Client-Side Web GIS

CSS

Image from NASA: https://commons.wikimedia.org/wiki/File:Earth_Western_Hemisphere_transparent_background.png#filelinks

Cascading Style Sheets (CSS) are used to style webpages and can also be used for responsive web design.

In this module, we discuss basic CSS. In the next module, we discuss expanding and simplifying CSS using the Bootstrap and w3.css frameworks.

Module Topics

1. Linking CSS to HTML
2. Inline vs. internal vs. external
3. Element, id, and class selectors
4. Precedence
5. Combination selectors
6. Pseudo-class selectors
7. CSS comments
8. Working with color
9. Working with type and font
10. Styling different types of HTML tags or content
11. Margins and padding
12. Overflow
13. Height and width
14. Max-width
15. Positioning
16. Float and clear
17. Block, inline, and inline-block

2

These are the topics covered in this module.

CSS Selectors

Where to store CSS

Inline = as an attribute in HTML tag

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Document</title>
</head>
<body>
   <p style="color:crimson;font-
size:24pt">Some text</p>
</body>
</html>
```

Generally not preferred

Internal = in style tags in HTML document, generally in header

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Document</title>
   <style>
      p{
         color:crimson;
         font-size:24pt;
      }
   </style>
</head>
<body>
   <p>Some text</p>
</body>
</html>
```

External = in separate .css file

```
<!DOCTYPE html>
<html lang="en">
<link rel="stylesheet“
 type="text/css" href="css_example.css">
<head>
   <meta charset="UTF-8">
   <title>Document</title>
</head>
<body>
   <p>Some text</p>
</body>
</html>
```

Generally preferred

4

CSS can be associated with HTML using three different methods.

First, CSS can be included within the tag of the HTML element that it is meant to style. This is done using a “style” argument followed by a list of CSS styles in quotes. This is the example on the left and is termed inline CSS.

The middle example represents including the CSS in a <style> tag within the HTML document. This is known as internal CSS. It is generally preferred to do this within the <head> tag; however, you can include style tags within the <body> as long as they are defined before they are used or referenced.

Lastly, you can store CSS in a separate file with a “.css” extension. This external CSS will then need to be associated with the HTML document using a <link> tag in the <head> tag. The link will be defined with the “href” argument. This can be a file within the website directory (for example, in the “style” folder) or CSS stored on another website or URL. We will talk about this in more detail when we cover Bootstrap and w3.css.

It is generally preferred to store your CSS in a separate file unless you are using it sparingly. I would suggest making an external file, storing it in your “style” folder, then referencing it. This will make for cleaner and easier to interpret HTML and CSS.

Selectors

- Used to find HTML elements based on element type, id, class, etc.

```
p{
   color:gray;
   font-size:24pt;
}

.para-style{
   color:darkgoldenrod;
   font-size:24pt;
}

#para-id{
   color:crimson;
   font-size:24pt;
}
```

```
<!DOCTYPE html>
<html lang="en">
<link rel="stylesheet" type="text/css"
href="css_example.css">
<head>
   <meta charset="UTF-8">
   <title>Document</title>
</head>
<body>
   <p>Some text</p>
   <p class="para-style">Some more text</p>
   <p id="para-id">Yet more text</p>
</body>
</html>
```

5

How do you apply styling to a specific element or a specific subset of elements?

This is accomplished using selectors that can be linked to or associated with specific HTML elements or groups of elements.

In the next few slides, we will talk about how this is accomplished and the options available.

Tag, Class, ID

- Element Selector = Styles can be applied to all elements of the same tag type
- Class Selector = Elements of the same class can be given the same style
- ID Selector = An id can be assigned to a specific element to assign a style to that specific element

```
p{
   color:gray;
   font-size:24pt;
}
```

```
.para-style{
   color:darkgoldenrod;
   font-size:24pt;
}
```

```
#para-id{
   color:crimson;
   font-size:24pt;
}
```

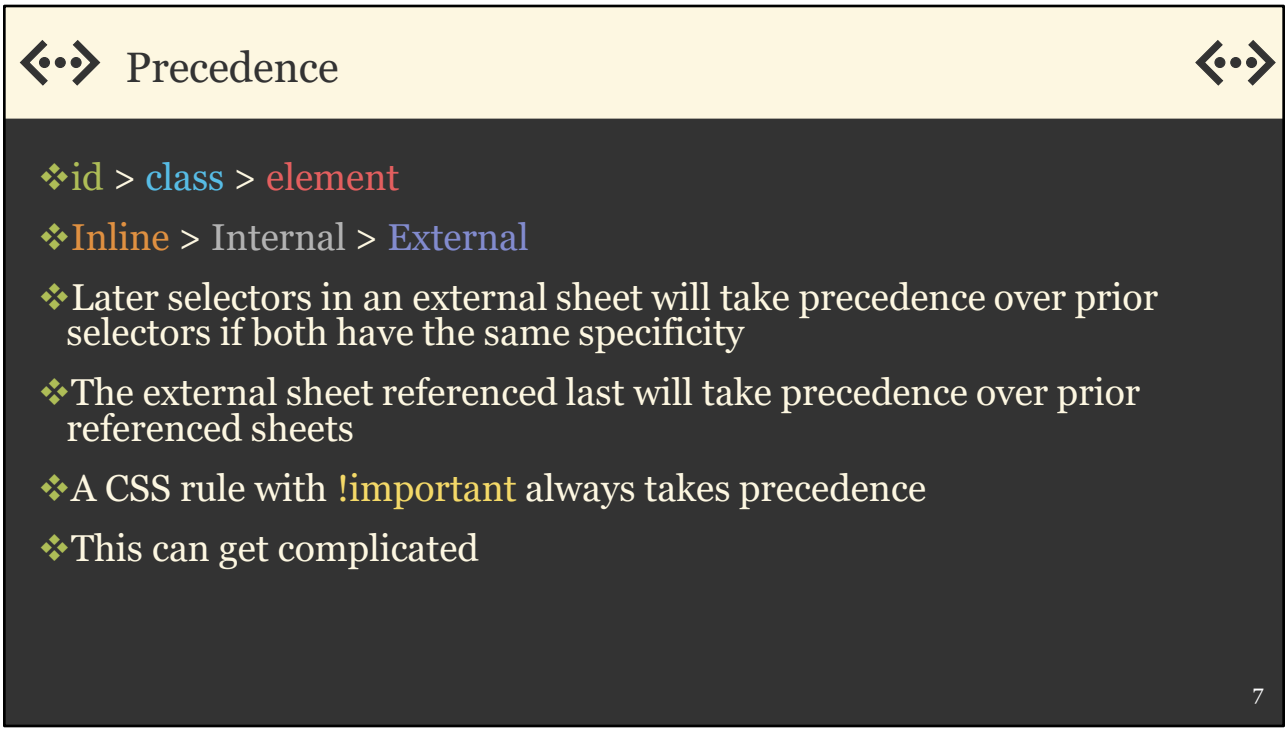
6

Element selectors are applied to all tags of the same type. The example on the left is an element selector for all paragraph, or <p>, tags in the HTML code.

A class selector can be used to apply styling to a subset of elements by associating each element with the CSS class. This is done using the “class” argument within the opening tag of the element to be styled.

ID selectors are used to style individual elements on a page. They are designed to be applied to only one element (for example, one paragraph or one image). Names cannot be reused. To associate the ID selector to an element, you will use the “id” argument within the opening tag.

Element selectors are defined just using the tag name (for example, p, h1, h2, h3, a, ol, ul, table, li, img, etc.). Class selectors must begin with a period while ID selectors begin with a pound or hashtag. When the selector is then referenced within a tag, the period or hash are not included.



It is possible that multiple CSS selectors could apply to the same element.

For example, a paragraph with a class defined could also accept styling from a <p> tag element selector.

So, how does the browser know which styles from which selectors to apply?

This is determined based on precedence. This slide highlights some important precedence rules.

First, ID selectors trump class selectors, which trump element selectors. Inline trumps internal which trumps external CSS.

In an external sheet, later defined selectors take precedence over earlier defined selectors. If you link to multiple external sheets, the last one to be referenced will be honored first.

