# Cascading Style Sheets (CSS): Making Web Pages Beautiful

If HTML is the structure of a web page, think of CSS as its wardrobe. CSS, short for Cascading Style Sheets, is a powerful language used to control the presentation and layout of web documents. It allows web developers to style HTML elements, making web pages visually appealing and user-friendly. In this introduction to CSS, we'll explore how it works and provide some examples to get you started.

## How CSS Works

CSS operates on the principle of selecting HTML elements and applying styles to them. These styles define how elements look and behave on a web page. Here's a breakdown of how CSS works:

**Selectors:** Selectors are patterns that target specific HTML elements you want to style. You can select elements by their tag names (e.g., p, h1, div), classes (e.g., .header, .button), IDs (e.g., #main-content), attributes, and more.

**Properties:** Properties are the aspects of an element you want to style, such as its color, font size, margin, padding, or background. Each property has a name and a value.

**Values:** Values are the settings you assign to properties. For example, you might set the color property to "red" or the font-size property to "16px."

**Declaration Blocks:** To apply styles, you group one or more property-value pairs inside a set of curly braces {}. This combination of properties and values within curly braces is known as a declaration block.

**CSS Rules:** A CSS rule consists of a selector and a declaration block. It defines which elements to style and how to style them. Multiple rules can be combined in a CSS file or directly within an HTML document.

**Cascading:** The "C" in CSS stands for "Cascading," which means that when multiple conflicting styles are applied to an element, CSS uses a set of rules to determine which style takes precedence. These rules include specificity, importance, and the order of the rules.

## CSS Example

Let's illustrate CSS with a simple example. Consider the following HTML structure:

```
<!DOCTYPE html>
<html>
<head>
  <title>My Web Page</title>
  <link rel="stylesheet" type="text/css"
href="styles.css">
</head>
<body>
  <header>
```

```
    <h1>Welcome to My Website</h1>
    <p>Where you'll find amazing content.</p>
  </header>
  <section class="main-content">
    <h2>About Us</h2>
    <p>We are a passionate team of web developers.</p>
  </section>
  <footer>
    <p>&copy; 2023 MyWebsite.com</p>
  </footer>
</body>
</html>

/* Global styles */
body {
  font-family: Arial, sans-serif;
  background-color: #f0f0f0;
}

/* Header styles */
header {
  background-color: #333;
  color: white;
  text-align: center;
  padding: 20px;
}

/* Heading styles */
h1 {
  font-size: 36px;
}

/* Main content styles */
.main-content {
  margin: 20px;
  padding: 10px;
  background-color: white;
  border: 1px solid #ddd;
  box-shadow: 0 0 5px rgba(0, 0, 0, 0.2);
}

/* Footer styles */
footer {
  text-align: center;
  padding: 10px;
  background-color: #333;
  color: white;
}
```

In this example:

We've linked an external CSS file, styles.css, to the HTML document using the <link> element.

CSS selectors like body, header, and .main-content are used to select specific elements.

Property-value pairs define styles, such as font-family, background-color, font-size, and more.

The CSS rules cascade, allowing us to define global styles and override them for specific elements.

When you load this HTML document in a web browser, you'll see that CSS has styled the elements to create an attractive and consistent design.

CSS is a vast language with numerous properties and capabilities, allowing you to customize web pages to your heart's content. As you delve deeper into web development, you'll discover CSS's full potential and become adept at creating stunning and responsive web designs.

## Exercise 1: Changing Text Color

Description: Create a CSS rule that changes the text color of all <p> elements to red.

**Steps:**

1. Create an HTML file with multiple <p> elements.
2. Create a CSS file (e.g., styles.css).
3. In the CSS file, select <p> elements using a selector.
4. Set the color property to "red" in the declaration block.

Solution:

**HTML (index.html):**

```
<!DOCTYPE html>
<html>
<head>
  <title>Text Color Exercise</title>
  <link rel="stylesheet" type="text/css"
href="styles.css">
</head>
<body>
  <p>This is a red paragraph.</p>
  <p>This is another red paragraph.</p>
</body>
</html>
```

**CSS (styles.css):**

```
/* Select all <p> elements and change text color to red
*/
p {
  color: red;
}
```

## Exercise 2: Setting Background Color

Description: Create a CSS rule that sets the background color of the <body> element to light blue.

