SEC201.2 Web-Based Programming

Cascading Style Sheets (CSS)

Adding Style to your Web Pages

Outline

- What is CSS?
- CSS Syntax
- CSS How to
  - Inline Style
  - Internal Style Sheet
  - External Style Sheet
- “Cascading” Part of CSS
- CSS Simple Selectors
  - Element
  - Class
  - Id
- CSS Combinator Selectors
- CSS Colors
- CSS Backgrounds
- Style Text
  - CSS Text
  - CSS Fonts

Review: What Kinds of Files

Files Viewed through a Web Browser

Hypertext Markup Language

Cascading Style Sheets

JavaScript

Images, pictures, art & graphics

Structure **Style** **Behavior**

What is CSS?

- **CSS** stands for **C**ascading **S**tyle **S**heets
- CSS is a language that describes the style of an HTML document
- CSS describes **how HTML elements are to be displayed on screen, paper, or in other media**
- CSS **saves a lot of work**
  - It can control the layout of multiple web pages all at once
  - With an external stylesheet file (saved in external .css file), you can change the look of an entire website by changing just one file!

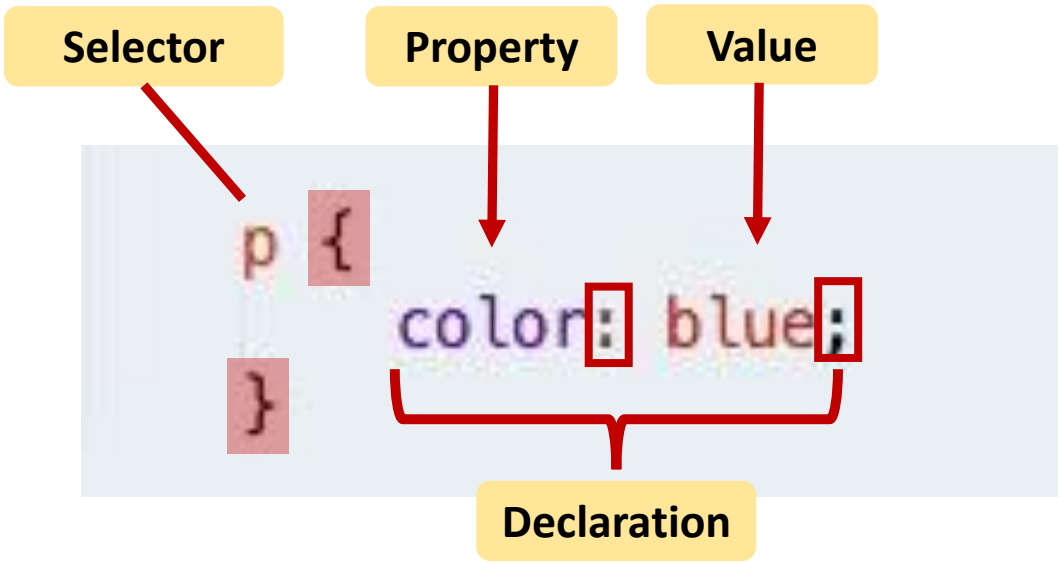
    Ex: https://www.w3schools.com/css/css_intro.asp
- Why use CSS?
  - CSS is used to define styles for your web pages, including the design, layout and variations in display for different devices and screen sizes

CSS Solved a Big Problem

- HTML was **NEVER** intended to contain tags for formatting a web page!
- HTML was created to **describe the content/structure** of a web page, like:
  - **<h1>**This is a heading**</h1>**
  - ** **This is a paragraph.** **
- When tags like **<font>**, and **color** attributes were added to the HTML 3.2 specification, it started a nightmare for web developers
  - Development of large websites, where fonts and color information were added to every single page, became a long and expensive process
- To solve this problem, the World Wide Web Consortium (W3C) created CSS
- CSS removed the style formatting from the HTML page!

CSS Syntax

- A CSS rule-set consists of a **selector** and a **declaration block**:
  - The **selector** points to the HTML element you want to style
  - The **declaration block** contains one or more declarations separated by semicolons
  - Each declaration includes a CSS property name and a value, separated by a colon
    - *Ex:* **color** is the property and **blue** is the property value
  - A CSS declaration always ends with a semicolon, and declaration blocks are surrounded by curly braces
- CSS defined generic rules that can apply to multiple elements



Multiple Properties

```
p {
    color: blue;
    font-size: 20px;
    width: : 200px;
}
```

Zero or More Declarations are allowed

Style Sheet

```
p {
    color: blue;
    font-size: 20px;
    width: : 200px;
}

h1 {
    color: green;
    font-size: 36px;
    text-align: center;
}

...
```

Stylesheet

When a browser reads a style sheet, it will format the HTML document according to the information in the style sheet