You can make sure that something is honored by including !important after the selector.

When you use multiple external sheets and/or a combination of inline, internal, and external CSS, this can get complicated. If a selector is not

honored correctly, then another selector may be superseding it. Development tools can be useful for sorting this out.

Note that it is possible and common for elements to accept styling from multiple selectors. For example, if a class is assigned to a paragraph, the text color could still be defined by a <p> tag element selector as long as no color is defined in the class selector.

When Elements have the same properties

- Can combine using commas

```
.p1, .p2, .table1, .list1{
   font-size:40px;
   color: black;
   border: solid;
}
```

8

When multiple HTML elements have the same desired style, the styling can be defined at once as demonstrated on this slide.

Combination Selectors

- Can combine multiple selectors to be more specific

```
.para-div p{
   color:gray;
   font-size:24pt;
}

p{
   color:crimson;
   font-size:24pt;
}
```

```
<!DOCTYPE html>
<html lang="en">
<link rel="stylesheet" type="text/css" href="css_example.css">
<head>
   <meta charset="UTF-8">
   <title>Document</title>
</head>
<body>
  <div class="para-div">
   <p>Some text</p>
   <p>Some more text</p>
   </div>
   <p>Yet more text</p>
</body>
</html>
```

9

It is possible to combine selectors to be even more specific. These are known as combination selectors.

In the example, two selectors are defined: a combination selector and an element selector.

The combination selector is set up to be applied to only paragraphs occurring in tags with the “para-div” class defined. So, the first two paragraphs take this styling, since they are paragraphs and in a assigned to the “para-div” class. In contrast, the last paragraph takes on the style defined by the element selector since it is a paragraph but not in the associated with the “para-div” class.

Combination Selectors

- Descendant Selector = for all descendants of a specific element

```
.para-div p{
   color:gray;
   font-size:24pt;
}
```

- Child Selector = for only immediate children of a specific element

```
.para-div > p{
   color:gray;
   font-size:24pt;
}
```

- Adjacent Sibling Selector = for only the adjacent sibling occurring after a specific element

```
.para-div + p{
   color:gray;
   font-size:24pt;
}
```

- General Sibling Selector = for all elements that are siblings of a specific element

```
.para-div ~ p{
   color:gray;
   font-size:24pt;
}
```

10

This slide describes a variety of combination selector subtypes. The subsequent video will explain these in more detail.

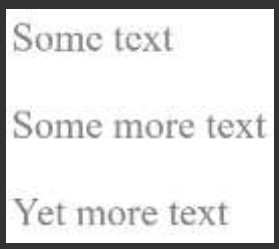
Descendent selectors are meant to be applied to all descendants of a specific element. In the example, the styling will be applied to all paragraphs included in or descending from a assigned to the “para-div” class. We saw this example in the prior slide.

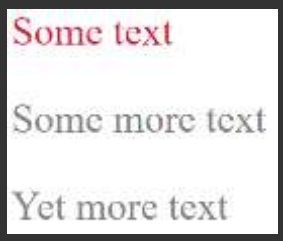
Child selectors only apply to immediate children or descendants of a specific element as opposed to all descendants. In the example, the styling will be applied to all paragraphs that are direct descendants of a assigned to the “para-div” class. If a subsequent was defined within this , paragraphs within it would not be styled using this selector since they are not immediate children.

Adjacent sibling selectors are only applied to an adjacent sibling occurring after a specific element. In the example, the styling will be applied to the paragraph following the assigned to the “para-div” class. Note that this is not a paragraph within the but a paragraph at the same level as the in the HTML document. In contrast to adjacent sibling selectors, general sibling selectors will apply to all siblings of the defined element, not just the next, adjacent sibling.

Pseudo-Class Selectors

- Defines special state of an element
  - Style an element when user mouses over it
  - Style visited and unvisited links differently
  - Style an element when it gets focus






```
p{
   color:gray;
   font-size:24pt;
}

p:hover{
   color:crimson;
   font-size:24pt;
}
```

11

Pseudo-class selectors are used to apply styling relative to a specific state.

For example, text could change color when it is moused over, which is commonly used for links.

So, the element will only take on this styling when the specific state is met.

Example Pseudo-Class Selectors

Selector	Description
:active	Selects the active link
:focus	Selects element that has focus
:hover	Selects links on mouse over
:link	Selects all unvisited links
:invalid	Selects elements with an invalid value
:valid	Selects elements with a valid value
:visisted	Selects all visited links

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

12

Here are some examples of different pseudo-class selectors.

Visit the w3schools.com link to see a full list with descriptions.

CSS Comments

```
p{
   color:gray;
   /*This is a comment*/
   font-size:24pt;
}

/*
This is
another
comment
*/
```

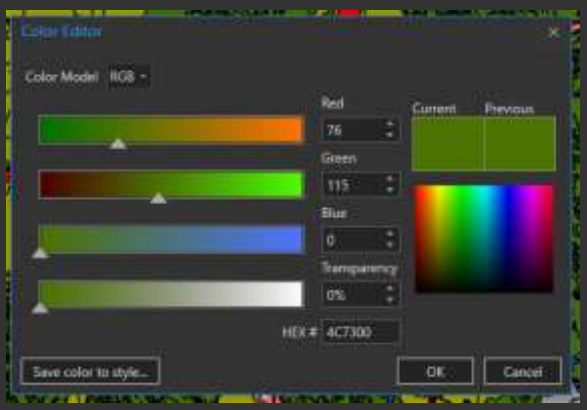
13

This slide demonstrates how to define comments within a CSS document or style tag. Note that this is different from the method used for defining HTML comments or JavaScript comments.

Working With Colors

RGB

Example from Adobe Illustrator



Example from ArcGIS Pro

Example from QGIS

15

One common use of CSS is to define colors (for example, text color, background color, border color, line color, etc.)

There are multiple ways to define colors using CSS. Generally, web colors are defined using additive color space as opposed to subtractive color space since we are dealing with monitors and projected light as opposed to inks and print.

Additive colors are generally defined based on relative proportions of the additive primaries: red, green, and blue.

It is common to define colors using 8-bit color space where 256 gray or brightness levels are available for each primary color. By mixing relative portions of these 0 to 255 values, many colors can be defined (256 X 256 X 256 = more than 16 million!).

Hex Codes

Integer	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Hexadecimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

#ff0000

(15 * 16) + 15 = 255

(0 * 16) + 0 = 0

(0 * 16) + 0 = 0

(255, 0, 0) = Red

#ffff00

(15 * 16) + 15 = 255

(15 * 16) + 15 = 255

(0 * 16) + 0 = 0

(250, 0, 0) = Yellow

#B6AAC2

(11 * 16) + 6 = 182

(10 * 16) + 10 = 170

(12 * 16) + 2 = 194

(182, 170, 194)

Example from ArcGIS Pro

16

Instead of using RGB values as 0 to 255, colors can also be defined using hex codes.

As the slide describes, 0 through 9 indicate 0 through 9 while A through F indicate 10 through 15.

The first two characters define the red value, the middle two define the green value, and the last two define the blue value.

In each pair, the first value in multiplied by 16 and then the second value is added to get a 0 to 255 value for each primary color.

So, this offers a means to define an 8-bit RGB color using six characters.

Note that when you write hex codes you commonly start with a pound sign or hashtag.

Let's go through the examples on the slide.

The first example represents red since the first two values yield 255 and the remaining pairs yield 0 and 0. So, full red, no blue or green.

The middle example represents yellow: full red plus full green yields yellow.

The last example is a mix of red, green, and blue and yields the color in which the resulting RGB values are displayed.

Note that many software packages and websites allow you to find hex codes from RGB values or RGB values from hex codes including ArcGIS Pro, Adobe Photoshop, and Adobe Illustrator.

RGBA

R = Red (0-255)

G = Green (0-255)

B = Blue (0-255)

A = Alpha (0-1 or 0 to 100%)

A controls transparency/opacity

Example from ArcGIS Pro

17

RGBA includes an alpha channel along with the RGB channels to control transparency.

The RGB values range from 0 to 255 while alpha varies from 0 to 1 or 0% to 100%, where 0 or 0% indicates fully transparent while 1 or 100% yields fully opaque.