**Steps:**

1. Modify the HTML file from Exercise 1.
2. In the CSS file, select the <body> element.
3. Set the background-color property to "lightblue" in the declaration block.
4. Solution:

**HTML (index.html):**

```
<!DOCTYPE html>
<html>
<head>
  <title>Background Color Exercise</title>
  <link rel="stylesheet" type="text/css"
href="styles.css">
</head>
<body>
  <p>This is a red paragraph.</p>
  <p>This is another red paragraph.</p>
</body>
</html>
```

**CSS (styles.css):**

```
/* Set the background color of <body> to light blue */
body {
  background-color: lightblue;
}
```

## Exercise 3: Styling Links

Description: Create a CSS rule that changes the color and removes the underline of all links (<a> elements).

Steps:

1. Modify the HTML file from Exercise 2 to include some links.
2. In the CSS file, select <a> elements.
3. Set the color property to your desired color and use text-decoration to remove the underline.

Solution:

**HTML (index.html):**

```
<!DOCTYPE html>
<html>
<head>
  <title>Link Styling Exercise</title>
  <link rel="stylesheet" type="text/css"
href="styles.css">
</head>
```

```
<body>
  <p>This is a <a href="#">link</a>.</p>
  <p>This is another <a href="#">link</a>.</p>
</body>
</html>
```

**CSS (styles.css):**

```
/* Style <a> elements */
a {
  color: blue; /* Set your desired link color */
  text-decoration: none; /* Remove the underline */
}
```

## Exercise 4: Centering Elements

Description: Create a CSS rule that centers an <h1> element both horizontally and vertically on the page.

Steps:

1. Create an HTML file with an <h1> element.
2. In the CSS file, select the <h1> element.
3. Use CSS properties like text-align and line-height to center the element.

Solution:

**HTML (index.html):**

```
<!DOCTYPE html>
<html>
```

```
<head>
  <title>Centering Exercise</title>
  <link rel="stylesheet" type="text/css"
href="styles.css">
</head>
<body>
  <h1>Center Me</h1>
</body>
</html>
```

**CSS (styles.css):**

```
/* Center the <h1> element */
h1 {
  text-align: center; /* Horizontal centering */
  line-height: 100vh; /* Vertical centering based on
viewport height */
  margin: 0; /* Remove default margin to center
precisely */
}
```

## Exercise 5: Creating a Navigation Menu

Description: Create a CSS rule that styles a navigation menu. Style the links as a horizontal menu.

Steps:

1. Create an HTML file with an unordered list (<ul>) containing list items (<li>) as links (<a>).
2. In the CSS file, select the <ul> and <li> elements.
3. Apply styles to create a horizontal navigation menu.

Solution:

**HTML (index.html):**

```
<!DOCTYPE html>
<html>
<head>
  <title>Navigation Menu Exercise</title>
  <link rel="stylesheet" type="text/css"
href="styles.css">
</head>
<body>
  <ul class="nav-menu">
    <li><a href="#">Home</a></li>
    <li><a href="#">About</a></li>
    <li><a href="#">Services</a></li>
    <li><a href="#">Contact</a></li>
  </ul>
</body>
</html>
```

**CSS (styles.css):**

```
/* Style the navigation menu */
.nav-menu {
  list-style-type: none; /* Remove list bullet points
*/
  margin: 0;
  padding: 0;
}

.nav-menu li {
  display: inline; /* Display list items horizontally
*/
  margin-right: 20px; /* Add space between menu items
*/
}

.nav-menu a {
  text-decoration: none; /* Remove underlines from
links */
  color: #333; /* Set link color */
  font-weight: bold;
}
```

## Exercise 6: Creating a Navigation Bar

Description:

Create a horizontal navigation bar with three links that are evenly spaced and have different background colors when hovered over.

Steps:

1. Create an HTML structure with an unordered list <ul> and list items <li> for each navigation item.
2. Use CSS to style the navigation bar by setting the list items to display horizontally and giving them padding, margin, and background colors.
3. Add hover effects to change the background color when a navigation item is hovered.