CSS How To…

<u>Don't forget</u> → ***Browsers also have default styling!!***

- The same html file may look different when viewed on different browsers
    - Some tags are supported, some aren't
    - Browsers may have different ***default styles***
- In general, default looks are plain

CSS How To…

There are three ways of inserting a style sheet

- Inline Style
- Internal Style Sheet
- External Style Sheet

Inline Styles

- To use inline styles, add the ***style attribute*** to the relevant element
- The ***style attribute*** can contain any CSS property

```
<h1 style = "color:blue">Styled Heading</h1>
```

Styled Heading

- Violated separation of content/style
- An inline style may be used to apply a unique style for a single element

Tip: ***An inline style loses many of the advantages of a style sheet (by mixing content with presentation). Use this method sparingly***

Internal Style Sheet

- Internal styles are defined within the **<style>** element, inside the **<head>** section of an HTML page
  - Styling is defined within **<head>**
  - Rules are defined within **<style>**
  - Styles are applied to all elements in that file

Tip: ***An internal style sheet may be used if one single page has a unique style***

internalStyle.html

```
<!doctype html>
<html>
<head>
        <meta charset="utf-8">
        <title>Internal Styles</title>

        <style>

            h1 {
                color:blue;
            }

            body {
                background-color: lightblue;
            }

        </style>

</head>

<body>
    <h1>Styled Heading</h1>

    <p> An internal style sheet may be used if
    one single page has a unique style.
    Internal styles are defined within the &lt;
    style&gt; element, inside the &lt;head&gt;
    section of an HTML page.</p>
</body>
</html>
```

Don't forget to close the style tag!!

week6-sampleCodes/internalStyle.html

External Style Sheet

- You can put rules in an external file (***don't use the style tag***!!)
- With an **external style sheet**, you can change the look of an entire website by changing just one file!
- Each page must include a reference to the external style sheet file inside the **<link>** element, inside the **head** section
  - The **<link>** tag defines a link between a document and an external resource
  - The **<link>** element is an empty element, it contains attributes only
    - ***rel:*** Specifies the relationship between the current document and the linked document (***required***)
    - ***type***: Specifies the media type of the linked document
    - ***href:*** Specifies the location of the linked document
  - The **<link>** element goes inside the **<head>** section:

```
<head>
    <link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
```

- Styles are applied to all elements in all files that links to the style sheet

External Style Sheet

- An external style sheet can be written in any text editor, and must be saved with a "**.css**" extension
- The external "**.css**" file should not contain any HTML tags
- Here is how the "**mystyle.css**" file looks like:

externalStyle.html

```html
<!doctype html>
<html>
    <head>
        <meta charset="utf-8">
        <title>External Styles</title>

        <link rel="stylesheet" type="text/css" href="css/mystyle.css">
    </head>


    <body>
        <h1>Styled Heading</h1>

        <p>With an external style sheet, you can change the look of an
        entire website by changing just one file! Each page must include
        a reference to the external style sheet file inside the &lt;link&
        gt; element. The &lt;link&gt; element goes inside the &lt;head&gt
        ; section. An external style sheet can be written in any text
        editor. The file should not contain any html tags. The style
        sheet file must be saved with a .css extension.</p>

    </body>
</html>
```

mystyle.css

```css
h1 {
    color: blue;
    /* margin-left: 20px; */
}

body {
    background-color: lightblue;
}
```

week6-sampleCodes/externalStyle.html
week6-sampleCodes/css/mystyle.css

The “*Cascading*” Part of CSS

What style will be used when there is more than one style specified for an HTML element?

- Generally speaking we can say that all the styles will "cascade" into a new "virtual" style sheet by the following rules (HTML is top-down)
  - Browser default
  - **External** or **Internal** style sheets (in the head section)
  - **Inline** style (inside an HTML element)
- So, an **inline** style (inside a specific HTML element) has the **highest priority**, which means that it will override a style defined inside the <head> tag, or in an external style sheet, or a browser default value

Multiple Style Sheets

- If some properties have been defined for the same selector (element) in different style sheets, the value from the last read style sheet will be used.
  - Assume that an **external style sheet** has the following style for the <h2> element:

```
h2 {
    color: green;
}
```

  - Then, assume that an **internal style sheet** also has the following style for the **<h2>** element:

```
<style>
    h2 {
        color: orange;
    }
</style>
```

Multiple Style Sheets

- If the **internal style** is defined **after** the **link to the external style** sheet, the <h2> elements will be "**orange**"

```
<head>
    <meta charset="utf-8">
    <title>Multiple Style Sheets</title>
    <link rel="stylesheet" type="text/css" href="css/multipleStyles.css">
    <style>
        h2 {
            color: orange;
        }
    </style>
</head>
```

week6-sampleCodes/external_than_internal.html

Multiple Style Sheets

- However, if the **internal style** is defined **before** the **link to the external style sheet**, the <h2> elements will be ”**green**":

```
<head>
    <meta charset="utf-8">
    <title>Multiple Style Sheets</title>
    <style>
        h2 {
            color: orange;
        }
    </style>
    <link rel="stylesheet" type="text/css" href="css/multipleStyles.css">
</head>
```

What if there is also an inline style defined for <h2>?