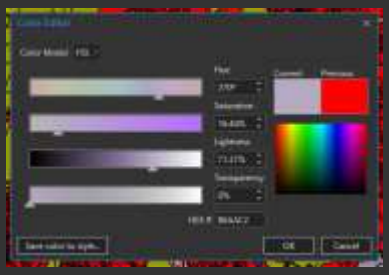
Hue, Saturation, Lightness

Hue ~ base color

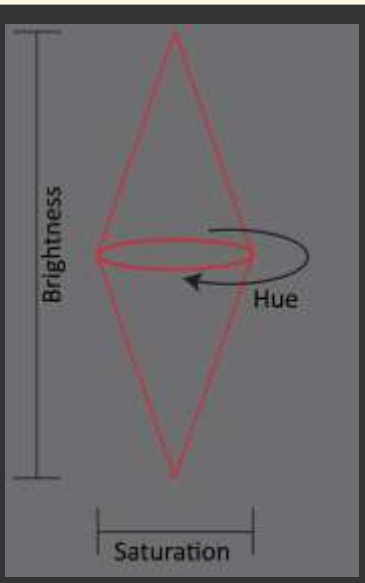
Saturation ~ dullness/color mixed with gray

Lightness ~ intensity, lightness/darkness

Example from Adobe Illustrator



Example from ArcGIS Pro



18

An alternative means to represent color is using the hue, intensity, lightness, or HSL, space.

In this space, a color is defined using a base color (hue), a saturation (mixing the base color with gray), and a lightness (full brightness to full dark or white to black).

This space is often visualized as two cones that share a common base. The hue is represented as rotating around the cone, the saturation corresponds to moving toward or away from the center of the cone, and the brightness corresponds to moving up and down the cone.

Hue is commonly defined as an angle between 0 and 360 while saturation and lightness are scaled from 0 to 1 or 0% to 100%. It is also possible to include an alpha or transparency argument (HSLA).

Defining Colors in CSS

- Named colors = https://www.w3schools.com/colors/colors_names.asp
- rgb(R, G, B)
- Hex Code: #ffffff
- hsl(H, S, L)
- rgba(R, G, B, A)
- hsla(H, S, L, A)

```
body{
    background-color: rgb(55, 54, 61);
    font-family: "Tahoma";
    font-size: 14pt;
}
```

19

The slide describes how colors are defined in CSS using these different methods.

Note that there are named colors that can be called as a string without quotes. A list of these colors is available at the provided w3schools.com link.

If you use hex codes, you must include a pound sign or hash tag.

For rgb(), hsl(), rgba(), and hsla(), you must provide numbers within the defined or appropriate range.

Note that colors can also be specified using subtractive color space (CYMK); however, we will not discuss that method in this web course, as subtractive space is commonly used for printing while additive space is used for projected light, such as computer monitors.

I generally prefer to define colors using hex codes or named colors, but that is just personal preference.

Change Background Color


```
body{
   background-color: darkgreen;
   font-family: "Tahoma";
   font-size: 14pt;
}
```


```
body{
   background-color: #37363D;
   font-family: "Tahoma";
   font-size: 14pt;
}
```

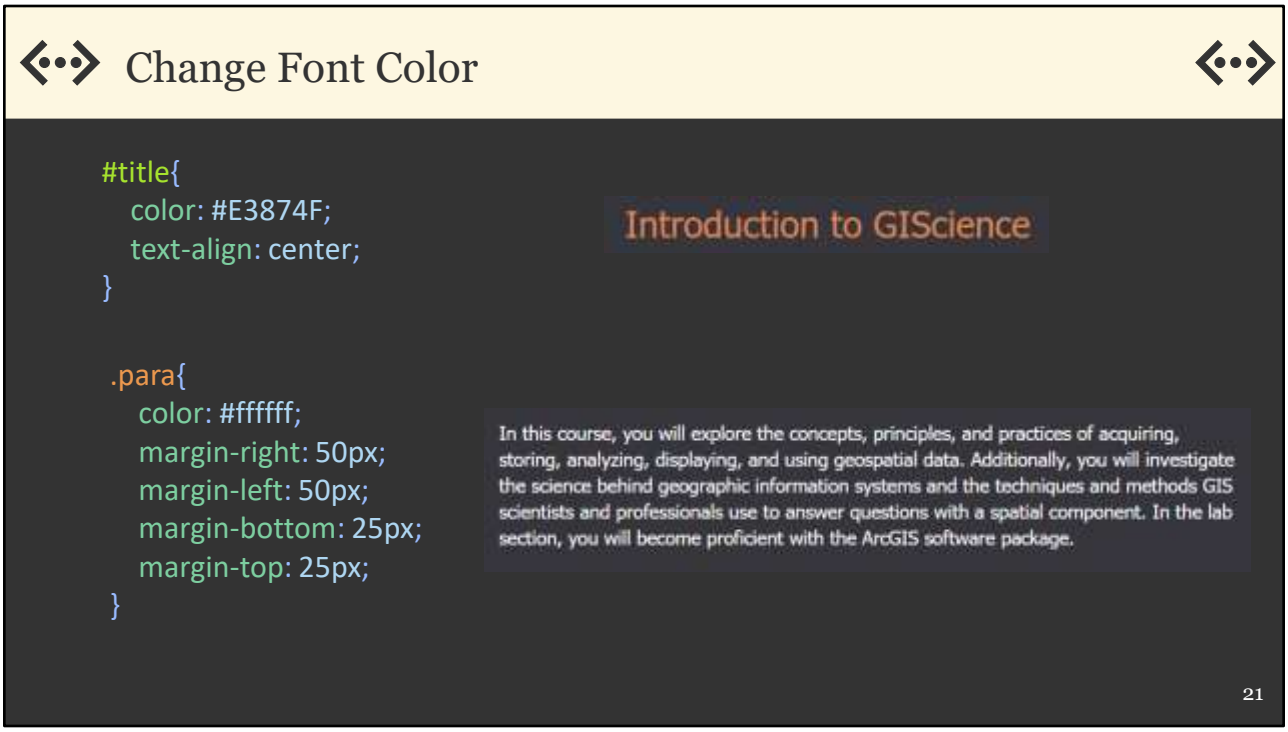
20

We will now work through a variety of examples relating to some common styling operations.

First, the background color can be set using the “background-color” argument. Here, I am defining the background color using a named color and a hex code.

I am also using an element selector applied to the entire <body> tag, so this will set the background color for the entire page.

Using additional selectors and precedence, you can change the background color for portions of the page, such as the nav bar or a specific .



Text color can be set using the “color” argument.

In the example, I am defining the text color for the title using a hex code. This is only applied to the single HTML element to which the “title” ID selector is applied.

The paragraph text is defined as white using the appropriate hex code. Specifically, this would only apply to paragraphs assigned to the “para” class selector.

Note that if you do not explicitly define background and/or text colors, defaults will be used. Generally, this is black text on a white background.

Working with Type

Type

Type Family = group of type designs that reflect common design characteristics and share a common base name (Calibri, Arial, Comic Sans, Cambria, Courier New, Times New Roman, Verdana)

Type Style = Roman (normal), bold, *italic* (Less common: condensed, expanded, light, and extra bold)

Type Face = Family + Style (Gill Sans Ultra Bold)

Type Size = height (11 pt, 14 pt, 18 pt)

Font = a set of all alphanumeric and special characters of a particular family, style, and size (Times New Roman 12 pt)

23

Here is a review of some of the characteristics of fonts and type.

Take some time to read through these descriptions.

Case

Title Case = lowercase letters with the first letter of each word set to upper case (except linking words like in, on, or, of, per, by, for, with, the, and, over, etc.)

