Everything from now on must be JavaScript" until it sees a closing `</script>` tag.

Just like CSS and HTML, this content inside the `<script>` tag is handled by a separate parser, the JavaScript parser:



JavaScript doesn't have any "Object Model" or any trees, because it's a general-purpose programming language, so the result of the JS parser are JavaScript instructions that are executed line by line by the JavaScript engine, according to that language's rules:



Although you can put the `<script>` tag inside the `<head>` just like with `<style>`, the best practice is to put this tag right at the very end before the closing `</body>` tag for performance reasons.[36]

Separating HTML, CSS, and JS

Here's what a document would look like with HTML, CSS, and JavaScript:

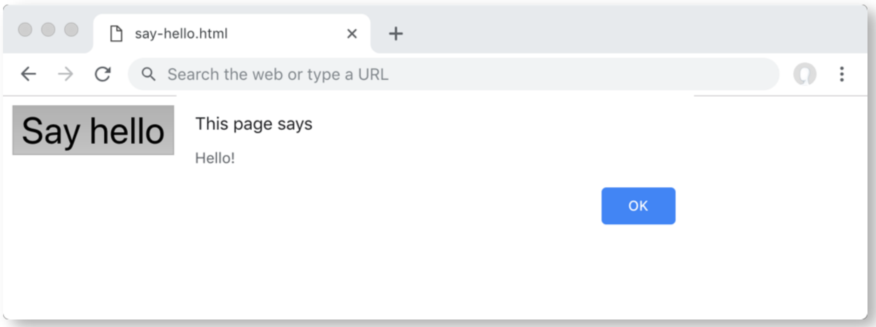
`Source code: chapter06/say-hello.html`

```html
<!DOCTYPE html>
<html>
  <head>
    <style>
      button {
        font-size: 200%;
      }
    </style>
  </head>
  <body>
    <button onclick="hello()">Say hello</button>
    <script>function hello() { alert('Hello!') }</script>
  </body>
</html>
```

This is a page that displays a "Hello!" popup when you click on the button:

[36] Why is it better to put JavaScript at the bottom of the file? As you can see from the diagram, JavaScript does not go through a "Paint" operation, so it's ok if we don't get to it until later. Since CSS and HTML are required for *anything* to show up on the screen, we prioritize parsing it first by putting them nearer to the top of the file, that way the user can see *something* on the screen while they wait for the JavaScript to load, decreasing the perceived load time of the page.

This technique is also known as avoiding *render blocking*, so named because we are trying not to block the "render" operation with things that are not necessary for rendering the page.



In a real webpage though, we'll have a lot more CSS and JavaScript on the page, and instead of putting everything in the same file, it's easier to separate them out into different files to keep things organized. Instead of using a `<style>` tag for CSS, we copy our CSS into its own `say-hello.css` file and load it using the `href` attribute on a `<link>` tag. We also copy our JS into its own `say-hello.js` file and load it using the `src` attribute on the `<script>` tag. Our website now looks like this:

`Source code: chapter06/say-hello-separated.html`

```html
<!DOCTYPE html>
<html>
  <head>
    <link href="say-hello.css" rel="stylesheet" />
  </head>
  <body>
    <button onclick="hello()">Say hello</button>
    <script src="say-hello.js"></script>
  </body>
</html>
```

`Source code: chapter06/say-hello.css`

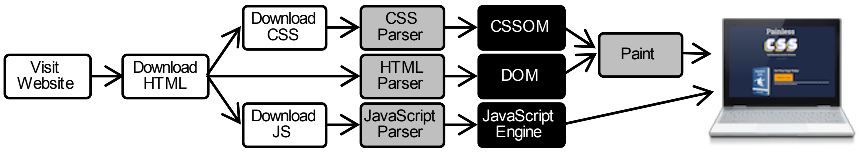
```css
button {
  font-size: 200%;
}
```

`Source code: chapter06/say-hello.js`

```js
function hello() { alert('Hello!') }
```

When we use `<link>` instead of `<style>`, we are telling the browser that our CSS lives in some other file, and it needs to download the other file and parse it. This separate file is also known as a ***stylesheet***, and it's where the "SS" in "CSS" comes from (Cascading *Stylesheets*).

Now when you request this website from your browser, your computer downloads the HTML file from the server, but then after parsing the HTML file it realizes that it needs the extra CSS and JS files too. Then, your browser will download these two additional files from the server as well:



Armed with this knowledge, let's see if we can clean up our travel blog and make it easier to understand and more maintainable.