Rule Precedence

- What if one selector is defined in two external files?
  - The rules from the most recent file have precedence
- What if one selector has more than one rule in the same file?
  - The most recent rule has precedence

```
<style>
    h1{
        color: blue;
        font-family: Arial;
    }

    h1{
        font-family: Courier;
    }
</style>
```

week6-sampleCodes/twoExternalSheets.html
week6-sampleCodes/multiple.html

CSS Selectors

- CSS selectors are used to "find" (or select) HTML elements you want to style
- We can divide CSS selectors into 5 categories:
  1. **Simple Selectors** (select elements based on element name, id, class)
     - The **element** Selector
     - The **class** Selector
     - The **id** Selector
  2. **Combinator Selectors** (select elements based on a specific relationship between them)
  3. **Pseudo-class Selectors** (select elements based on a certain state) *(next week!!)*
  4. **Pseudo-elements Selectors** (select and style a part of an element) *(next week!!)*
  5. **Attribute Selectors** (select elements based on an attribute or attribute value) *(next week!!)*

Simple Selectors: The element Selector

- The **element** selector selects elements based on the **element name**

Element Name

```
p {
    color: blue;
}
```

- We can select all **<p>** elements on a page like above
  - In this case, all **<p>** elements will be with a blue text color

Example: The element Selector

Element Name

```
p {
    color: blue;
}
```

Blue Text

```
<body>
    . . .
    <p> . . . </p>
    . . .
    <p> . . . </p>
    . . .
</body>
```

Blue Text

Simple Selectors: The class Selector

- The **class** selector selects elements with a specific **class attribute**
- To select elements with a specific class, write a **period (.)** character, followed by the ***name of the class***

Class Name

```
.blue {
    color: blue;
}
```

The class Selector

Class Name: ***Defined with dot (.)***

```
.blue {
    color: blue;
}
```

Used without dot (.)

Blue Text

```
<body>
    . . .
    <p class="blue"> . . . </p>
    <p> . . . </p>
    <div class="blue"> . . . </div>
    . . .
</body>
```

Unaffected text

Blue Text

Example: The class Selector

Ex1: All HTML elements with class="center" will be red and center-aligned

classSelector.html

```
<!DOCTYPE html>
<html>
<head>
<style>
.center {
    text-align: center;
    color: red;
}
</style>
</head>
<body>

<h1 class="center">Red and center-aligned heading</h1>
<p class="center">Red and center-aligned paragraph.</p>
<p> This paragraph does not have any styling. </p>

</body>
</html>
```

Example: The class Selector

Ex2: We can also specify that only specific HTML elements should be affected by a class

- Below, only <p> elements with **class="center"** will be center-aligned

classSelector2.html

```
<!DOCTYPE html>
<html>
    <head>
        <style>
            p.center {
            text-align: center;
            color: red;
            }
        </style>
    </head>
    <body>

        <h1 class="center">This heading will not be affected</h1>
        <p class="center">This paragraph will be red and center-aligned.</p>

</body>
</html>
```

Example: The class Selector

Ex3: HTML elements can also refer to more than one class

- The second <p> element will be styled according to **class="center"** and to **class="large"**

classSelector3.html

```
<!DOCTYPE html>
<html>
<head>
<style>
p.center {
    text-align: center;
    color: red;
}

p.large {
    font-size: 300%;
}
</style>
</head>
<body>

<h1 class="center">This heading will not be affected</h1>
<p class="center">This paragraph will be red and center-aligned.</p>
<p class="center large">This paragraph will be red, center-aligned, and in a large font-size.</p>

</body>
</html>
```

Note: A class name cannot start with a number!

