SEC201.2 Web-Based Programming

Cascading Style Sheets (CSS)

Adding Style to your Pages – Part II

Review: What is CSS?

- **CSS** stands for **C**ascading **S**tyle **S**heets
- 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!

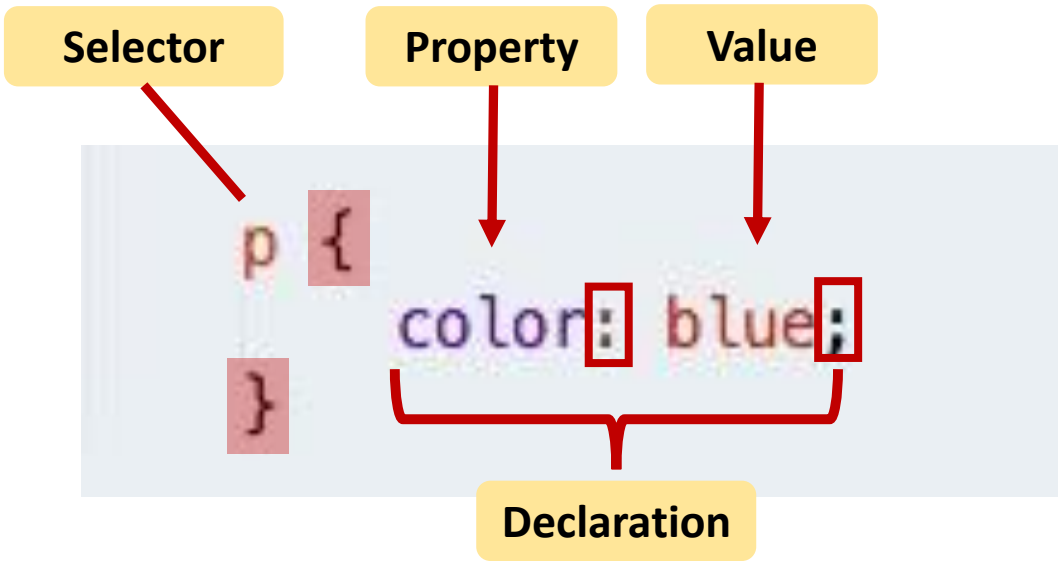
    Ex: https://www.w3schools.com/css/css_intro.asp
- Why use CSS?
  - CSS is used to define styles for your web pages, including the design, layout and variations in display for different devices and screen sizes

Review: 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!

Review: 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
  - 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



Review: CSS How To…

Don't forget → ***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

Review: CSS How To…

- There are three ways of inserting a style sheet
  - Inline Style
  - Internal Style Sheet
  - External Style Sheet

Review: 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*

Review: 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!!

Review: 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
  - 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

Review: External Style Sheet

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;
}
```

Review: 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

Review: 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>
```

Review: 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>
```

Review: 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>?

Review: 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>
```

Review: 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)
  4. **Pseudo-elements Selectors** (select and style a part of an element)
  5. **Attribute Selectors** (select elements based on an attribute or attribute value)

Review: CSS Selectors
(1) Simple Selectors: element, class & id

CSS selectors are used to "find" (or select) HTML elements based on their element name, id, class attribute, and more (***this week!!)***

- The **element** Selector
- The **class** Selector
- The **id** Selector

Summary: CSS Selectors
(1) 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

Element Selector

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

Class Selector

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"**

id Selector

Element Name

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

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

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

id Value: Defined with hash (#)

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

Review: CSS Selectors
(2) 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/CSS3:
  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**)

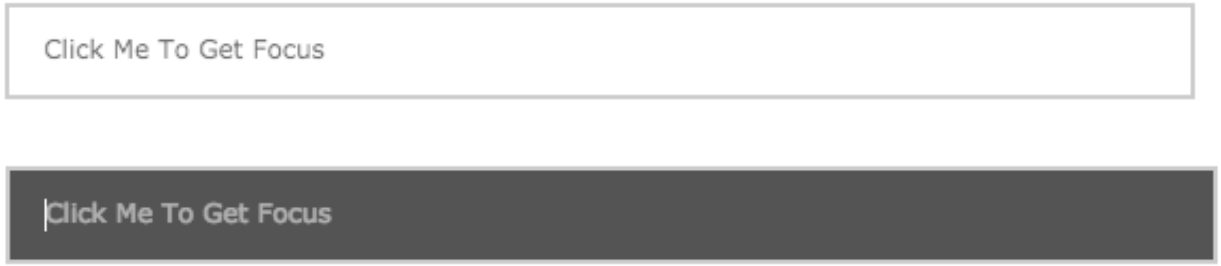
Outline

- **CSS Selectors (cont.)**
  - **Pseudo-Class Selectors**
- CSS Links
- CSS Selectors (cont.)
  - Pseudo-elements Selectors
  - Attribute Selectors
- CSS Lists
- The Box Model
  - CSS Borders
  - CSS Margins
  - CSS Padding
  - CSS Height/Width
  - CSS Box Model
- CSS Overflow

CSS Selectors:
(3) Pseudo-Class Selectors

What are Pseudo-classes?

- A pseudo-class is used to define a special **state** of an element
- For example, it can be used to:
  - Style an element when a user mouse over it
  - Style visited and unvisited links differently
  - Style an element when it gets focus



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

Pseudo-Class Selector - Syntax

```
selector:pseudo-class {
    property:value;
}
```

- Many pseudo-class selectors exist
  → **:link**, **:visited**, **:hover**, **:active**, **:nth-child(…)**
- Let's look at pseudo-class selectors in ***CSS Links***

Outline

- CSS Selectors (cont.)
  - Pseudo-Class Selectors
- **CSS Links**
- CSS Selectors (cont.)
  - Pseudo-elements Selectors
  - Attribute Selectors
- CSS Lists
- The Box Model
  - CSS Borders
  - CSS Margins
  - CSS Padding
  - CSS Height/Width
  - CSS Box Model
- CSS Overflow

Styling Links: CSS Links

- Links can be styled with any CSS property
  - E.g. **color**, **font-family**, **background**, etc.

```
<a href="https://www.w3schools.com/html/">Visit our HTML tutorial</a>
```

- Some links are blue, some are purple, etc.
  - Why??? → ***Pseudo-classes***
  - Links can also be styled differently depending on what **state** they are in

Styling Links: Anchor Pseudo-classes

The four link states are:

- **a:link** – selects all unvisited links
- **a:visited** – selects all visited links
- **a:hover** – selects links on mouse over
- **a:active** – selects the active link (the moment it is clicked)

CSS Links: Precedence of Rules

- When setting the style for several link states, there are some order rules:
  - **a:hover** MUST come after **a:link** and **a:visited** in the CSS definition in order to be effective!
  - **a:active** MUST come after **a:hover** in the CSS definition in order to be effective!

```html
<!DOCTYPE html>
<html>
<head>
<style>
/* unvisited link */
a:link {
    color: red;
}

/* visited link */
a:visited {
    color: green;
}

/* mouse over link */
a:hover {
    color: hotpink;
}

/* selected link */
a:active {
    color: blue;
}
</style>
</head>
<body>

<p><b><a href="http://google.com" target="_blank">
This is a link to Google</a></b></p>
<p><b>Note:</b> a:hover MUST come after a:link and
a:visited in the CSS definition in order to be
effective.</p>
<p><b>Note:</b> a:active MUST come after a:hover in
the CSS definition in order to be effective.</p>

</body>
</html>
```

week7-sampleCodes/stylingLinks/stylingLinks.html

Example: CSS Links

```html
<!DOCTYPE html>
<html>
<head>
<style>
a:link, a:visited {
    background-color: #f44336;
    color: white;
    padding: 14px 25px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
}


a:hover, a:active {
    background-color: blue;
}
</style>
</head>
<body>

<a href="http://google.com" target="_blank">This is a link to Google</a>

</body>
</html>
```