**HTML:**

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" type="text/css"
href="styles.css">
</head>
<body>
  <ul class="navbar">
    <li><a href="#">Home</a></li>
    <li><a href="#">About</a></li>
    <li><a href="#">Contact</a></li>
  </ul>
</body>
</html>
```

**CSS (styles.css):**

```
.navbar {
  list-style-type: none;
  padding: 0;
  margin: 0;
  display: flex;
}

.navbar li {
  margin: 0 10px;
  background-color: #3498db;
  padding: 10px;
  border-radius: 5px;
}

.navbar li:hover {
  background-color: #e74c3c;
}
```

## Exercise 7: Creating a Centered Box

Description:

Create a CSS box that is centered both horizontally and vertically on the page.

Steps:

1. Create an HTML container element (e.g., <div>) that will hold the centered box.
2. Use CSS to set the container's position to relative and center the box using flexbox or absolute positioning.

**HTML:**

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" type="text/css"
href="styles.css">
</head>
<body>
  <div class="center-container">
    <div class="center-box">Centered Box</div>
  </div>
</body>
</html>
```

**CSS (styles.css):**

```
.center-container {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
}

.center-box {
  width: 200px;
  height: 100px;
  background-color: #3498db;
  text-align: center;
  padding: 20px;
  border-radius: 10px;
}
```

## Exercise 8: Creating a Responsive Grid

Description:

Create a responsive grid layout with three columns that stack vertically on smaller screens.

Steps:

1. Define an HTML structure using a parent container and child elements for the grid items.
2. Use CSS to create a grid layout with three columns.

3. Apply media queries to change the layout to a single column on smaller screens.

**HTML:**

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" type="text/css"
href="styles.css">
</head>
<body>
  <div class="grid-container">
    <div class="grid-item">Item 1</div>
    <div class="grid-item">Item 2</div>
    <div class="grid-item">Item 3</div>
  </div>
</body>
</html>
```

**CSS (styles.css):**

```
.grid-container {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-gap: 20px;
}
```

```
.grid-item {
  background-color: #3498db;
  padding: 20px;
  text-align: center;
  border-radius: 10px;
}

@media (max-width: 768px) {
  .grid-container {
    grid-template-columns: 1fr;
  }
}
```

## Exercise 9: Creating a Dropdown Menu

Description:

Create a dropdown menu that appears when a button is clicked, and disappears when clicking outside of it.

Steps:

1. Create an HTML structure with a button and a hidden dropdown menu.
2. Use CSS to hide the dropdown menu initially.

3. Use CSS and JavaScript to toggle the visibility of the dropdown menu when the button is clicked or when clicking outside of the menu.

**HTML:**

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" type="text/css"
href="styles.css">
</head>
<body>
  <button id="dropdown-button">Toggle Dropdown</button>
  <div class="dropdown-menu" id="dropdown-menu">
    <a href="#">Option 1</a>
    <a href="#">Option 2</a>
    <a href="#">Option 3</a>
  </div>
</body>
</html>
```

**CSS (styles.css):**

```
#dropdown-menu {
  display: none;
  position: absolute;
  background-color: #3498db;
  border-radius: 5px;
  padding: 10px;
}

#dropdown-menu a {
  display: block;
  color: white;
  text-decoration: none;
  padding: 5px;
}

#dropdown-button:hover + #dropdown-menu,
#dropdown-menu:hover {
  display: block;
}

body {
  text-align: center;
  padding: 100px;
}
```

**JavaScript (script.js):**

```
document.getElementById('dropdown-button').addEventList
ener('click', function () {
  var menu = document.getElementById('dropdown-menu');
  menu.style.display = menu.style.display === 'block' ?
'none' : 'block';
});

document.addEventListener('click', function (event) {
  var menu = document.getElementById('dropdown-menu');
  var button =
document.getElementById('dropdown-button');

  if (event.target !== button && event.target !== menu)
{
    menu.style.display = 'none';
  }
});
```

## Exercise 10: Creating a Card with Hover Effects

Description:

Create a card element with hover effects that change the card's appearance.

Steps:

1. Create an HTML structure for the card.

2. Use CSS to style the card with a shadow and transition properties.
3. Apply hover effects to change the card's background color and size.

**HTML:**

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" type="text/css"
href="styles.css">
</head>
<body>
  <div class="card">
    <h2>Card Title</h2>
    <p>This is a card with hover effects.</p>
  </div>
</body>
</html>
```

**CSS (styles.css):**

```
.card {
  width: 300px;
  padding: 20px;
  background-color: #3498db;
  color: white;
  text-align: center;
  border-radius: 10px;
  box-shadow: 0px 4px 6px rgba(0, 0, 0, 0.1);
  transition: background-color 0.3s, transform 0.3s;
}

.card:hover {
  background-color: #e74c3c;
  transform: scale(1.05);
}
```

These CSS exercises cover various aspects of CSS, from creating navigation bars and centered elements to responsive grids and interactive dropdown menus. Feel free to modify and extend these exercises to further develop your CSS skills.