Simple Selectors: The id Selector

- The **id** selector uses the **id attribute** of an HTML element to select a specific element
- The **id** of an element should be **unique** within a page, so the **id** selector is used to select ***one unique element***!
- To select an element with a specific id, write a **hash (#)** character, followed by the ***id of the element***

id Value

```
#name {
    color: blue;
}
```

The id Selector

id Value: Defined with hash (#)

```
#name {
    color: blue;
}
```

Unaffected text

```
<body>
    . . .
    <p> . . . </p>
    <div id="name"> . . . </div>
    . . .
</body>
```

Used without hash (#)

Blue Text

Example: The id Selector

Ex: The style rule will be applied to the HTML element with **id="para1"**

idSelector.html

```html
<!DOCTYPE html>
<html>
<head>
<style>
#para1 {
    text-align: center;
    color: red;
}
</style>
</head>
<body>

<p id="para1">Hello World!</p>
<p>This paragraph is not affected by the style.</p>

</body>
</html>
```

week6-sampleCodes/idSelector.html

Grouping Selectors

- If you have elements with the same style definitions, like this

```
h1 {
    text-align: center;
    color: red;
}

h2 {
    text-align: center;
    color: red;
}

p {
    text-align: center;
    color: red;
}
```

- It will be better to **group** the selectors, to minimize the code

- To group selectors, separate each selector with a comma

```
h1, h2, p {
    text-align: center;
    color: red;
}
```

Grouping Selectors

Separate selectors with commas (,)

```
div, .blue {
    color: blue;
}
```

Unaffected text

```
<body>
    . . .
    <p class="blue"> . . . </p>
    <p> . . . </p>
    <div> . . . </div>
    . . .
</body>
```

Blue Text

Blue Text

week6-sampleCodes/simple-selectors-before.html
week6-sampleCodes/simple-selectors-after.html

Summary

CSS Simple Selectors: element, class & id

Selector | Property | Value

```
p {
    color: blue;
}
```

Declaration

selects elements based on the **element name** → **Ex:** **all <p>** elements will be with a blue text color

selector selects elements with a specific **class attribute** - write a period (.) character, followed by the name of the class → **Ex:** All HTML elements with **class="blue"** will be with a blue text color

uses the **id attribute** of an HTML element to select a ***specific element***

- **id** of an element should be **unique** within a page, so the id selector is used to select ***one unique element***!
- write a hash (#) character, followed by the id
- The style rule will be applied to the HTML element with **id="name"**

Element Selector

Class Selector

id Selector

Element Name

```
p {
    color: blue;
}
```

Class Name: ***Defined with dot (.)***

```
.blue {
    color: blue;
}
```

id Value: Defined with hash (#)

```
#name {
    color: blue;
}
```

Combining Selectors

- A CSS selector can contain more than one simple selector
- Between the simple selectors, we can include a combinator
- A **combinator** is something that explains the relationship between the selectors
- There are five different combinators in CSS:
  1. Element with class Selector → (**selector.class**)
  2. Child (direct) Selector (**>**) → (**selector > selector**)
  3. Descendant Selector (**space**) → (**selector selector**)
  4. Adjacent Sibling Selector (**+**) → (**selector + selector**)
  5. General Sibling Selector (**~**) → (**selector ~ selector**)

1. Element With Class Selector

Every **p** that has **class="big"**

```
p.big {
        font-size: 20px;
}
```

NOTE lack of space between element and class definition

Example: Element With Class Selector

```
p.big {
   font-size: 20px;
}
```

```
...
<p class="big"> ... </p>
<div class="big"> ... </div>
<p> ... </p>
...
```

Text with 20px

Unaffected text

week6-sampleCodes/advanceSelectors/elementWithClassSelector.html
week6-sampleCodes/advanceSelectors/element-with-class-before.html
week6-sampleCodes/advanceSelectors/element-with-class-after.html

2. Child (direct) Selector (>)

- The **child selector** selects all elements that are the immediate children of a specified **element**

- **Ex:** selects all **<p>** elements that are immediate children of an **<article>** element

Every **p** that is **direct** child of **article**

```
article > p {
    color: blue;
}
```

Example: Child (direct) Selector (>)

Every **p** that is **direct** child of **article**

```
article > p {
    color: blue;
}
```

```
<article> ...
    <p> ... </p>
</article>
...
<p> ... </p>
<article> ...
    <div><p> ... </p></div>
</article>
```

Blue text

Unaffected text

Unaffected text

week6-sampleCodes/advanceSelectors/childSelector.html
week6-sampleCodes/advanceSelectors/child-selector-before.html
week6-sampleCodes/advanceSelectors/child-selector-after.html

3. Descendant Selector (space)

- The **descendant selector** matches all elements that are descendants of a specified **element**

- **Ex:** selects all **<p>** elements inside **<article>** elements

Every **p** that is inside (at any level) of **article**

```
article p {
    color: blue;
}
```

Example: Descendant Selector (space)

Every **p** that is inside (at any level) of **article**

```
article  p {
    color: blue;
}
```

```
<article> ...
    <p> ... </p>
</article>
...
<p> ... </p>
<article> ...
    <div><p> ... </p></div>
</article>
```

Blue text

Unaffected text

Blue text

week6-sampleCodes/advanceSelectors/descendant-selector-before.html
week6-sampleCodes/advanceSelectors/descendant-selector-after.html