This is a link to Google

week7-sampleCodes/stylingLinks/stylingLinks2.html

Pseudo-classes and CSS Classes

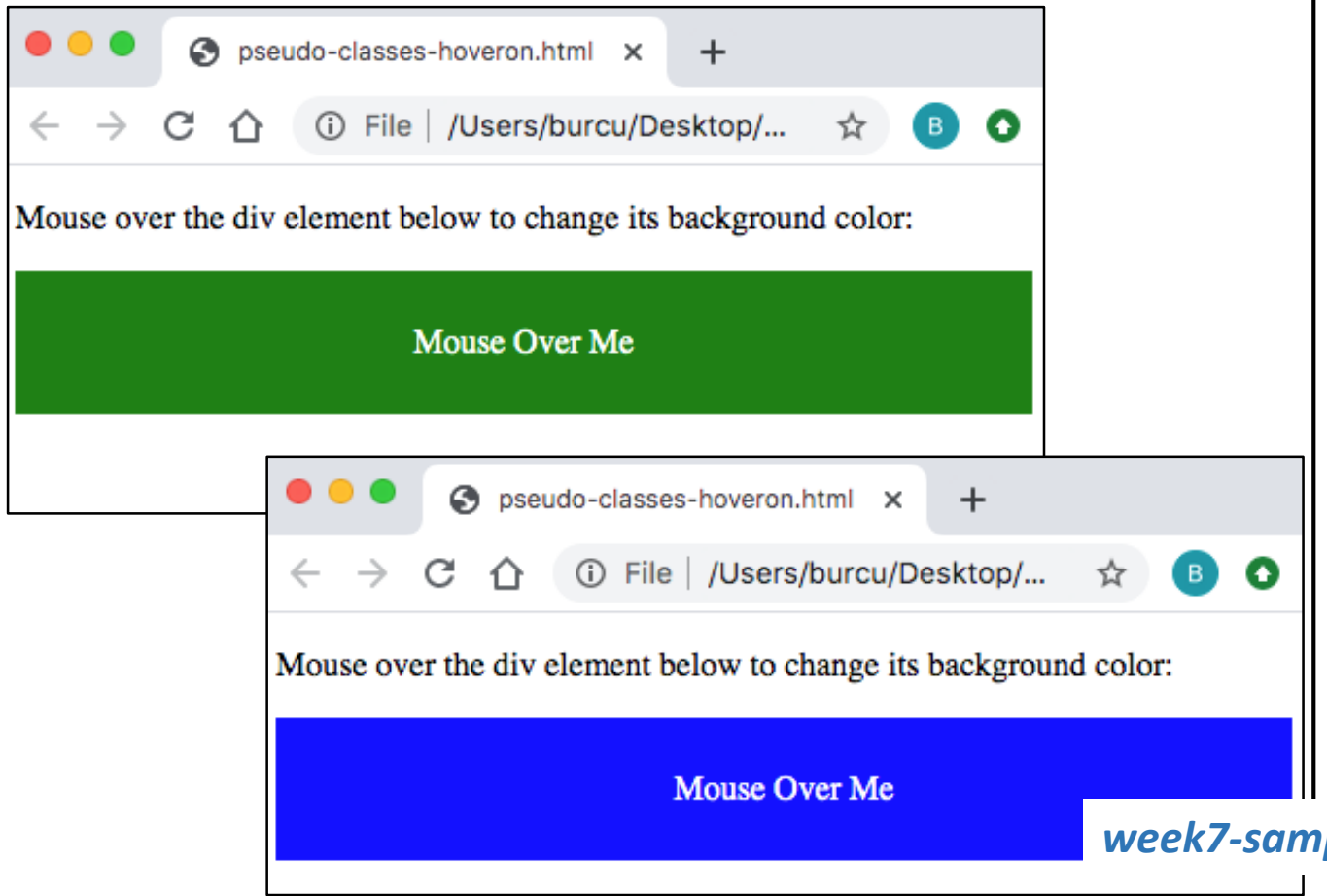
- Pseudo-classes can be combined with CSS classes
- ***Ex***: When you hover over the link in the below example, it will change color

```
<!DOCTYPE html>
<html>
<head>
        <style> a.highlight:hover { color: #ff0000; } </style>
</head>
<body>
<p><a class="highlight"
href="https://www.w3schools.com/css/css_pseudo_classes.asp">CSS Pseudo-
classes</a></p>
<p><a href="https://www.w3schools.com/css/default.asp">CSS Tutorial</a></p>

</body>
</html>
```

week7-sampleCodes/pseudo-selectors/pseudo-classes_CSSclasses.html

Example: Hover on

An example of using the **:hover** pseudo-class on a ** ** element

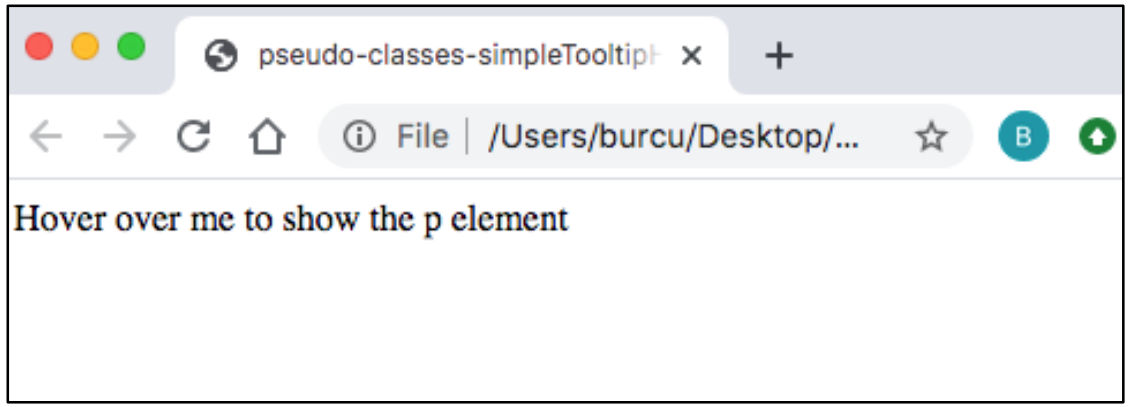


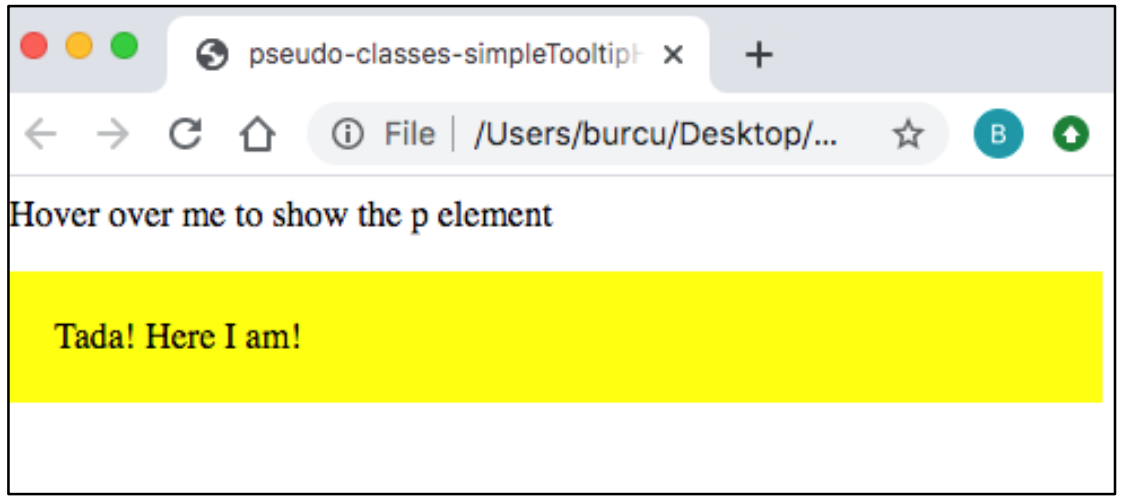

```html
<!DOCTYPE html>
<html>
<head>
  <style>
    div {
      background-color: green;
      color: white;
      padding: 25px;
      text-align: center;
    }

    div:hover {
        background-color: blue;
    }
  </style>
</head>

<body>
<p>Mouse over the div element below to change its
background color:</p>
<div>Mouse Over Me</div>
</body>
</html>
```

week7-sampleCodes/pseudo-selectors/pseudo-classes-hoveron.html

Example: Simple Tooltip Hover

```html
<!DOCTYPE html>
<html>
<head>
    <style>
        p {
          display: none;
          background-color: yellow;
          padding: 20px;
        }

        div:hover p {
            display: block;
        }
    </style>
</head>
<body>
<div>Hover over me to show the p element
  <p>Tada! Here I am!</p>
</div>
</body>
</html>
```





week7-sampleCodes/pseudo-selectors/pseudo-classes-simpleTooltipHover.html

The *:nth-child(…)* Pseudo-class

- The **:nth-child(*n*)** selector matches every element that is the *n*th child, regardless of type, of its parent