Sentence case = Only first letter of first word in the sentence is capitalized

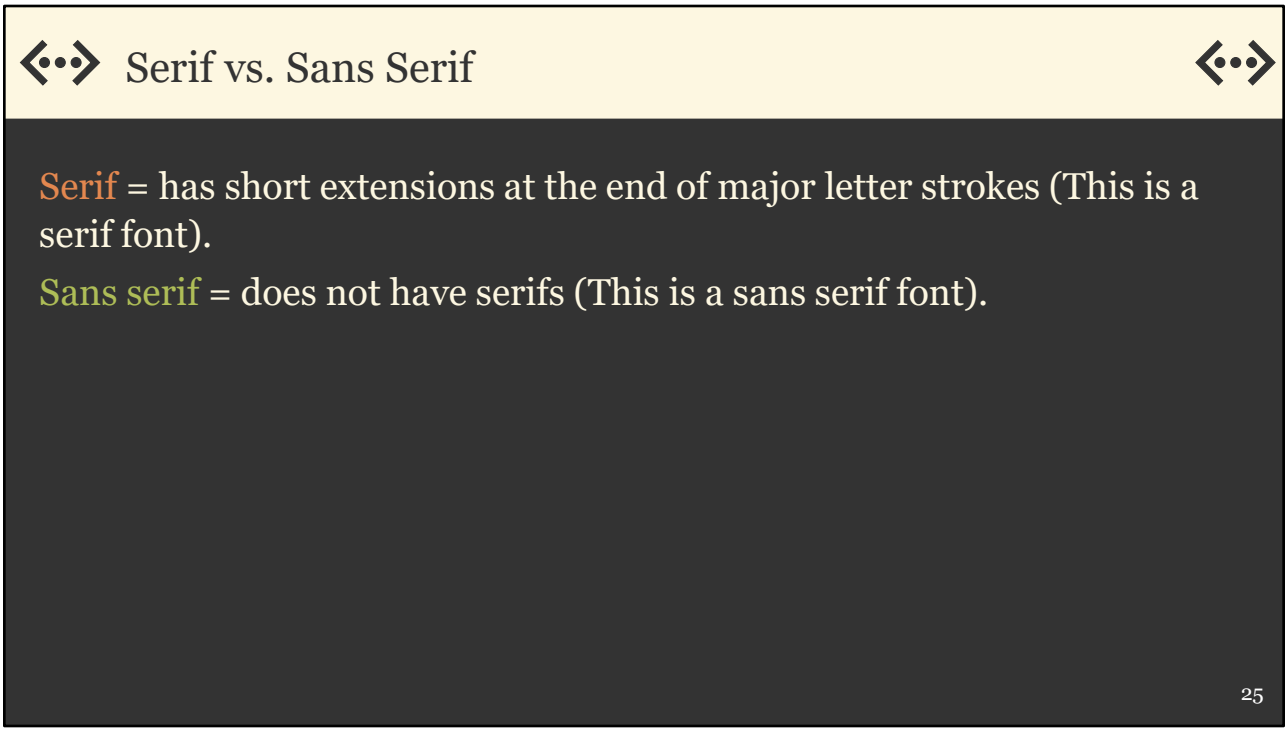
UPPERCASE = All caps

24

Title case indicates capitalizing only the first letter of each word, except linking words.

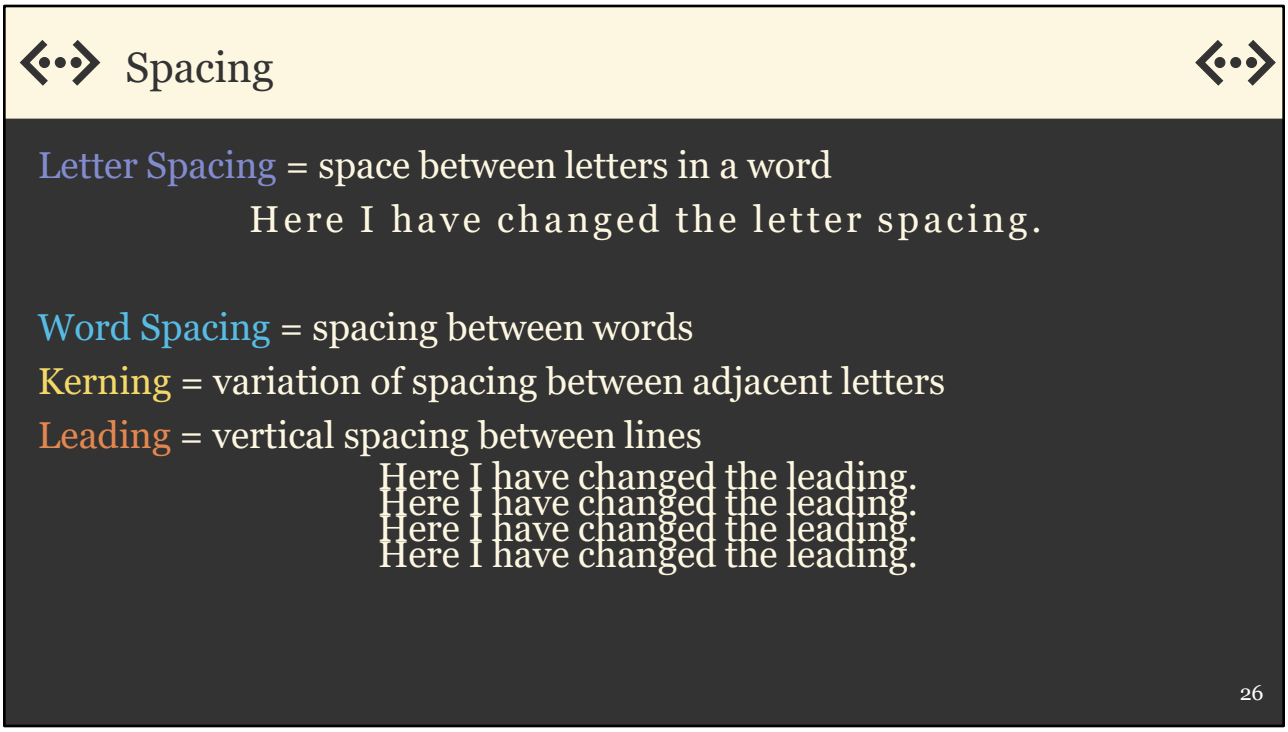
Sentence case means to capitalize only the first letter of the first word.

Uppercase implies that everything is capitalized.



Serif fonts have serifs, or extensions at the end of major letter stokes.

San serif fonts do not have serifs.



Letter spacing refers to the spacing between letters in a word while word spacing relates to spacing between words.

Kerning relates to variation of spacing between adjacent letters. Letter spacing adjusts spacing between letters consistently while kerning applies specific spacing depending on the letter combination in question.

Leading relates to spacing between lines of text.

Change Font

- Can provide multiple font families as a list just in case browser does not have access to chosen font

```
p {
  font-family: "Times New Roman", Times, serif;
  font-style: italic;
  font-size: 50px;
  font-weight: bold;
  color: brown;
}
```

27

A variety of font characteristics can be defined. Note that if you do not define a font, a default font will be used.

It is common to define multiple fonts just in case a specific font is not available on a client machine. In this example, “Times New Roman” is the preferred font. However, if this is not available, then an available “Times” font should be used. If a “Times” font is not available, then an available “serif” font should be used.

I am also specifying a font style, size, weight, and color.

Change Font

Property	Description
font	Set all font properties in one declaration
font-family	Specifies the font family for text
font-size	Specifies the font size of text
font-style	Specifies the font style for text
font-variant	Specifies whether or not text should be displayed in a small-caps font
font-weight	Specifies the weight of a font

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

1em = 12pt = 16px = 100%

28

This slide provides a list of font properties available in CSS, which are documented by w3schools.com at the provided link.

When defining sizes of text and other HTML features, there are different units of measurement that can be used.

This slide shows the equivalency or conversion factors between em, pt, px, and %.

Both em and percent are relative. The default font for the page is equivalent to 1 em or 100%.

px (pixel) and pt (point) are not relative and do not scale with changing window sizes.

Styling HTML Elements

Links

```
a:link {
  color: black;
  background-color:dimgray;
}

/* mouse over link */
a:hover {
  color: red;
  background-color:dimgray;
}
```

```
<!DOCTYPE html>
<html lang="en">
<link rel="stylesheet" type="text/css" href="css_example.css">
<head>
   <meta charset="UTF-8">
   <title>Document</title>
</head>
<body>
   <a href="http://maxwellae.wixsite.com/maxwell-
geospatial">Link</a>
</body>
</html>
```

30

This slide demonstrates CSS applied to a link.