4. Adjacent Sibling Selector (+)

- The **adjacent sibling selector** selects all elements that are the adjacent siblings of a specified **element**

- **Sibling elements** must have the **same parent** element, and "**adjacent**" means "**immediately following**"

- **Ex:** selects all **<p>** elements that are placed immediately after **<article>** elements

Every **p** that is placed immediately after **article**

```
article + p {
    color: blue;
}
```

```html
<!DOCTYPE html>
<html>
<head>
<style>
article + p {
    color: blue;
}
</style>
</head>
<body>

  <article>
    <p>Paragraph 1 in the article.</p>
    <p>Paragraph 2 in the article.</p>
  </article>

  <p>Paragraph 3. Not in a article. Immediately after article...</p>
  <p>Paragraph 4. Not in a article. Immediately after p.</p>

  <h5>
      <article>
          <p>Paragraph 5. in the article.</p>
      </article>
      <p>Paragraph 6. Not in a article. Immediately after article...</p>
  </h5>

  <p>Paragraph 7. Not in a article. Immediately after h5.</p>

  <article>
      <article>
          <p>Paragraph 8. in the article.</p>
      </article>
      <p>Paragraph 9. Not in a article. Immediately after article...</p>
  </article>

  <h5>
    <p>Paragraph 10. Not in a article, but in h5.</p>
  </h5>

</body>
</html>
```

Paragraph 1 in the article.

Paragraph 2 in the article.

Paragraph 3. Not in a article. Immediately after article...

Paragraph 4. Not in a article. Immediately after p.

Paragraph 5. in the article.

Paragraph 6. Not in a article. Immediately after article...

Paragraph 7. Not in a article. Immediately after h5.

Paragraph 8. in the article.

Paragraph 9. Not in a article. Immediately after article...

Paragraph 10. Not in a article, but in h5.

Example: Adjacent Sibling Selector (+)

week6-sampleCodes/advanceSelectors/adjacentSiblingSelector.html

5. General Sibling Selector (~)

- The **general sibling selector** selects all elements that are siblings of a specified **element**

- **Sibling elements** must have the **same parent** element

- **Ex:** selects all **<p>** elements that are siblings of **<article>** elements

Every **p** that is sibling of **article**

```
article ~ p {
    color: blue;
}
```

Example: General Sibling Selector (~)

Every **p** that is sibling of **article**

```html
<!DOCTYPE html>
<html>
<head>
<style>
article ~ p {
    color: blue;
}
</style>
</head>
<body>

    <p>Paragraph 1.</p>

    <article>
      <code>Some code.</code>
      <p>Paragraph 2.</p>
    </article>

    <p>Paragraph 3.</p>
    <code>Some code.</code>
    <p>Paragraph 4.</p>

    <h3>Some header</h3>
    <p>Paragraph 5.</p>

    <h6>Smallest header
        <p>Paragraph 6.</p>
    </h6>

    <p> Paragraph 7.</p>


</body>
</html>
```

Paragraph 1.

Some code.

Paragraph 2.

Paragraph 3.

Some code.

Paragraph 4.

Some header

Paragraph 5.

Smallest header

Paragraph 6.

Paragraph 7.

week6-sampleCodes/advanceSelectors/generalSiblingSelector.html

Not Limited to Element Selectors

```
.colored p {
    color: blue;
}
```

```
p > .colored {
    color: blue;
}
```

Every **p** that is inside (at any level) an element with **class="colored"**

Every element with **class="colored"** that is **<u>direct</u>** child of **p** element

Example: Question1

Given the following HTML code:

```
<body>
  <div>
    <p>I am feeling blue</p>
  </div>
  <section>
    <p>I just want to be left alone!</p>
  </section>
</body>
```

Which of the following CSS rules turns the text of the first <p> tag blue, but **NOT** the second <p> tag?

```
<style>
  div > p {
    color: blue;
  }
</style>
```

```
<style>
  div.p {
    color: blue;
  }
</style>
```

```
<style>
  div p {
    color: blue;
  }
</style>
```

```
<style>
  p {
    color: blue;
  }
</style>
```

Example: Question2

Given the following HTML code:

```html
<body>
  <div>
    <div>
      <div class="makeMeBlue">
        <p>I am feeling blue</p>
      </div>
    </div>
    <section class="makeMeBlue">
      <p>I just want to be left alone!</p>
    </section>
  </div>
</body>
```

Which of the following CSS rules turns the text of the first <p> tag blue, but **NOT** the second <p> tag?

```style
<style>
  .makeMeBlue > p {
    color: blue;
  }
</style>
```

```style
<style>
  div.makeMeBlue p {
    color: blue;
  }
</style>
```

```style
<style>
  div > div > p {
    color: blue;
  }
</style>
```

```style
<style>
  div p {
    color: blue;
  }
</style>
```

CSS Colors

Colors in CSS are most often specified by:

1) **<u>Predefined Color Names</u>** – like ”**blue**, **red**, **yellow**, etc.”