- ***n*** can be a number, a keyword, or a formula

```
<style>
    /* Styles go here. */
    header li {
      list-style: none;
    }
    a:link, a:visited {
      text-decoration: none;
      background-color: green;
      border: 1px solid blue;
      color: white;
      display: block;
      width: 200px;
      text-align: center;
      margin-bottom: 1px;
    }
    a:hover, a:active {
      background-color: red;
      color: purple;
    }
    header li:nth-child(3) {
      font-size: 36px;
    }
    section div:nth-child(odd) {
      background-color: lightblue;
    }
    section div:nth-child(4):hover {
      background-color: pink;
      cursor: pointer;
}
</style>
```

week7-sampleCodes/pseudo-selectors/pseudo-selectors-before.html
week7-sampleCodes/pseudo-selectors/pseudo-selectors-after.html

The *:first-child(…)* Pseudo-class

- The **:first-child** pseudo-class matches a specified element that is the first child of another element
- *Ex1*: Match the first <p> element
  - The selector matches any <p> element that is the first child of any element

```
p:first-child {
  color: blue;
}
```

week7-sampleCodes/pseudo-selectors/pseudo-classes-first-child-example1.html

- *Ex2*: Match the first <i> element in all <p> elements
  - The selector matches the first <i> element in all <p> elements

```
p i:first-child {
  color: blue;
}
```

week7-sampleCodes/pseudo-selectors/pseudo-classes-first-child-example2.html

- *Ex3*: Match all <i> elements in all first child <p> elements
  - The selector matches all <i> elements in <p> elements that are the first child of another element

```
p:first-child i {
  color: blue;
}
```

week7-sampleCodes/pseudo-selectors/pseudo-classes-first-child-example3.html

The *:lang* Pseudo-class

- The **:lang** pseudo-class allows you to define special rules for different languages

- In this example, **:lang** defines the quotation marks for <q> elements with lang="no"

```
<!DOCTYPE html>
<html>
<head>
    <style>
        q:lang(no) {
            quotes: "~" "~";
        }
    </style>
</head>

<body>

<p>Some text <q lang="no">A quote in a
paragraph</q> Some text.</p>
<p>In this example, :lang defines the
quotation marks for q elements with
lang="no":</p>
<p><b>Note:</b> IE8 supports the :lang
pseudo class only if a !DOCTYPE is
specified.</p>

</body>
</html>
```

All CSS Pseudo Classes
https://www.w3schools.com/css/css_pseudo_classes.asp

Outline

- CSS Selectors (cont.)
  - Pseudo-Class Selectors
- CSS Links
- **CSS Selectors (cont.)**
  - **Pseudo-elements Selectors**
  - Attribute Selectors
- CSS Lists
- The Box Model
  - CSS Borders
  - CSS Margins
  - CSS Padding
  - CSS Height/Width
  - CSS Box Model
- CSS Overflow

CSS Selectors:
(4) Pseudo-elements Selectors

What are Pseudo-Elements?

- A CSS pseudo-element is used to style specified parts of an element
- For example, it can be used to:
  - Style the first letter, or line, of an element
  - Insert content before, or after, the content of an element

Pseudo-element Selector - Syntax

```
selector::pseudo-element {
    property:value;
}
```

- Many pseudo-element selectors exist

  → **::first-line, ::first-letter, ::before, ::after, ::selection**

The *::first-line* Pseudo-element

- The **::first-line** pseudo-element is used to add a special style to the first line of a text
- *Ex*: Format the first line of the text in all <p> elements

```
p::first-line {
  color: #ff0000;
  font-variant: small-caps;
}
```

week7-sampleCodes/pseudo-selectors/pseudo-elements-first-line.html

- ***Note***: The **::first-line** pseudo-element can only be applied to **block-level elements**
- The following properties apply to the **::first-line** pseudo-element: *font properties, color properties, background properties, word-spacing, letter-spacing, text-decoration, vertical-align, text-transform, line-height, clear*

The *::first-letter* Pseudo-element

- The **::first-letter** pseudo-element is used to add a special style to the first letter of a text.
- *Ex*: Format the first letter of the text in all <p> elements

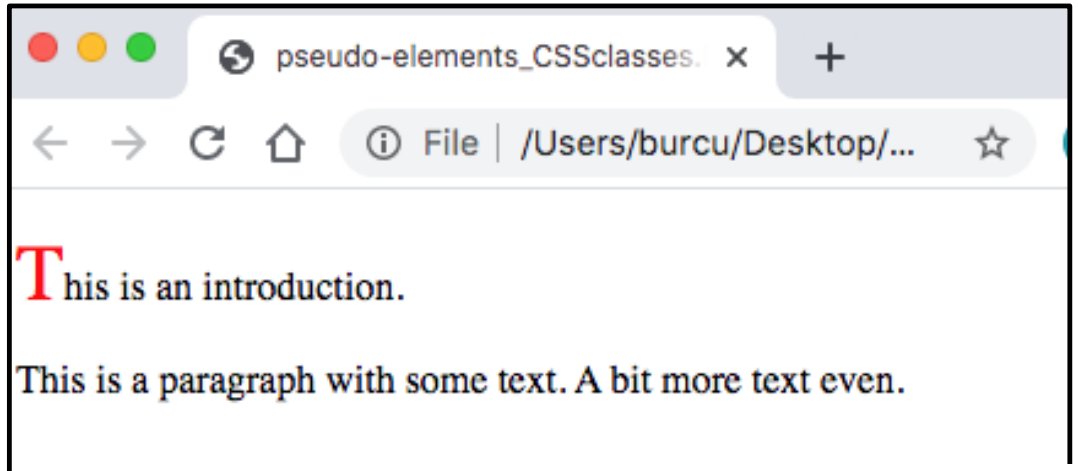
```
p::first-letter {
  color: #ff0000;
  font-size: xx-large;
}
```

week7-sampleCodes/pseudo-selectors/pseudo-elements-first-letter.html

- ***Note***: The **::first-letter** pseudo-element can only be applied to **block-level elements**
- The following properties apply to the **::first-letter** pseudo-element: *font properties, color properties, background properties, margin properties, padding properties, border properties, text-decoration, vertical-align (only if "float" is "none"), text-transform, line-height, float, clear*

Pseudo-elements and CSS Classes

- Pseudo-elements can be combined with CSS classes!!
- This example will display the first letter of paragraphs with class="intro", in red and in a larger size



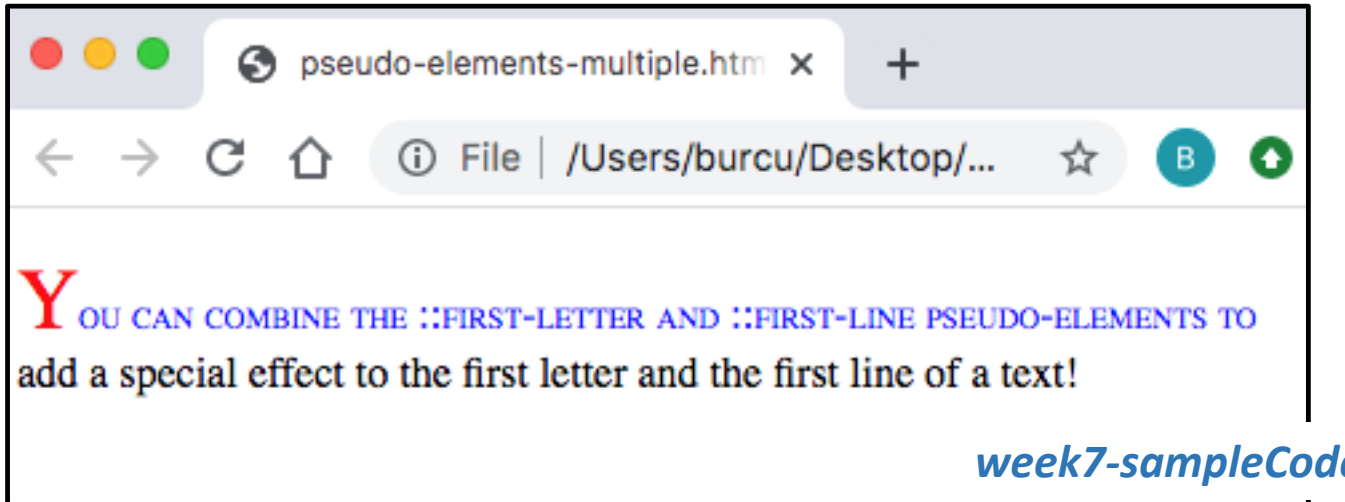

```html
<!DOCTYPE html>
<html>
<head>
    <style>
        p.intro::first-letter {
          color: #ff0000;
          font-size:200%;
        }
    </style>
</head>
<body>

<p class="intro">This is an introduction.</p>
<p>This is a paragraph with some text. A bit
more text even.</p>

</body>
</html>
```

