Chapter 1: Getting started with CSS

Version	Release Date
1	1996-12-17
2	1998-05-12
3	2015-10-13

Section 1.1: External Stylesheet

An external CSS stylesheet can be applied to any number of HTML documents by placing a `<link>` element in each HTML document.

The attribute `rel` of the `<link>` tag has to be set to `"stylesheet"`, and the `href` attribute to the relative or absolute path to the stylesheet. While using relative URL paths is generally considered good practice, absolute paths can be used, too. In HTML5 the `type` attribute can be omitted.

It is recommended that the `<link>` tag be placed in the HTML file's `<head>` tag so that the styles are loaded before the elements they style. Otherwise, users will see a flash of unstyled content.

Example

hello-world.html

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <link rel="stylesheet" type="text/css" href="style.css">
    </head>
    <body>
        <h1>Hello world!</h1>
        <p>I ♥ CSS</p>
    </body>
</html>
```

style.css

```
h1 {
    color: green;
    text-decoration: underline;
}
p {
    font-size: 25px;
    font-family: 'Trebuchet MS', sans-serif;
}
```

Make sure you include the correct path to your CSS file in the href. If the CSS file is in the same folder as your HTML file then no path is required (like the example above) but if it's saved in a folder, then specify it like this `href="foldername/style.css"`.

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

External stylesheets are considered the best way to handle your CSS. There's a very simple reason for this: when you're managing a site of, say, 100 pages, all controlled by a single stylesheet, and you want to change your link

colors from blue to green, it's a lot easier to make the change in your CSS file and let the changes "cascade" throughout all 100 pages than it is to go into 100 separate pages and make the same change 100 times. Again, if you want to completely change the look of your website, you only need to update this one file.

You can load as many CSS files in your HTML page as needed.

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

CSS rules are applied with some basic rules, and order does matter. For example, if you have a main.css file with some code in it:

```
p.green { color: #00FF00; }
```

All your paragraphs with the 'green' class will be written in light green, but you can override this with another .css file just by including it *after* main.css. You can have override.css with the following code follow main.css, for example:

```
p.green { color: #006600; }
```

Now all your paragraphs with the 'green' class will be written in darker green rather than light green.

Other principles apply, such as the '!important' rule, specificity, and inheritance.

When someone first visits your website, their browser downloads the HTML of the current page plus the linked CSS file. Then when they navigate to another page, their browser only needs to download the HTML of that page; the CSS file is cached, so it does not need to be downloaded again. Since browsers cache the external stylesheet, your pages load faster.

Section 1.2: Internal Styles

CSS enclosed in **<style></style>** tags within an HTML document functions like an external stylesheet, except that it lives in the HTML document it styles instead of in a separate file, and therefore can only be applied to the document in which it lives. Note that this element *must* be inside the **<head>** element for HTML validation (though it will work in all current browsers if placed in `body`).

```
<head>
    <style>
        h1 {
            color: green;
            text-decoration: underline;
        }
        p {
            font-size: 25px;
            font-family: 'Trebuchet MS', sans-serif;
        }
    </style>
</head>
<body>
    <h1>Hello world!</h1>
    <p>I ♥ CSS</p>
</body>
```

Section 1.3: CSS @import rule (one of CSS at-rule)

The @import CSS at-rule is used to import style rules from other style sheets. These rules must precede all other types of rules, except @charset rules; as it is not a nested statement, @import cannot be used inside conditional group at-rules. `@import`.

How to use @import

You can use @import rule in following ways:

A. With internal style tag

```
<style>
 @import url('/css/styles.css');
</style>
```

B. With external stylesheet

The following line imports a CSS file named `additional-styles.css` in the root directory into the CSS file in which it appears:

```
@import '/additional-styles.css';
```

Importing external CSS is also possible. A common use case are font files.

```
@import 'https://fonts.googleapis.com/css?family=Lato';
```

An optional second argument to `@import` rule is a list of media queries:

```
@import '/print-styles.css' print;
@import url('landscape.css') screen and (orientation:landscape);
```

Section 1.4: Inline Styles

Use inline styles to apply styling to a specific element. Note that this is ***not*** optimal. Placing style rules in a **`<style>`** tag or external CSS file is encouraged in order to maintain a distinction between content and presentation.

Inline styles override any CSS in a **`<style>`** tag or external style sheet. While this can be useful in some circumstances, this fact more often than not reduces a project's maintainability.

The styles in the following example apply directly to the elements to which they are attached.

```
<h1 style="color: green; text-decoration: underline;">Hello world!</h1>
<p style="font-size: 25px; font-family: 'Trebuchet MS';">I ♥ CSS</p>
```

Inline styles are generally the safest way to ensure rendering compatibility across various email clients, programs and devices, but can be time-consuming to write and a bit challenging to manage.

Section 1.5: Changing CSS with JavaScript

Pure JavaScript

It's possible to add, remove or change CSS property values with JavaScript through an element's `style` property.

```
var el = document.getElementById("element");
el.style.opacity = 0.5;
el.style.fontFamily = 'sans-serif';
```

Note that style properties are named in lower camel case style. In the example you see that the css property `font-family` becomes `fontFamily` in javascript.

As an alternative to working directly on elements, you can create a **`<style>`** or **`<link>`** element in JavaScript and append it to the **`<body>`** or **`<head>`** of the HTML document.

jQuery

Modifying CSS properties with jQuery is even simpler.

```
$('#element').css('margin', '5px');
```

If you need to change more than one style rule:

```
$('#element').css({
    margin: "5px",
    padding: "10px",
    color: "black"
});
```

jQuery includes two ways to change css rules that have hyphens in them (i.e. `font-size`). You can put them in quotes or camel-case the style rule name.

```
$('.example-class').css({
    "background-color": "blue",
    fontSize: "10px"
});
```

See also

- JavaScript documentation – Reading and Changing CSS Style.
- jQuery documentation – CSS Manipulation

Section 1.6: Styling Lists with CSS

There are three different properties for styling list-items: `list-style-type`, `list-style-image`, and `list-style-position`, which should be declared in that order. The default values are disc, outside, and none, respectively. Each property can be declared separately, or using the `list-style` shorthand property.

`list-style-type` defines the shape or type of bullet point used for each list-item.

Some of the acceptable values for `list-style-type`:

- disc
- circle
- square
- decimal
- lower-roman
- upper-roman
- none

(For an exhaustive list, see the W3C specification wiki)

To use square bullet points for each list-item, for example, you would use the following property-value pair:

```
li {
    list-style-type: square;
}
```

The `list-style-image` property determines whether the list-item icon is set with an image, and accepts a value of `none` or a URL that points to an image.

```
li {
    list-style-image: url(images/bullet.png);
}
```

The `list-style-position` property defines where to position the list-item marker, and it accepts one of two values: "inside" or "outside".

```
li {
  list-style-position: inside;
}
```