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

<h1 style="color:Tomato;">Tomato</h1>
<h1 style="color:Orange;">Orange</h1>
<h1 style="color:DodgerBlue;">DodgerBlue</h1>
<h1 style="color:MediumSeaGreen;">MediumSeaGreen</h1>
<h1 style="color:Gray;">Gray</h1>
<h1 style="color:SlateBlue;">SlateBlue</h1>
<h1 style="color:Violet;">Violet</h1>
<h1 style="color:LightGray;">LightGray</h1>

</body>
</html>
```

Tomato

Orange

DodgerBlue

MediumSeaGreen

Gray

SlateBlue

Violet

LightGray

CSS Colors

Colors in CSS are most often specified by:

2) **<u>A RGB Value</u>** – like "***rgb(255,0,0)***" displayed as ***red***
   - A color can be specified as an RGB value, using this formula: ***rgb(red, green, blue)***
   - Each parameter (red, green, blue) defines the intensity of the color between 0 and 255
     - To display the color ***black***, all color parameters must be set to 0, like this: ***rgb(0, 0, 0)***
     - To display the color **<u>white</u>**, all color parameters must be set to 255, like this: ***rgb(255, 255, 255)***

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

<h1 style="color:rgb(255, 0, 0)">rgb(255, 0, 0)</h1>
<h1 style="color:rgb(255, 165, 0)">rgb(255, 165, 0)</h1>
<h1 style="color:rgb(0, 0, 255)">rgb(0, 0, 255)</h1>
<h1 style="color:rgb(60, 179, 113)">rgb(60, 179, 113)</h1>
<h1 style="color:rgb(106, 90, 205)">rgb(106, 90, 205)</h1>
<h1 style="color:rgb(238, 130, 238)">rgb(238, 130, 238)</h1>

<p>In HTML, you can specify colors using RGB values.</p>

</body>
</html>
```

rgb(255, 0, 0)

rgb(255, 165, 0)

rgb(0, 0, 255)

rgb(60, 179, 113)

rgb(106, 90, 205)

rgb(238, 130, 238)

In HTML, you can specify colors using RGB values.

CSS Colors

#0000FF #FF0000 #FFFF00

Colors in CSS are most often specified by:

3) **<u>A Hexadecimal Value</u>** – like "**#000FF**, **#FF0000**, **#FFFF00**"

- A color can be specified using a **hexadecimal** value
- **#RRGGBB**, where **RR** (red), **GG** (green), **BB** (blue) are hexadecimal values between 00 and FF (same as decimal 0-255)

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

<h1 style="color:#ff0000;">#ff0000</h1>
<h1 style="color:#0000ff;">#0000ff</h1>
<h1 style="color:#3cb371;">#3cb371</h1>
<h1 style="color:#ee82ee;">#ee82ee</h1>
<h1 style="color:#ffa500;">#ffa500</h1>
<h1 style="color:#6a5acd;">#6a5acd</h1>

<p>In HTML, you can specify colors using Hex values.
</p>

</body>
</html>
```

#ff0000

#0000ff

#3cb371

#ee82ee

#ffa500

#6a5acd

In HTML, you can specify colors using Hex values.

CSS Colors

hsl(240, 100%, 50%)

Colors in CSS are most often specified by:

- In addition, CSS3 also introduces:

4) **A RGBA Value**
   - RGBA color values are an extension of RGB color values with an **alpha channel** - which specifies the **opacity**
   - An RGBA color value is specified with: ***rgba(red, green, blue, alpha)***
   - The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (not transparent at all)

5) **A HSL Value**
   - A color can be specified using **hue**, **saturation** and **lightness** (HSL) in the form:
     hsl(hue, saturation, lightness)
     - Hue is a degree on the color wheel from 0 to 360, where 0 is red, 120 is green, and 240 is blue
     - Saturation is a percentage value, 0% means a shade of gray, and 100% is the full color
     - Lightness is also a percentage, 0% is black, 50% is neither light or dark, 100% is white

6) **A HSLA Value**
   - HSLA color values are an extension of HSL color values with an **alpha channel** - which specifies the **opacity**
   - An HSLA color value is specified with: ***hsla(hue, saturation, lightness, alpha)***
   - The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (not transparent at all)

CSS Colors

We can set the color of, what?

Hello World

Hello World

Hello World

- **Text Color**
  - We can set the color of text

```
<h1 style="color:Tomato;">Hello World</h1>
```

- **Background Color**
  - We can set the background color for HTML elements