week7-sampleCodes/pseudo-selectors/pseudo-elements_CSSclasses.html

Multiple Pseudo-elements

- Several pseudo-elements can be combined
- *Ex*: The first letter of a paragraph will be red, in an xx-large font size. The rest of the first line will be blue, and in small-caps. The rest of the paragraph will be the default font size and color

```html
<!DOCTYPE html>
<html>
<head>
    <style>
        p::first-letter {
          color: #ff0000;
          font-size: xx-large;
        }

        p::first-line {
          color: #0000ff;
          font-variant: small-caps;
        }
    </style>
</head>
<body>

<p>You can combine the ::first-letter and
::first-line pseudo-elements to add a
special effect to the first letter and the
first line of a text!</p>

</body>
</html>
```



week7-sampleCodes/pseudo-selectors/pseudo-elements-multiple.html

The *::before* & The *::after* Pseudo-element

- The **::before** pseudo-element can be used to insert some content before the content of an element.
- *Ex*: Inserts an image before the content of each <h1> element

```
h1::before {
  content: url(smiley.gif);
}
```

This is a heading

week7-sampleCodes/pseudo-selectors/pseudo-elements-before.html

- The **::after** pseudo-element can be used to insert some content after the content of an element.
- *Ex*: Inserts an image after the content of each <h1> element

```
h1::after {
  content: url(smiley.gif);
}
```

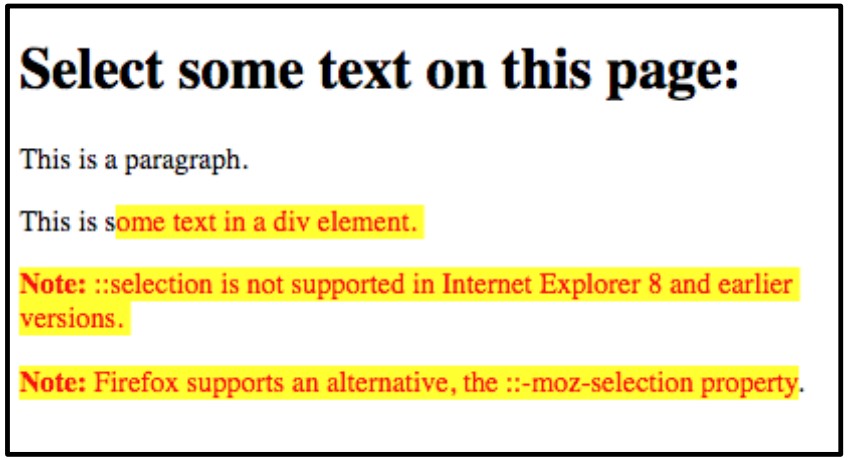
This is a heading

week7-sampleCodes/pseudo-selectors/pseudo-elements-after.html

The ::selection Pseudo-element

- The **::selection** pseudo-element matches the portion of an element that is selected by a user
- The following CSS properties can be applied to **::selection** → **color**, **background**, **cursor**, and **outline**
- *Ex*: Make the selected text red on a yellow background

```
::selection {
  color: red;
  background: yellow;
}
```



week7-sampleCodes/pseudo-selectors/pseudo-elements-selection.html

Outline

- CSS Selectors (cont.)
  - Pseudo-Class Selectors
- CSS Links
- **CSS Selectors (cont.)**
  - Pseudo-elements Selectors
  - **Attribute Selectors**
- CSS Lists
- The Box Model
  - CSS Borders
  - CSS Margins
  - CSS Padding
  - CSS Height/Width
  - CSS Box Model
- CSS Overflow

CSS Selectors: (5) Attribute Selectors

- It is possible to style HTML elements that have specific attributes or attribute values with the attribute selectors
- They can be useful especially for styling forms without class or ID
- CSS **[attribute]** Selector
  - The **[attribute]** selector is used to select elements with a specified attribute
  - *Ex*: Select all **<a>** elements with a **target** attribute

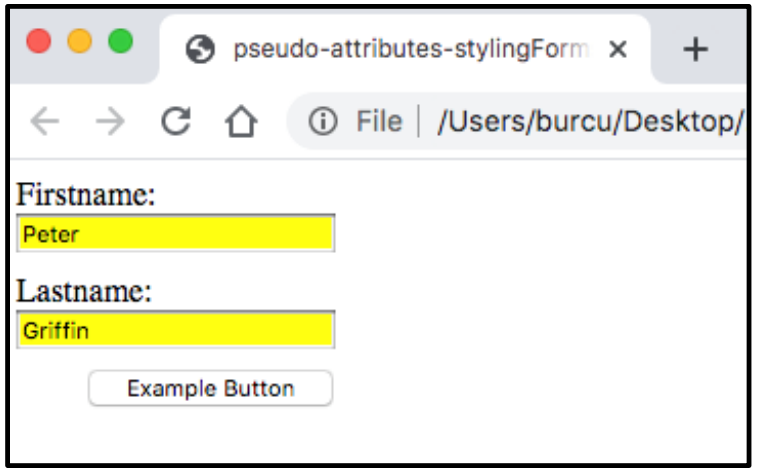
```
<!DOCTYPE html>
<html>
<head>
    <style>
        a[target] {
          background-color: yellow;
        }
    </style>
</head>
<body>

<p>The links with a target attribute gets a yellow background:</p>

<a href="https://www.w3schools.com">w3schools.com</a>
<a href="http://www.disney.com" target="_blank">disney.com</a>
<a href="http://www.wikipedia.org" target="_top">wikipedia.org</a>

<p><b>Note:</b> For [<i>attribute</i>] to work in IE8 and
earlier, a DOCTYPE must be declared.</p>

</body>
</html>
```

week7-sampleCodes/pseudo-selectors/pseudo-attributes.html

All CSS Attribute Selectors

Selector	Example	Example Description
[attribute]	[target]	Selects all elements with a target attribute
[attribute=value]	[target=_blank]	Selects all elements with target="_blank"
[attribute~=value]	[title~=flower]	Selects all elements with a title attribute containing the word "flower"
[attribute|=value]	[lang|=en]	Selects all elements with a lang attribute value starting with "en"
[attribute^=value]	a[href^="https"]	Selects every <a> element whose href attribute value begins with "https"
[attribute$=value]	a[href$=".pdf"]	Selects every <a> element whose href attribute value ends with ".pdf"
[attribute*=value]	a[href*=”ceng"]	Selects every <a> element whose href attribute value contains the substring ”ceng"

Example: Styling Forms




```html
<!DOCTYPE html>
<html>
<head>
  <style>
    input[type="text"] {
      width: 150px;
      display: block;
      margin-bottom: 10px;
      background-color: yellow;
    }

    input[type="button"] {
      width: 120px;
      margin-left: 35px;
      display: block;
    }
  </style>
</head>

<body>
  <form name="input" action="" method="get">
    Firstname:<input type="text" name="Name" value="Peter" size="20">
    Lastname:<input type="text" name="Name" value="Griffin" size="20">
    <input type="button" value="Example Button">
  </form>
</body>
</html
```

week7-sampleCodes/pseudo-selectors/pseudo-attributes-stylingForms.html

Outline

- CSS Selectors (cont.)
  - Pseudo-Class Selectors
- CSS Links
- CSS Selectors (cont.)
  - Pseudo-elements Selectors
  - Attribute Selectors
- **CSS Lists**
- The Box Model
  - CSS Borders
  - CSS Margins
  - CSS Padding
  - CSS Height/Width
  - CSS Box Model
- CSS Overflow