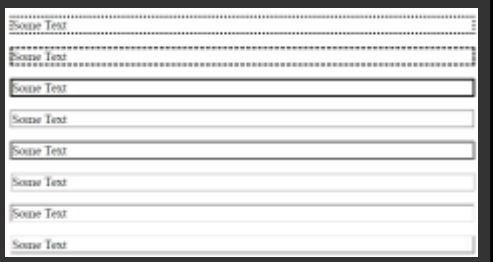
Here, I am using pseudo-class selectors. The first selector defines styling for a link that has not been visited while the second one defines styling when a link is moused over.

These selectors will be applied to all <a> tags.

Border Style

```
.dotted{
   border-style: dotted;}
.dashed{
   border-style: dashed;}
.solid{
   border-style: solid;}
.double{
   border-style: double;}
.groove{
   border-style: groove;}
.ridge{
   border-style: ridge;}
.inset{
   border-style: inset;}
.outset{
   border-style: outset;}
```

```
<!DOCTYPE html>
<html lang="en">
<link rel="stylesheet“
 type="text/css" href="css_example.css">
<head>
   <meta charset="UTF-8">
   <title>Document</title>
</head>
<body>
   <p class="dotted">Some Text</p>
   <p class="dashed" >Some Text</p>
   <p class="solid" >Some Text</p>
   <p class="groove" >Some Text</p>
   <p class="double">Some Text</p>
   <p class="ridge" >Some Text</p>
   <p class="inset">Some Text</p>
   <p class="outset" >Some Text</p>
</body>
</html>
```



31

This slide provides some examples of styling options for borders of HTML elements.

The “border-style” option defines the line style. It is also possible to change the color and thickness.

Border Width and Color

```css
.brd-style{
   border-style: solid;
   border-width: 7px;
   border-color: red;
   background-color: gray;
}
```

```html
<!DOCTYPE html>
<html lang="en">
<link rel="stylesheet" type="text/css" href="css_example.css">
<head>
   <meta charset="UTF-8">
   <title>Document</title>
</head>
<body>
   <p class="brd-style" >Some Text</p>
</body>
</html>
```

In this example, I am further augmenting a border.

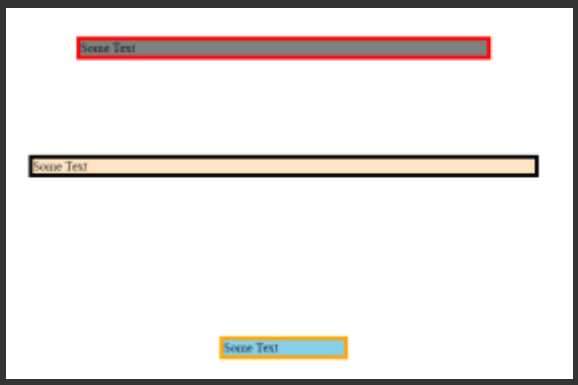
The paragraph is assigned to the “brd-style” class.

I then define a border style (solid), border width in pixels (7px), border color (red), and the background color (gray).

Margins

- Create space around elements, outside of any defined borders

```
.margin1{
   border-style:solid ;border-width: 5px;
   background-color: gray; border-color: red;
   margin-top: 80px; margin-bottom: 80px;
   margin-right: 120px; margin-left: 120px;}
.margin2{
   border-style:solid; border-width: 5px;
   background-color: bisque; border-color: black;
   margin-top: 120px; margin-bottom: 120px;
   margin-right: 60px; margin-left: 60px;}
.margin3{
   border-style:solid; border-width: 5px;
   background-color: skyblue; border-color: orange;
   margin-top: 200px; margin-bottom: 200px;
   margin-right: 300px; margin-left: 300px;}
```



33

The concepts of margins and padding are commonly confused when using CSS.

Margin adds space around an element and outside of any defined borders while padding adds space within the borders.

This slide demonstrates margins.

Margins can be styled collectively using “margin” or individually using “margin-top”, “margin-bottom”, “margin-right”, and/or “margin-left.”

In this example, I have applied different margin setting to three <p> tags. Note that the gaps are applied between the elements as opposed to within each element.

Padding

- Create space around an element's content, inside of any defined borders

```
.margin1{
   border-style:solid; border-width: 5px;
   background-color: gray; border-color: red;
   padding-top: 80px; padding-bottom: 80px;
   padding-right: 120px; padding-left: 120px;}
.margin2{
   border-style:solid; border-width: 5px;
   background-color: bisque; border-color: black;
   padding-top: 120px; padding-bottom: 120px;
   padding-right: 60px; padding-left: 60px;}
.margin3{
   border-style:solid; border-width: 5px;
   background-color: skyblue; border-color: orange;
   padding-top: 200px; padding-bottom: 200px;
   padding-right: 300px; padding-left: 300px}
```

34

This example replicates the prior example using padding as opposed to margins. Similar to margins, all paddings can be defined collectively using "padding" or individually using "padding-top", "padding-bottom", "padding-left", and "padding-right".

Now, the space is added within each element as opposed to between elements.

Overflow

- Specify what to do if content overflows its container

Option	Description
visible	Content renders outside of element's box (default)
hidden	Overflow content is clipped
scroll	Scrollbar added once contain overflows
auto	Adds scrollbar only when necessary

35

When the contents of an element cannot be stored within the defined space, the overflow option can be applied to specify the desired behavior.

Using the "visible" option, content will render outside of the element box while "hidden" causes the content to be clipped to the extent of the box.

"Scroll" will add a scrollbar to the container while "auto" will add a scrollbar only when necessary.

The default is "visible".

Height and Width

- Set height and width of an element
- Default is auto (Browser calculates height and width)
- Can specify in px, length units, or % of containing block

```
<!DOCTYPE html>
<html lang="en">
<link rel="stylesheet" type="text/css" href="css_example.css">
<head>
   <meta charset="UTF-8">
   <title>Document</title>
</head>
<body>
  <div class="container-div">
    <p class="margin1">Some Text</p>
    <p class="margin2">Some Text</p>
    <p class="margin3">Some Text</p>
  </div>
</body>
</html>
```

```
.container-div{
    background-color: burlywood;
    width: 500px;
    height: 200px;
}
```

36

The dimensions of an element, such as a , can be defined using the "width" and "height" options. It is common to define height and width using pixels or a percent of the containing object. For example, a within the <body> that has a width of 50% defined will only stretch across half of the page, regardless of the size of the viewport.

By default, "width" and "height" are set to "auto", which allows the browser to calculate these dimensions. Generally, the browser will adjust elements to fill the entire width of its container and have the minimum height required to hold its content. For example, if a contains multiple paragraphs, then the will take on the required height to hold all the text content.

Max-Width

- Set a maximum width of an element
- Can be used to avoid browser adding scrollbar to the page

```
.container-div{
    background-color: burlywood;
    max-width: 1200px;
    height: 200px;
}
```