```
<h1 style="background-color:DodgerBlue;">Hello World</h1>
```

- **Border Color**
  - We can set the color of borders

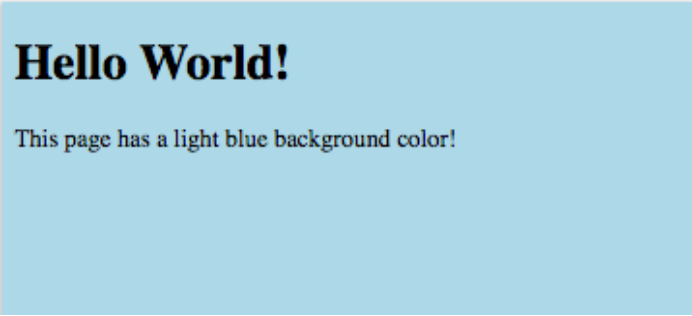
```
<h1 style="border:2px solid Tomato;">Hello World</h1>
```

https://www.w3schools.com/css/css_colors.asp

CSS Backgrounds

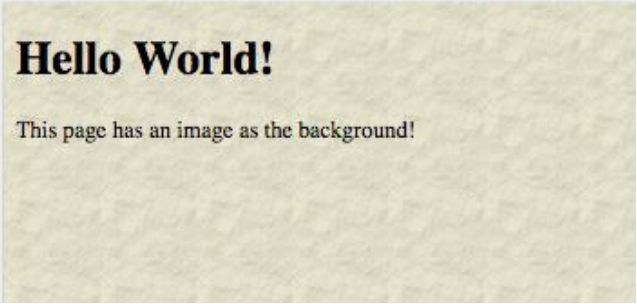
- CSS **background** properties are used to define the background effects for elements
- CSS background properties are
  - **background-color** → specifies the background color of an element
    - The background color of a page is set like this:

```
<style>
    body {
        background-color: lightblue;
    }
</style>
```



  - **background-image** → specifies an image to use as the background of an element
    - By default, the image is repeated so it covers the entire element

```
<style>
    body {
        background-image: url("paper.gif");
    }
</style>
```



CSS Backgrounds

```
body {
    background-image: url("img_tree.png");
    background-repeat: no-repeat;
    background-position: right top;
    background-attachment: fixed;
}
```

- CSS background properties are
  - **background-repeat** → Some images should be repeated only horizontally or vertically
    - ***no-repeat***, ***repeat-x*** (to repeat an image horizontally), ***repeat-y*** (to repeat an image vertically)
  - **background-attachment** → To specify that the background image should scroll or be **fixed**
    - ***fixed, scroll*** (will not scroll with the rest of the page)

    Ex: *https://www.w3schools.com/css/tryit.asp?filename=trycss_background-image_attachment*
  - **background-position** → sets the starting position of a background image
    - By default a background-image is placed at the top-left corner of an element, and repeat both vertically and horizontally
    - left top, left center, left bottom, right top, right center, right bottom, center top, center center, center bottom

week6-sampleCodes/background/background-before.html
week6-sampleCodes/background/background-after.html

- Background – Shorthand property
  - Specify all the background properties in one single property, namely **background** property

```
body {
    background: #ffffff url("img_tree.png") no-repeat right top;
}
```

Styling Text: CSS Text

- **Text Color**
  - The **color** property is used to set the color of the text
  - h1 { **color**: **blue**; }
  - The default text color for a page is defined in the body selector
- **Text Alignment**
  - The **text-align** property is used to set the horizontal alignment of a text
  - Values for text-align property: **left**, **right**, **center**, or **justify**
  - h1 { **text-align**: **left**; }
- **Text Decoration**
  - The **text-decoration** property is used to set or remove decorations from text
  - The property-value pair **text-decoration: none;** is often used to remove underlines from links
    - a { **text-decoration**: **none**; }
  - Other values for text-decoration property used to decorate text: **overline**, **underline**, **line-through**

Styling Text: CSS Text

- **Text Transformation**
  - The **text-transform** property is used to specify uppercase and lowercase letters in a text
  - Values for text-transformation property: turn everything into **uppercase** or **lowercase** letters, or **capitalize** the first letter of each word
  - p { **text-transform**: **capitalize**}
- **Text Indentation**
  - The **text-indent** property is used to specify the indentation of the first line of a text
  - p { **text-indent**: **50px**; }
- **Letter Spacing**
  - The **letter-spacing** property is used to specify the space between the characters in a text
  - h1 { **letter-spacing**: **3px**; }

Styling Text: CSS Text

- **Line Height**
  - The **line-height** property is used to specify the space between lines, hence not affects font
  - p.small { **line-height**: **0.8**; } or p.big { **line-height**: **1.8**; }
- **Word Spacing**
  - The **word-spacing** property is used to specify the space between the words in a text
  - h1 { **word-spacing**: **10px**; } or h2 { **word-spacing**: **-5px**; }
- **Text Shadow**
  - The **text-shadow** property adds shadow to text
  - h1 { **text-shadow**: **3px 2px red**; }

  → the position of the horizontal shadow (3px),
  the position of the vertical shadow (2px)
  and the color of the shadow (red)

Text-shadow effect

week6-sampleCodes/stylingText/styling-text-before.html
week6-sampleCodes/stylingText/styling-text-after.html

Styling Text: CSS Fonts

CSS font property define the font family, boldness, size, and style of a text

- **Font Family**
  - The font family of a text is set with the **font-family** property
  - The **font-family** property should hold several font names as a "fallback" system
    - If the browser does not support the first font, it tries the next font, and so on
  - p { **font-family**: **"Times New Roman", Times, serif**; }
- **Font Style**
  - The **font-style** property is mostly used to specify italic text. It has three values
    - **normal** - The text is shown normally
    - **italic** - The text is shown in italics
    - **oblique** - The text is "leaning" (oblique is very similar to italic, but less supported)
  - p.italic { **font-style**: **italic**; }

Styling Text: CSS Fonts