Styling Lists: CSS Lists

- In HTML, there are two main types of lists
  - unordered lists (**<ul>**) - the list items are marked with bullets
  - ordered lists (**<ol>**) - the list items are marked with numbers or letters

- The CSS list properties allow us to
  - Set different **list item markers** for **ordered lists**
  - Set different **list item markers** for **unordered lists**
  - Set an **image** as the **list item marker**
  - Add **background colors** to **lists and list items**

Different List Item Markers

```
ol {
    list-style-type: upper-alpha;
}
```

- The **list-style-type** property specifies the type of list item marker that is usually positioned to the left of any list item

A. Knight Rider

B. A-Team

- **<u>For ordered lists</u>**
  - You can choose different ways of having the numbers shown: decimal, decimal-leading-zero, lower-roman, upper-roman, lower-alpha, upper-alpha, as well as several of the worlds languages: armenian, georgian, hebrew, simp-chinese-formal, and many others .....!

decimal	decimal-leading-zero	lower-roman	upper-alpha	simp-chinese-formal
1. eggs	01. eggs	i. eggs	A. eggs	壹、eggs
2. milk	02. milk	ii. milk	B. milk	貳、milk
3. bread	03. bread	iii. bread	C. bread	叁、bread

Different List Item Markers

```
ul {
    list-style-type: disc;
}
```

- The **list-style-type** property specifies the type of list item marker

- Knight Rider
- A-Team

- **For unordered lists**
  - Several popular values: circle, disc, square, or none

html	default	disc	circle	square	none

`<ul>`
`<li>eggs</li>`
`<li>milk</li>`
`<li>bread</li>`
`</ul>`
	• eggs
	• milk
	• bread
		• eggs
		• milk
		• bread
			○ eggs
			○ milk
			○ bread
				▪ eggs
				▪ milk
				▪ bread
					eggs
					milk
					bread

week7-sampleCodes/stylingLists/stylingLists.html

Example:
Different List Item Markers for Ordered and Unordered Lists

```html
<!DOCTYPE html>
<html>
<head>
<style>
ul.a {
    list-style-type: circle;
}

ul.b {
    list-style-type: square;
}

ol.c {
    list-style-type: upper-roman;
}

ol.d {
    list-style-type: lower-alpha;
}
</style>
</head>
<body>

<p>Example of unordered lists:</p>
<ul class="a">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ul>
<ul class="b">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ul>
<p>Example of ordered lists:</p>
<ol class="c">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ol>
<ol class="d">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ol>
</body>
</html>
```

Example of unordered lists:

- Coffee
- Tea
- Coca Cola

- Coffee
- Tea
- Coca Cola

Example of ordered lists:

I. Coffee
II. Tea
III. Coca Cola

a. Coffee
b. Tea
c. Coca Cola

week7-sampleCodes/stylingLists/differentListMarkerItems.html

Set an Image as the List Item Marker

- ***Use a custom image instead of traditional marker!***
- The **list-style-image** property specifies an image as the list item marker

```html
<!DOCTYPE html>
<html>
<head>
<style>
ul {
    list-style-image: url("https://mdn.mozillademos.org/files/11981/starsolid.gif");
}
</style>
</head>
<body>
    <p>
        <ul>
          <li>Coffee</li>
          <li>Tea</li>
          <li>Coca Cola</li>
        </ul>
    </p>

</body>
</html>
```

week7-sampleCodes/stylingLists/imageAsListItemMarker.html

Styling Lists with Colors

- We can also style lists with colors, to make them look more interesting
- Anything added to the **<ol>** or **<ul>** tag, affects the **entire list**, while properties added to the **<li>** tag will affect the **individual list items**

```html
<!DOCTYPE html>
<html>
<head>
<style>
ol {
    background: #ff9999;
    padding: 20px;
}

ul {
    background: #3399ff;
    padding: 20px;
}

ol li {
    background: #ffe5e5;
    padding: 5px;
    margin-left: 35px;
}

ul li {
    background: #cce5ff;
    margin: 5px;
}
</style>
</head>

<body>
<h1>Styling Lists With Colors:</h1>
<ol>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ol>
<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ul>
</body>
</html>
```

Styling Lists With Colors:

1. Coffee
2. Tea
3. Coca Cola

- Coffee
- Tea
- Coca Cola

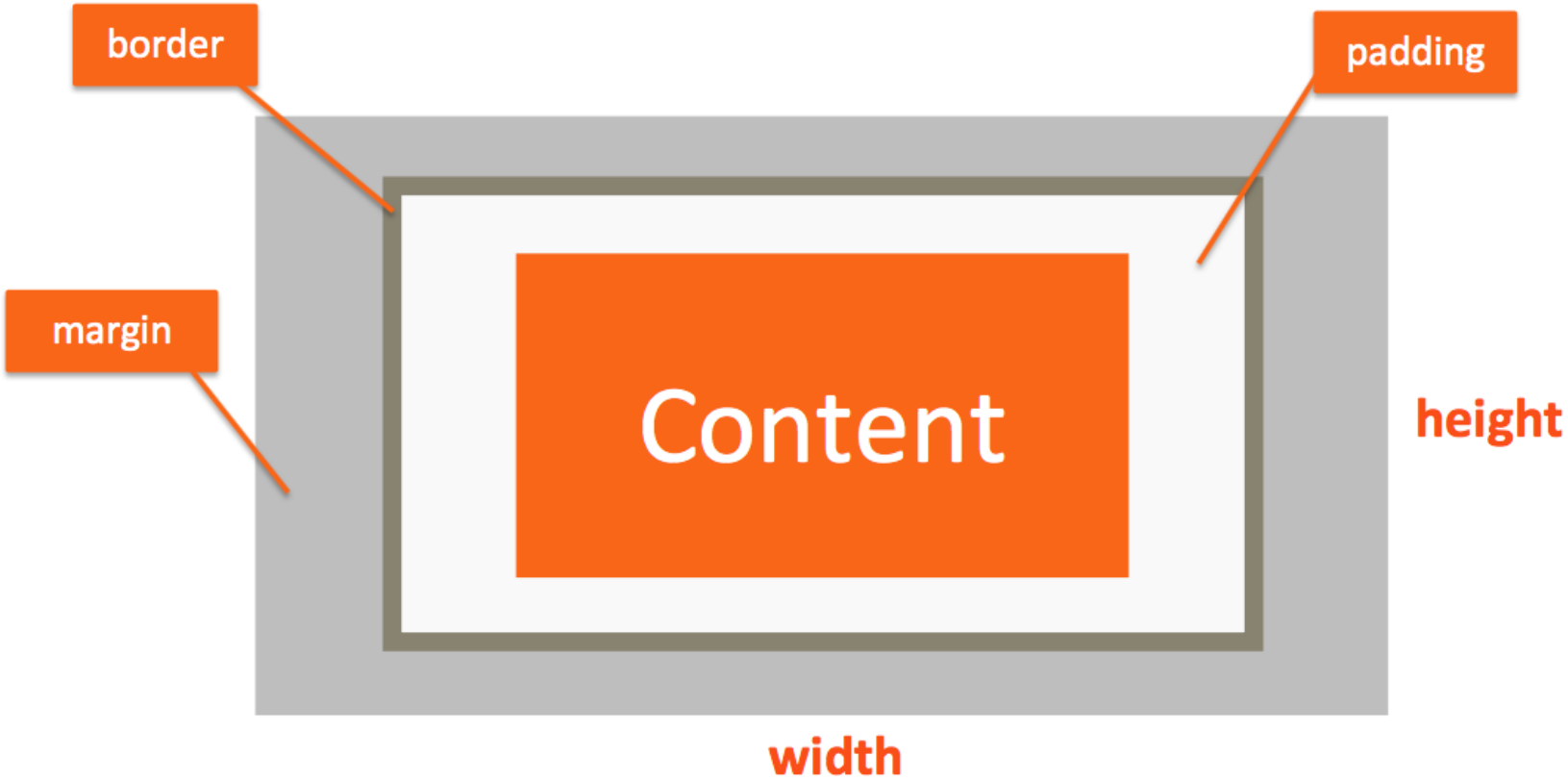
week7-sampleCodes/stylingLists/stylingListsWithColors.html

Outline

- CSS Selectors (cont.)
  - Pseudo-Class Selectors
- CSS Links
- CSS Selectors (cont.)
  - Pseudo-elements Selectors
  - Attribute Selectors
- CSS Lists
- **The Box Model**
  - **CSS Borders**
  - **CSS Margins**
  - **CSS Padding**
  - **CSS Height/Width**
  - **CSS Box Model**
- CSS Overflow

CSS Box Model

- All **HTML elements** can be considered as **boxes**
- In CSS, the term "**box model**" is used when talking about design and layout
- The CSS box model is essentially ***a box that wraps around every HTML element***
- Consists of
  - margins,
  - borders,
  - padding, and
  - the actual content



CSS Borders

The CSS **border** properties allow you to specify the **style**, **width**, and **color** of an element's border

I have borders on all sides.