37

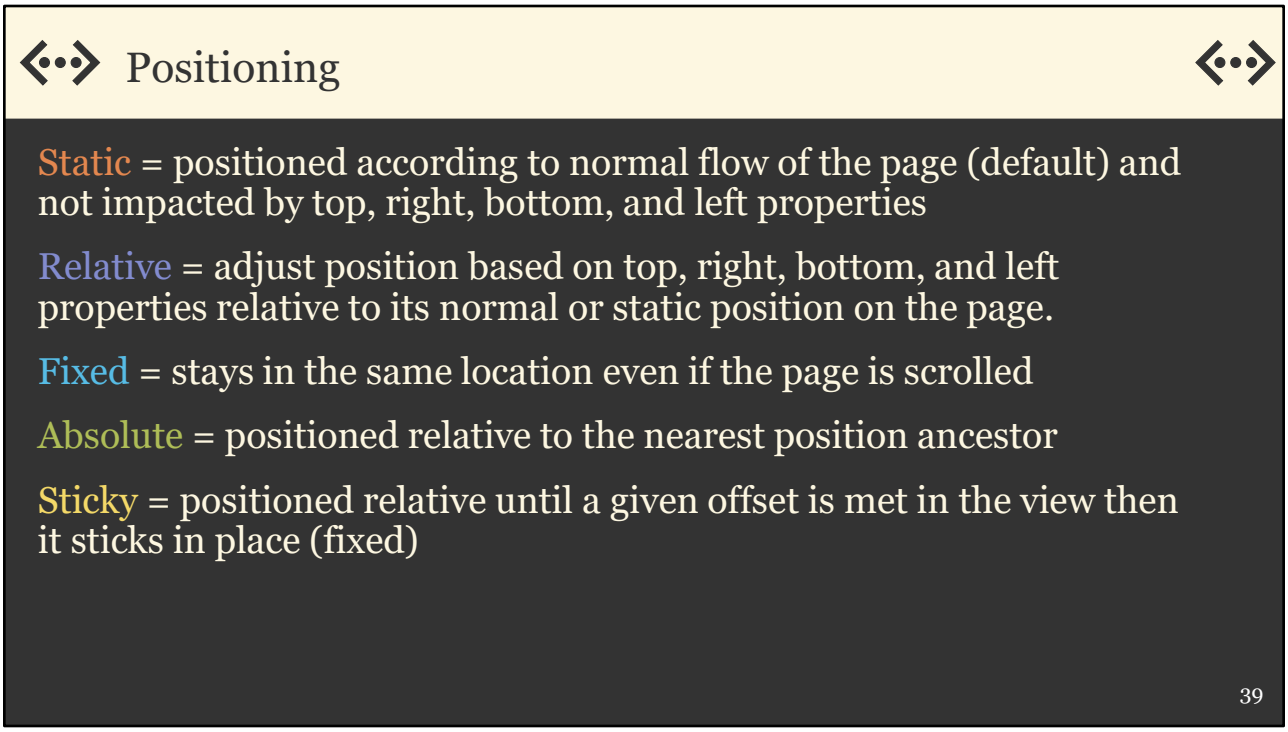
It is also possible to set a “max-width”. This will allow the browser to change the width, but only up to a certain width.

In the provided example, the “max-width” is set to 1200px for the shown with a tan background color. So, the will stretch across the full window until the window reaches a size of 1200 px. Then the will not get wider on the page.

One common use of the “max-width” argument is to stop the browser from adding a horizontal scrollbar to a page.

In this example, I am also specifying the “height” for the as 200 px. That is why the is larger than would be required to just hold the text contained in the paragraphs.

CSS for Positioning



I find positioning using CSS to be a bit confusing, but this is partially because there are a variety of options available which allow for a high degree of customization.

This slide describes some options for specifying positions.

Static positioning is the default and allows elements to be placed in the normal flow of the page. Static positioning will ignore “top”, “right”, “bottom”, and “left” properties.

Relative positioning allows for the user to define “top”, “right”, “bottom”, and “left” properties to position the element relative to its normal or static position within the viewport. For example, you could move an element left relative to its normal or static position with the “left” argument.

Fixed positioning causes an object to stay at the same location in the viewport even if the page is scrolled. For example, this can be used to make a nav bar always stay at the top of the page even if the page is scrolled up or down.

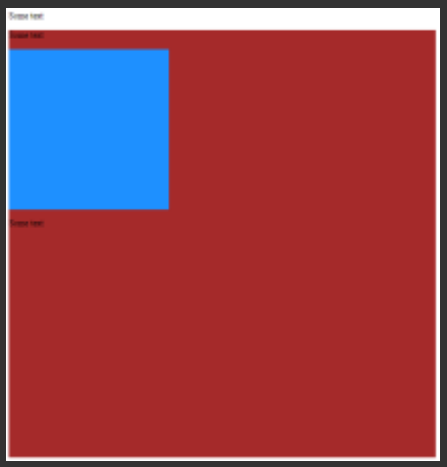
Absolute is used to define the position relative to the nearest position ancestor (for example, position a paragraph relative to the that contains it) or, if it does not have an ancestor, position it relative to the <body>.

Sticky combines relative and fixed. The element will maintain a relative position within the viewport until a given offset distance is met. Once this distance is met, the object will then stick in place or take on a fixed positioning.

Blue and Red Div Static

```
<!DOCTYPE html>
<html lang="en">
<link rel="stylesheet" type="text/css"
href="css_example.css">
<head>
   <meta charset="UTF-8">
   <title>Document</title>
</head>
<body>
 <p>Some text</p>
   <div class="div1">
      <p>Some text</p>
      <div class="div2">
      </div>
      <p>Some text</p>
   </div>
</body>
</html>
```

```
.div1{
   background-color:brown;
   height:800px;
   width: 800px;
   position:static;
}

.div2{
   background-color:dodgerblue;
   width: 300px;
   height: 300px;
   position: static;
}
```



40

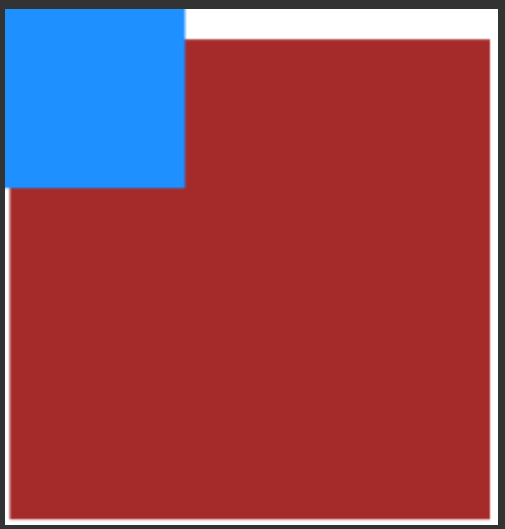
Let's now look at some specific examples of positioning. In this first example, the "div1" class selector is applied to the red while the "div2" class selector is applied to the blue .

Both <divs> are positioned using static. So, they occur in the normal flow of the page within the viewport. The blue occurs within the red since the red contains it.

Absolute Blue and Red Div Static

```
<!DOCTYPE html>
<html lang="en">
<link rel="stylesheet" type="text/css" href="css_example.css">
<head>
   <meta charset="UTF-8">
   <title>Document</title>
</head>
<body>
 <p>Some text</p>
   <div class="div1">
      <p>Some text</p>
      <div class="div2">
      </div>
      <p>Some text</p>
   </div>
</body>
</html>
```

```
.div1{
   background-color:brown;
   height:800px;
   width: 800px;
   position:static;
}

.div2{
   background-color:dodgerblue;
   width: 300px;
   height: 300px;
   position: absolute;
   top:0;
   left: 0;
}
```



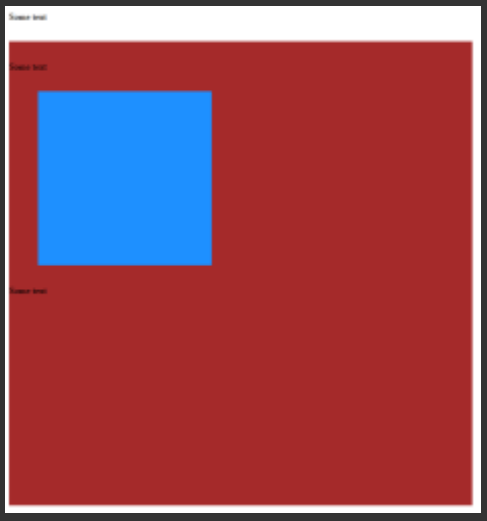
41

In this example, the red is still static while the blue is positioned absolute in the view port. Since the “top” and “left” arguments are set to 0, it is positioned in the upper-left corner of the page as opposed to being positioned within the normal flow of the page, as in the prior example.

Blue Relative and Red Static

```
<!DOCTYPE html>
<html lang="en">
<link rel="stylesheet" type="text/css"
href="css_example.css">
<head>
   <meta charset="UTF-8">
   <title>Document</title>
</head>
<body>
  <p>Some text</p>
   <div class="div1">
      <p>Some text</p>
      <div class="div2">
      </div>
      <p>Some text</p>
   </div>
</body>
</html>
```

```
.div1{
   background-color:brown;
   height:800px;
   width: 800px;
   position:static;
}

.div2{
   background-color:dodgerblue;
   width: 300px;
   height: 300px;
   position: relative;
   top:0;
   left: 50px;
}
```



42

Now, the blue is being positioned using relative. In this case, it is positioned relative to its normal flow in the page using the “top” and “left” arguments.