```
p.normal {
    font-weight: normal;
}
p.thick {
    font-weight: bold;
}
```

This is a paragraph.

This is a paragraph.

- **Font Weight**
  - The **font-weight** property specifies the weight of a font (***normal***, ***bold***)
- **Font Variant**
  - The **font-variant** property specifies whether or not a text should be displayed in a small-caps font
  - In a **small-caps font**, ***all lowercase letters are converted to uppercase letters*** However, the converted uppercase letters appears in a smaller font size than the original uppercase letters in the text

```
p.normal {
    font-variant: normal;
}

p.small {
    font-variant: small-caps;
}
```

My name is Hege Refsnes.

MY NAME IS HEGE REFSNES.

Styling Text: CSS Fonts

- **Font Size**
  - The **font-size** property sets the size of the text
  - Managing the text size is important in web design
    - However, you should not use font size adjustments to make paragraphs look like headings, or headings look like paragraphs
    - Always use the proper HTML tags, like <h1> - <h6> for headings and <p> for paragraphs
  - The font-size value can be an **absolute**, or **relative** size
  - **Absolute Size (px)**
    - Sets the text to a specified size
    - Does not allow a user to change the text size in all browsers (bad for accessibility reasons)
    - Absolute size is useful when the physical size of the output is known
  - **Relative Size (em)**
    - Sets the size relative to surrounding elements
    - Allows a user to change the text size in browsers

Note: If you do not specify a font size, the default size for normal text, like paragraphs, is 16px (16px=1em)

Styling Text: CSS Fonts

Absolute Size

- Set Font Size with **Pixels**
  - Setting the text size with pixels gives you full control over the text size

Relative Size

- Set Font Size with **Em**
  - To allow users to resize the text (in the browser menu), many developers use **em** instead of pixels
  - Recommended by the W3C
  - 1em = 16px
  - The size can be calculated from pixels to em using this formula: ***pixels*/16=*em***

Quote from "The Principles of Beautiful Web Design"

"An **em** is a **CSS** unit that measures the size of a font, from the top of a font's cap height to the bottom of its lowest descender. Originally, the **em** was equal to the width of the capital letter M, which is where its name originated."

Styling Text: CSS Fonts

In this example the text size is ***same*** in both. However, with the ***em*** size, it is possible to adjust the text size in all browsers

week6-sampleCodes/font-size/absoluteSize.html
week6-sampleCodes/font-size/relativeSize.html

Absolute Size

```
<style>
  h1 {
      font-size: 40px;
  }

  h2 {
      font-size: 30px;
  }

  p {
      font-size: 14px;
  }
</style>
```

Relative Size

```
<style>
  h1 {
      font-size: 2.5em; /* 40px/16=2.5em */
  }

  h2 {
      font-size: 1.875em; /* 30px/16=1.875em */
   }

  p {
      font-size: 0.875em; /* 14px/16=0.875em */
  }
</style>
```

Unfortunately, there is still a problem with older versions of IE with the em size → The text becomes larger than it should when made larger, and smaller than it should when made smaller!

Solution: Use a Combination of Percent and Em

The solution that works in all browsers, is to set a default **font-size** in **percent** for the **<body>** element

```
body {
    font-size: 100%;
}

h1 {
    font-size: 2.5em;
}

h2 {
    font-size: 1.875em;
}

p {
    font-size: 0.875em;
}
```

Our code now works great! It shows the same text size in all browsers and allows all browsers to zoom or resize the text!

week6-sampleCodes/font-size/combination.html
week6-sampleCodes/font-size/font-size-before.html
week6-sampleCodes/font-size/font-size-after.html