I have a red bottom border

I have rounded borders.

I have a blue left border.

CSS Borders – Border Style

```
p {
    border-top-style: dotted;
    border-right-style: solid;
    border-bottom-style: dotted;
    border-left-style: solid;
}
```

- The **border-style** property specifies what kind of border to display
- The following values are allowed:
  - **dotted** - Defines a dotted border
  - **dashed** - Defines a dashed border
  - **solid** - Defines a solid border
  - **double** - Defines a double border
  - **groove** - Defines a 3D grooved border. The effect depends on the border-color value
  - **ridge** - Defines a 3D ridged border. The effect depends on the border-color value
  - **inset** - Defines a 3D inset border. The effect depends on the border-color value
  - **outset** - Defines a 3D outset border. The effect depends on the border-color value
  - **none** - Defines no border
  - **hidden** - Defines a hidden border

```
p {
    border-style: dotted solid;
}
```

- The **border-style** property can have from one to four values (for the **top border**, **right border**, **bottom border**, and the **left border**)

Example: Border Styles

```html
<!DOCTYPE html>
<html>
<head>
<style>
p.dotted {border-style: dotted;}
p.dashed {border-style: dashed;}
p.solid {border-style: solid;}
p.double {border-style: double;}
p.groove {border-style: groove;}
p.ridge {border-style: ridge;}
p.inset {border-style: inset;}
p.outset {border-style: outset;}
p.none {border-style: none;}
p.hidden {border-style: hidden;}
p.mix {border-style: dotted dashed solid
double;}
</style>
</head>
<body>

<h2>The border-style Property</h2>
<p>This property specifies what kind of
border to display:</p>

<p class="dotted">A dotted border.</p>
<p class="dashed">A dashed border.</p>
<p class="solid">A solid border.</p>
<p class="double">A double border.</p>
<p class="groove">A groove border.</p>
<p class="ridge">A ridge border.</p>
<p class="inset">An inset border.</p>
<p class="outset">An outset border.</p>
<p class="none">No border.</p>
<p class="hidden">A hidden border.</p>
<p class="mix">A mixed border.</p>

</body>
</html>
```

The border-style Property

This property specifies what kind of border to display:

A dotted border.

A dashed border.

A solid border.

A double border.

A groove border.

A ridge border.

An inset border.

An outset border.

No border.

A hidden border.

A mixed border.

CSS Borders – Border Width

- The **border-width** property specifies the **width** of the ***four borders***
  - The **border-width** property can have from one to four values (for the **top border**, **right border**, **bottom border**, and the **left border**)
- The **width** can be set as
  - A specific size (in px, em) or
  - By using one of the three pre-defined values: **thin**, **medium**, or **thick**

```
p.one {
    border-style: solid;
    border-width: 5px;
}

p.seven {
    border-style: solid;
    border-width: 2px 10px 4px 20px;
}
```

Solid Border Style with 5px border-width.

Some text.

CSS Borders – Border Color, Shorthand Property, Rounded Borders

- Border Color
  - **The border-color** property is used to set the **color** of the ***four borders***
- Border – Shorthand Property
  - There are many properties to consider when dealing with borders
  - To shorten the code, it is also possible to specify all the individual border properties in one property – the **border** property
    - **border-width**
    - **border-style** (required)
    - **border-color**
  - We can also specify all the individual border properties for **just one side**
    - **border-left**, **border-bottom**

```
p {
    border: 5px solid red;
}
```

Some text

```
p {
    border-left: 6px solid red;
    background-color: lightgrey;
}
```

Some text

- Rounded Borders
  - The **border-radius** property is used to add **rounded borders** to an element

```
p.round2 {
    border: 2px solid red;
    border-radius: 8px;
}
```

Rounder border

Tip: None of the OTHER CSS border properties described so far will have ANY effect unless the **border-style** property is set (it must be specified)!

CSS Padding

- The CSS **padding** properties are used to generate space around content
- **Padding** is ***additional space between the element and its border***
- The **padding** clears an area around the content (inside the border) of an element
- With CSS, we have full control over the padding
  - There are CSS properties for setting the padding for each side of an element (top, right, bottom, and left)
    - **padding-top**
    - **padding-right**
    - **padding-bottom**
    - **padding-left**

```
p {
    padding-top: 50px;
    padding-right: 30px;
    padding-bottom: 50px;
    padding-left: 80px;
}
```

  - Padding - Shorthand Property
    - The **padding** property is a shorthand property for above individual four padding prop

```
p {
    padding: 50px 30px 50px 80px;
}
```

  - All the padding properties can have the following values
    - ***length*** - specifies a padding in px, pt, cm, etc.
    - ***%*** - specifies a padding in % of the width of the containing element
    - **inherit** - specifies that the padding should be inherited from the parent element

Tip1: *Negative values are not* ***allowed!***

Tip2: *Paddings are transparent. Hence, they do not take color!!*

CSS Margins

- The CSS **margin** properties are used to generate space around elements (between you and neighbor)
- The **margin** properties ***set the size of the white space outside the border***
- With CSS, we have full control over the margins
  - There are CSS properties for setting the margin for each side of an element (top, right, bottom, and left)
    - **margin-top**
    - **margin-right**
    - **margin-bottom**
    - **margin-left**

```
p {
    margin-top: 100px;
    margin-right: 150px;
    margin-bottom: 100px;
    margin-left: 80px;
}
```

  - Margin - Shorthand Property
    - The **margin** property is a shorthand property for above individual four margin properties

```
p {
    margin: 100px 150px 100px 80px;
}
```

  - All the margin properties can have the following values
    - **auto** - the browser calculates the margin
    - ***length*** - specifies a margin in px, pt, cm, etc.
    - ***%*** - specifies a margin in % of the width of the containing element
    - **inherit** - specifies that the margin should be inherited from the parent element