Specifically, I have moved the left by 50px.

Float and Clear

- Float specifies how an element should float
  - left: float to left of container
  - right: float to right of container
  - none: will not float (default)
  - inherit: inherits float value of parent
- Clear specifies what elements can float beside the cleared element and on which side
  - none: allows float elements on both sides
  - left: no floating elements on left side
  - right: no floating elements on right side
  - both: no floating elements on either the right or left side
  - inherit: the element inherits the clear value of its parent

43

Float can also be used to position and format page content within the viewport. Float is commonly used to position images or graphics relative to text.

Float “left” and “right” will cause the element to float to the left or right of its container. “None” is the default and will cause the element to be positioned in its normal location relative to other elements. “Inherit” will cause the element to inherit or take on the float assigned to its parent object (for example, an image within a).

Clear determines what elements can float beside the cleared element and on which side.

“None” indicates that elements can float to the left or right of the element. “left” will not allow elements to float to the left while “right” will not allow elements to float to the right. “both” indicates that elements cannot float to the left or right. “inherit” indicates that the feature will inherit the clear setting of its parent element.

Image Float Right

```
<!DOCTYPE html>
<html lang="en">
<link rel="stylesheet" type="text/css"
href="css_example.css">
<head>
   <meta charset="UTF-8">
   <title>Document</title>
</head>
<body>
   <p><img src="dt_2.JPG">Lorem ipsum dolor sit
amet, consectetur adipisicing elit. Non accusantium
accusamus sed sapiente soluta magnam recusandae,
doloribus quis assumenda voluptatem aut. Aut ipsa hic
nihil natus excepturi facere, quia eveniet?</p>
</body>
</html>
```

```
img{
   max-width:500px;
   float: right;
}

p{
   font-size:40px;
}
```

44

Now, let's look at some examples of float and clear.

Here, we are placing an element to the right of text using float "right".

Image Float Left

```
<!DOCTYPE html>
<html lang="en">
<link rel="stylesheet" type="text/css"
href="css_example.css">
<head>
   <meta charset="UTF-8">
   <title>Document</title>
</head>
<body>
   <p><img src="dt_2.JPG">Lorem ipsum dolor sit
amet, consectetur adipisicing elit. Non accusantium
accusamus sed sapiente soluta magnam recusandae,
doloribus quis assumenda voluptatem aut. Aut ipsa hic
nihil natus excepturi facere, quia eveniet?</p>
</body>
</html>
```

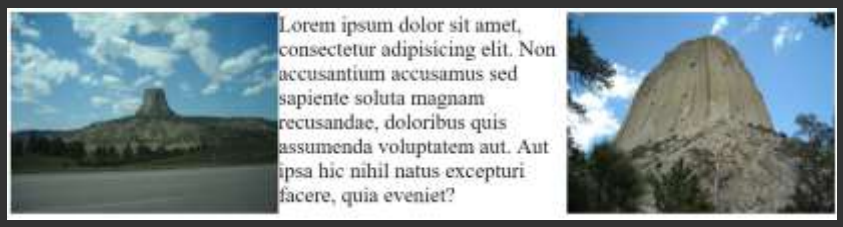
```
img{
   max-width:500px;
   float: left;
}

p{
   font-size:40px;
}
```

45

In this example, the image floats to the left of the text.

Image Float Left/Other Float Right

```
<!DOCTYPE html>
<html lang="en">
<link rel="stylesheet" type="text/css"
href="css_example.css">
<head>
   <meta charset="UTF-8">
   <title>Document</title>
</head>
<body>
   <p><img src="dt_2.JPG" class="img1"><img
src="dt_1.JPG" class="img2">Lorem ipsum dolor sit
amet, consectetur adipisicing elit. Non accusantium
accusamus sed sapiente soluta magnam
recusandae, doloribus quis assumenda voluptatem
aut. Aut ipsa hic nihil natus excepturi facere, quia
eveniet?</p>
</body>
</html>
```




```
.img1{
   max-width:500px;
   float: left;
}

.img2{
   max-width:500px;
   float: right;
}
```

46

Now, I am allowing images to float to the right or left of the text by applying "left" to one image and "right" to the other.

Clear Float

```
<!DOCTYPE html>
<html lang="en">
<link rel="stylesheet" type="text/css" href="css_example.css">
<head>
   <meta charset="UTF-8">
   <title>Document</title>
</head>
<body>
   <p class="p1"><img src="dt_2.JPG" class="img1">Lorem ipsum dolor sit amet, consectetur adipisicing elit. Non accusantium accusamus sed sapiente soluta magnam recusandae, doloribus quis assumenda voluptatem aut. Aut ipsa hic nihil natus excepturi facere, quia eveniet?</p>
   <p class="p2">Lorem ipsum dolor sit amet, consectetur adipisicing elit. Voluptate officia sit, nobis sed officiis corporis impedit nihil dignissimos facilis temporibus a perferendis earum consequuntur voluptas assumenda, sapiente ad, facere? Suscipit?</p>
</body>
</html>
```

```
.img1{
   max-width:500px;
   float: right;
}

.p1{
   font-size:40px;
}

.p2{
   font-size:40px;
   clear: right;
}
```

47

Now, I am using clear “right” so that the second paragraph is rendered below the image in the viewport. In other words, an image can no longer float to the right of this paragraph element.

Blocks vs. Inline

- Block
  - Starts on new line and take up the full width available
  - Default for , <h1> through <h6>, <p>, <form>, <header>, <footer>, <section>
- Inline
  - Does not start on a new line and only takes up as much width as necessary
  - Default for , <a>, <img>
- Can override the default for an element using CSS

48

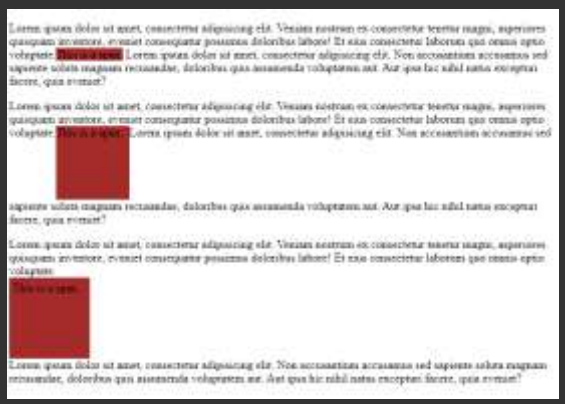
Some HTML elements are by default specified to start on a new line. These are known as block-level elements and some examples are provided on the slide.

In contrast, inline elements by default will not render to a new line and will take up as much width as necessary on the page. Some examples are provided here.

CSS can be used to override or alter these defaults to provide further customization of page layouts and positioning.

Inline-Block

- Allows for element width and height to be defined
- Can set top and bottom margin and padding
- Does not add line break after element



49

The display argument can also be of value for positioning elements.

The first span in the example is displayed using the default “inline” option. Here, the does not get rendered to a new line. Any height, width, and top or bottom margins or padding arguments are ignored.

In contrast, the second example uses “inline-block”. This is similar to “inline” in that the is still rendered to the same line. However, any height, width, and top or bottom margins or padding arguments are now honored. That is why the red span now has a larger extent as defined by the width and height arguments.

Display “block” will cause the element to act as a block-level element and render to a new line.

Center an Image


```
.img-style{
   max-width:400px;
   display: block;
   margin-left:auto;
   margin-right:auto;
}
```

```
<!DOCTYPE html>
<html lang="en">
<link rel="stylesheet" type="text/css"
href="css_example.css">
<head>
   <meta charset="UTF-8">
   <title>Document</title>
</head>
<body>
   <p>Some text</p>
   <img src="dt_2.JPG" class="img-style">
   <p>Some text</p>
</body>
</html>
```

50

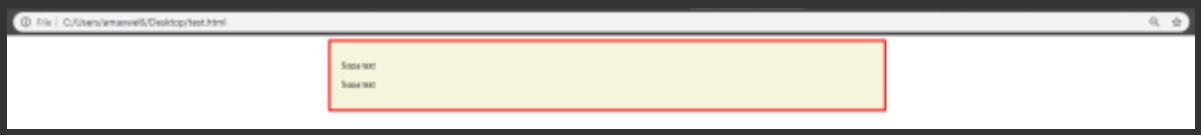
In this example an image is centered within a viewport. First, a “max-width” is set so that the image cannot fill the full width of the viewport if it is larger than 400px. Next, “display” is set to “block” so that the image acts like a block-level element. To center the image, the left and right margins are set to “auto.”