Tip1: *Negative values are allowed!*

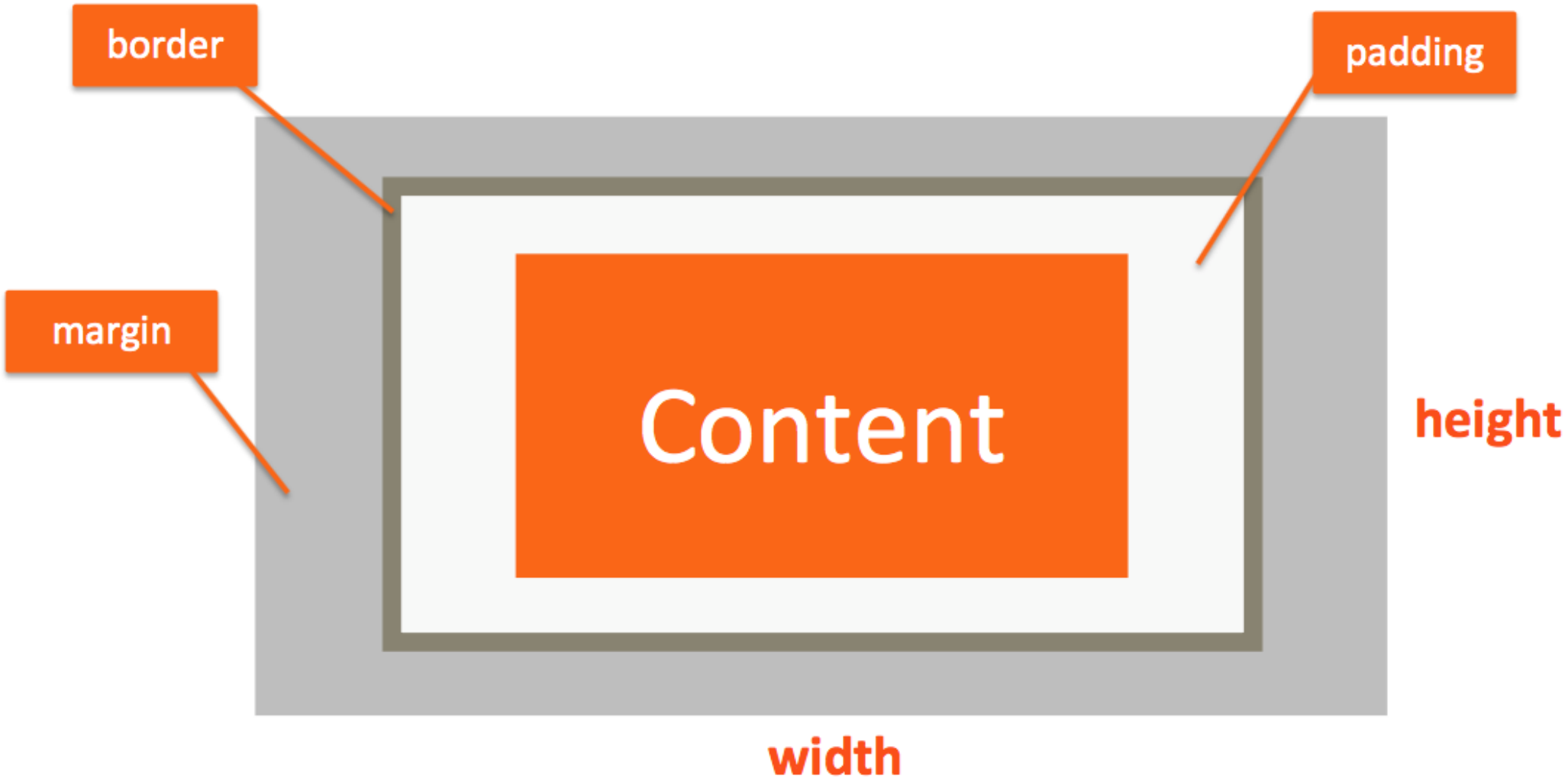
With positive margin elements move right/down. Whereas with negative margin elements move left/upward

Tip2: *Margins are transparent. Hence, they do not take a color!!*

CSS Height/Width

- The **height** and **width** properties are used to set the height and width of an element
- The **height** and **width** can be set to
  - **auto** – this is default → Means that the browser calculates the height and width)
  - Specified in ***length values***, like px, cm, etc.
  - Specified in **percent (%)** of the containing block
- ***In order to set the width and height of an element correctly in all browsers, we need to know how the box model works!!!***

The Box Model



Box-Sizing Property

- If **box-sizing** property has the value as **content-box** (which is ***default additive***)
  - When we set the **width** and **height** properties of an element with CSS, we just set the width and height of the **content area**
  - To calculate the full size of an element, we must also add padding, borders and margins

```
div {
    width: 100px;
    height: 50px;
    padding: 10px;
    border: 1px solid gray;
    margin: 5px;
}
```

What is the total width and height?

- The **total width** of an element should be calculated like :

Total element width = width + L padding + R padding + L border + R border + L margin + R margin

Total width = 100 + 10 + 10 + 1 + 1 + 5 + 5 = 132px

- The **total height** of an element should be calculated like:

Total element height = height + T padding + B padding + T border + B border + T margin + B margin

Total height = 50 + 10 + 10 + 1 + 1 + 5 + 5 = 82px

This means: *When we set the width/height of an element, the element often appear bigger than we have set (because the element's border and padding are added to the element's specified width/height)*

Box-Sizing Property

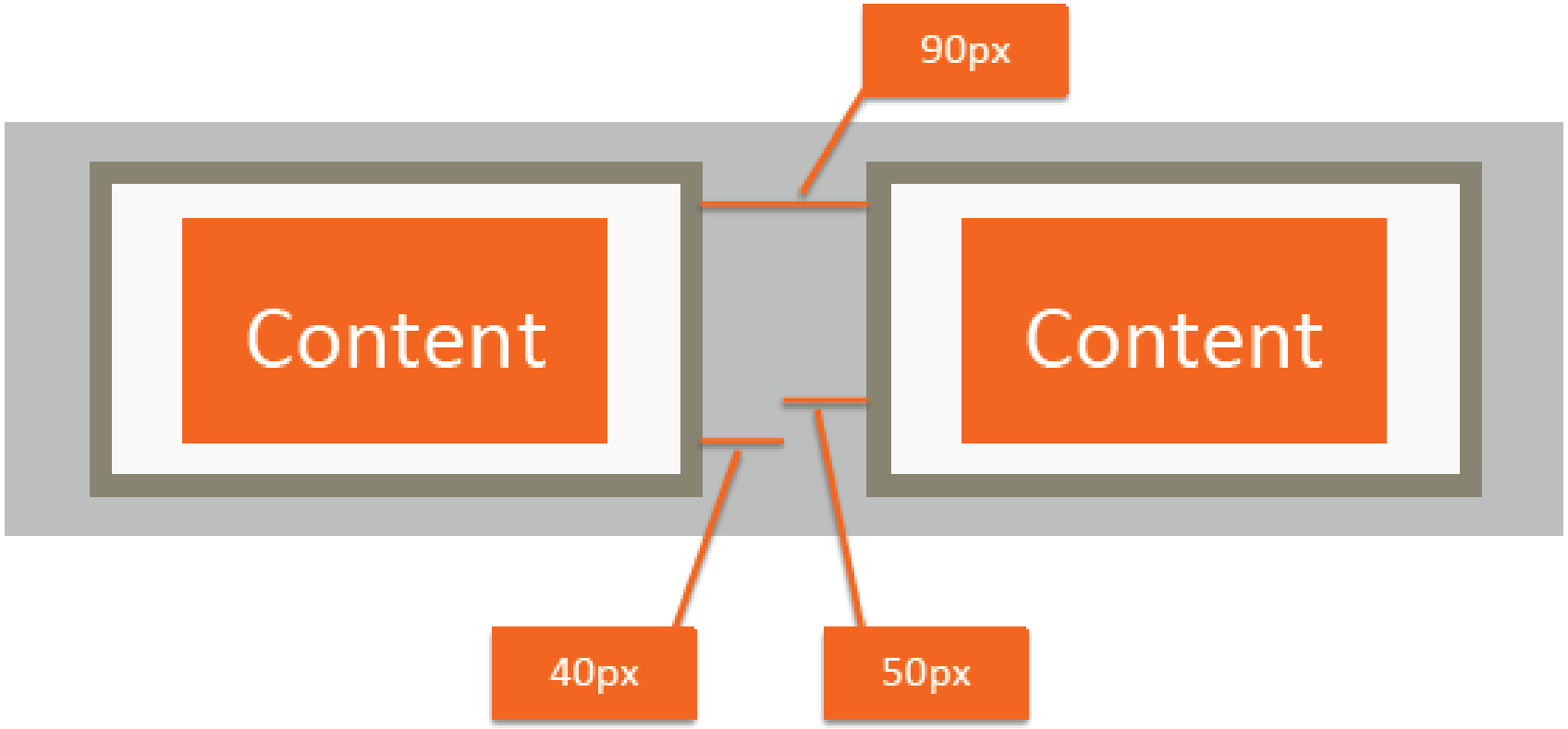
- With CSS3, the **box-sizing** property solves this problem (***prefer this!!!***)
- If we set **box-sizing: border-box;** on an element, padding and border are ***included*** in the **width** and **height**

```
div {
    width: 100px;
    height: 50px;
    padding: 10px;
    border: 1px solid gray;
    margin: 5px;
    box-sizing: border-box;
}
```

- Since the result of using the ***box-sizing: border-box;*** is so much better, many developers want **all elements** on their pages to work this way
  - The *** (universal) selector**

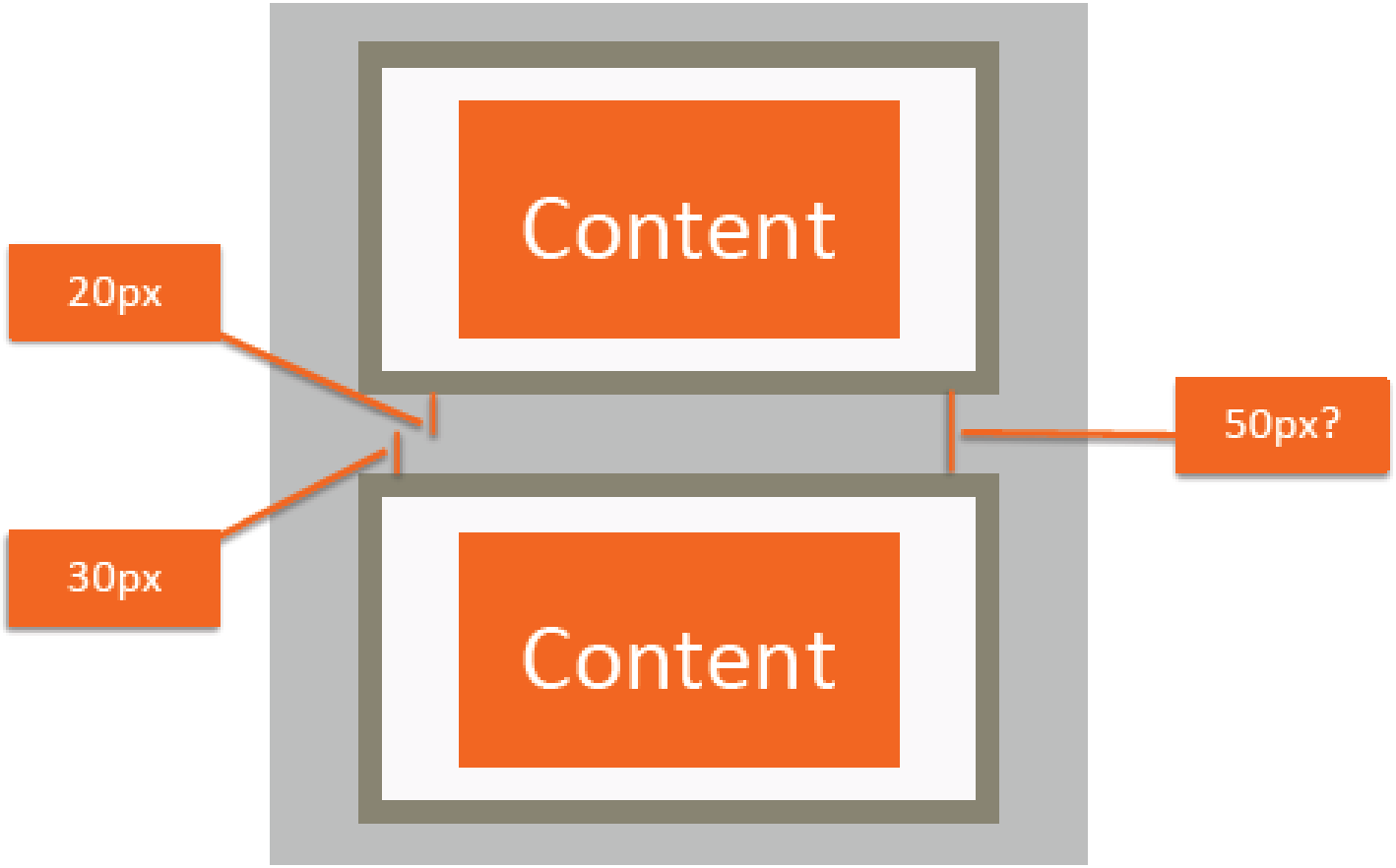
```
* {
    box-sizing: border-box;
}
```

Cumulative Margins

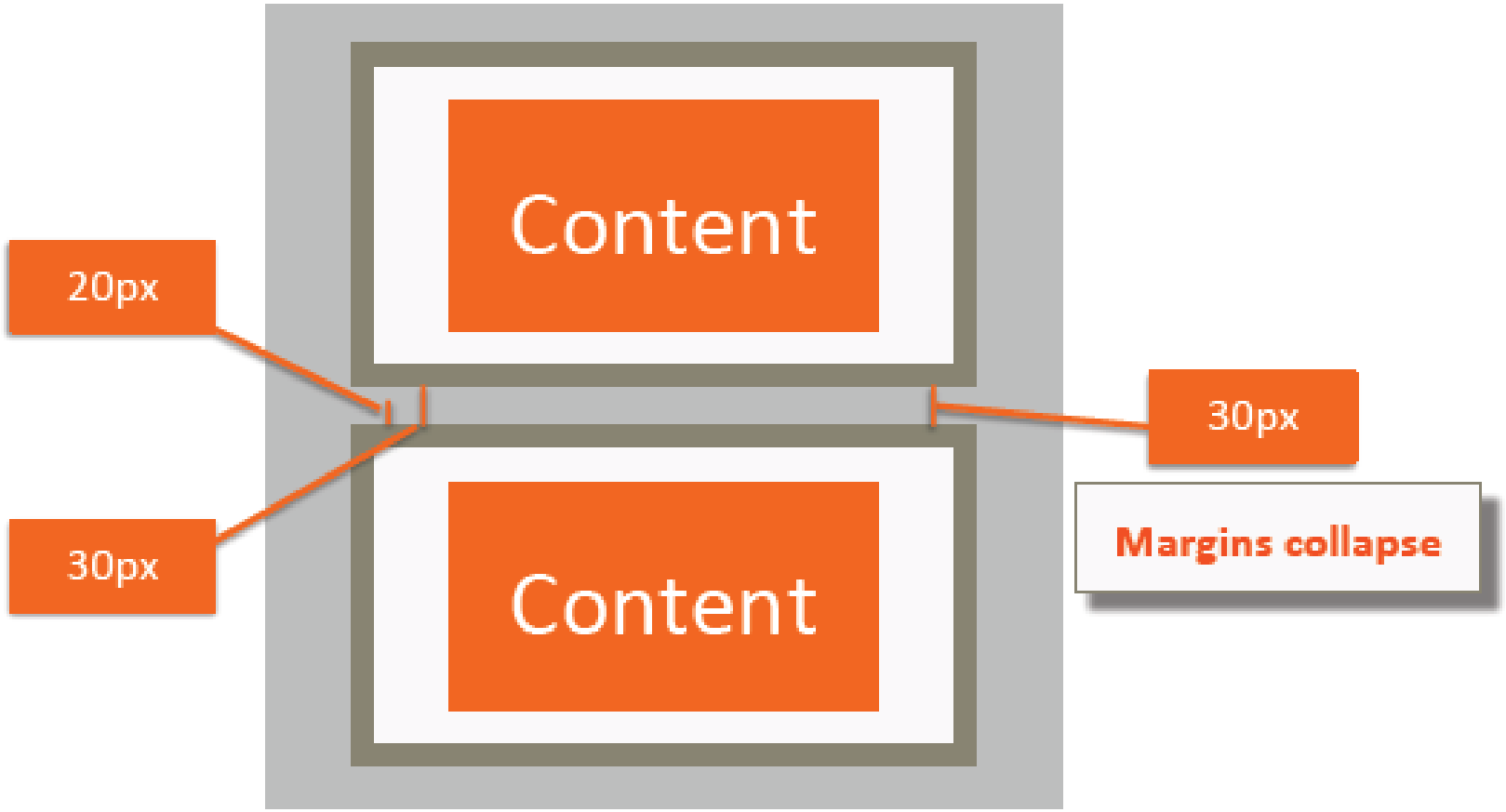


Margins that are left to right are cumulative

Cumulative Margins?



Cumulative Margins



The larger margin will win

Outline

- CSS Selectors (cont.)
  - Pseudo-Class Selectors
- CSS Links
- CSS Selectors (cont.)
  - Pseudo-elements Selectors
  - Attribute Selectors
- CSS Lists
- The Box Model
  - CSS Borders
  - CSS Margins
  - CSS Padding
  - CSS Height/Width
  - CSS Box Model
- **CSS Overflow**

CSS Overflow

- CSS **overflow** property specifies whether to clip content or to add scrollbars when the content of an element is too big to fit in a specified area
- The **overflow** property has the following values
  - **visible** – (default) The overflow is not clipped. It renders outside the element's box
  - **hidden** – The overflow is clipped, and the rest of the content will be invisible
  - **scroll** – The overflow is clipped, but a scrollbar is added to see the rest of the content (this will add a scrollbar both horizontally and vertically - even if you do not need it)
  - **auto** – If overflow is clipped, a scrollbar should be added to see the rest of the content (**<u>similar</u>** to scroll, only it add scrollbars when necessary)