So, by combining “block” display with right and left “auto” margins, the image can be centered in the viewport.

Center a Div

```css
.div1{
  margin: auto;
  width: 40%;
  padding: 20px;
  border: solid;
  border-color: red;
  background-color: beige;
}
```

```html
<!DOCTYPE html>
<html lang="en">
<link rel="stylesheet" type="text/css" href="css_example.css">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
</head>
<body>
 <div class="div1">
     <p>Some text</p>
     <p>Some text</p>
 </div>
</body>
</html>
```



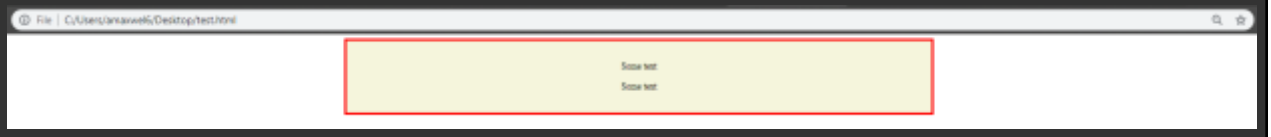
51

This is a similar example in which a is centered using “auto” margins.

Center Text in a Centered Div

```
.div1{
   margin: auto;
   width: 40%;
   padding: 20px;
   border: solid;
   border-color: red;
   background-color: beige;
}

.p1{
text-align: center;
}
```

```
<!DOCTYPE html>
<html lang="en">
<link rel="stylesheet" type="text/css" href="css_example.css">
<head>
   <meta charset="UTF-8">
   <title>Document</title>
</head>
<body>
 <div class="div1">
      <p class="p1">Some text</p>
      <p class="p1">Some text</p>
 </div>
</body>
</html>
```



52

Expanding upon the last example, I am now centering the text within the centered .

Text can be centered relative to its container or the <body> by setting the “text-align” option to “center”.

Move Div Right using Absolute

```
.div1{
   position: absolute;
   right: 0px;
   width: 300px;
   padding: 20px;
   border: solid;
   border-color: red;
   background-color: beige;
}

.p1{
text-align: center;
}
```

```
<!DOCTYPE html>
<html lang="en">
<link rel="stylesheet" type="text/css" href="css_example.css">
<head>
   <meta charset="UTF-8">
   <title>Document</title>
</head>
<body>
 <div class="div1">
      <p class="p1">Some text</p>
      <p class="p1">Some text</p>
 </div>
</body>
</html>
```

53

Here, I am moving the to the left by setting the “position” argument to “absolute”. I then define the right positon as 0px so that it moves to the right within the viewport.

Move Div Left using Float

```css
.div1{
   float: left;
   width: 300px;
   padding: 20px;
   border: solid;
   border-color: red;
   background-color: beige;
}

.p1{
text-align: center;
}
```

```html
<!DOCTYPE html>
<html lang="en">
<link rel="stylesheet" type="text/css" href="css_example.css">
<head>
   <meta charset="UTF-8">
   <title>Document</title>
</head>
<body>
 <div class="div1">
      <p class="p1">Some text</p>
      <p class="p1">Some text</p>
 </div>
</body>
</html>
```

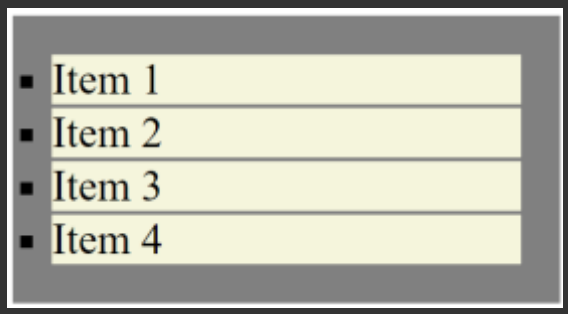
54

In this example, the is moved to the left of the viewport by setting “float” to “left”.

Format a List

```css
ul{
    list-style-type: square;
    background-color: gray;
    padding: 50px;
}

li{
    font-size:60px;
    background-color: beige;
    margin: 5px;
}
```

```html
<!DOCTYPE html>
<html lang="en">
<link rel="stylesheet" type="text/css" href="css_example.css">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
    <ul>
        <li>Item 1</li>
        <li>Item 2</li>
        <li>Item 3</li>
        <li>Item 4</li>
    </ul>
</body>
</html>
```



55

On this slide, I have provided an example of styling applied to a list object.

The unordered list is assigned a padding and background color. I am also using the “list-style-type” to change the bullet symbol used.

The individual list elements are styled using another selector to set the font size, background color, and margin. The 5px margin adds a small gap between each item in the list.

Format a Table

```css
th, td {
  padding: 8px;
  text-align: left;
  border-bottom: 1px solid #ddd;
  font-size: 30px;
}

tr:hover {background-color: #f5f5f5;}
```

```html
<!DOCTYPE html>
<html lang="en">
<link rel="stylesheet" type="text/css" href="css_example.css">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
</head>
<body>
  <table>
    <tr>
      <th>A</th>
      <th>B</th>
      <th>C</th>
    </tr>
    <tr>
      <td>1A</td>
      <td>1B</td>
      <td>1c</td>
    </tr>
    <tr>
      <td>2A</td>
      <td>2B</td>
      <td>2C</td>
    </tr>
    <tr>
      <td>3A</td>
      <td>3B</td>
      <td>3C</td>
    </tr>
    <tr>
      <td>4A</td>
      <td>4B</td>
      <td>4C</td>
    </tr>
  </table>
</body>
</html>
```

A	B	C
1A	1B	1c
2A	2B	2C
3A	3B	3C
4A	4B	4C

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

56

This is an example for formatting a table. The header and cells are styled by defining a padding, text alignment, bottom border, and font size.

I then use a pseudo selector to change style for a row when it is hovered over.

Create Navigation Bar

- Can create with inline-block

```
.nav {
  background-color: dimgray;
  border: solid;
  border-color: black;
  list-style-type: none;
  text-align: center;
  margin: 0;
  padding: 0;
}
.nav li {
  display: inline-block;
  font-size: 20px;
  padding: 20px;
}
```

```
<!DOCTYPE html>
<html lang="en">
<link rel="stylesheet" type="text/css" href="css_example.css">
<head>
   <meta charset="UTF-8">
   <title>Document</title>
</head>
<body>
<ul class="nav">
   <li><a href="http://maxwellae.wixsite.com/maxwell-geospatial">Home</a></li>
   <li><a href="http://maxwellae.wixsite.com/maxwell-geospatial">Courses</a></li>
   <li><a href="http://maxwellae.wixsite.com/maxwell-geospatial">Prof Maxwell</a></li>
   <li><a href="http://maxwellae.wixsite.com/maxwell-geospatial">Contact</a></li>
</ul>
</body>
</html>
```

57

This slide provides an example of how to generate a nav bar using CSS.

First, I create a to house the nav bar and style it by defining a background color, border, border color, bullet to use, text alignment, margin, and padding. Note that “list-style-type” is set to “none” so that no bullets are displayed.

Next, a list is generated to store each nav bar components. I then apply a padding and font size. An important change is to set “display” to “inline-block” so that the list items are arranged horizontally as opposed to vertically.

We will look at more nav bar options in the next module using Bootstrap or w3.css.

Learn More

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

58

I have tried to cover some of the common styling tasks used in web development; however, there are lots of options and styling tasks that can be performed.

For even more examples, please visit this w3schools.com link.

We did not cover responsive web design here. We will do so in the next module when we discuss Bootstrap and w3.css.

This is the end of this lecture module.

Please return to the West Virginia View Webpage for additional content.

Thanks! Hope you found this useful.