Client Side Technologies

CSS

(Cascade Style Sheets)

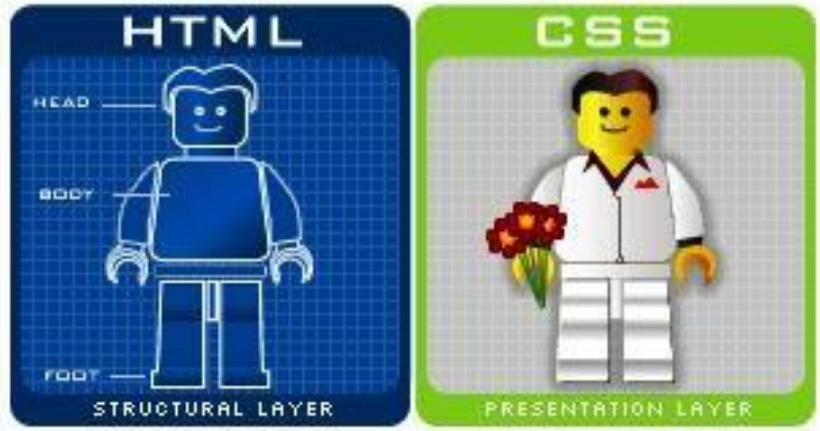
Dorothy Guirgues

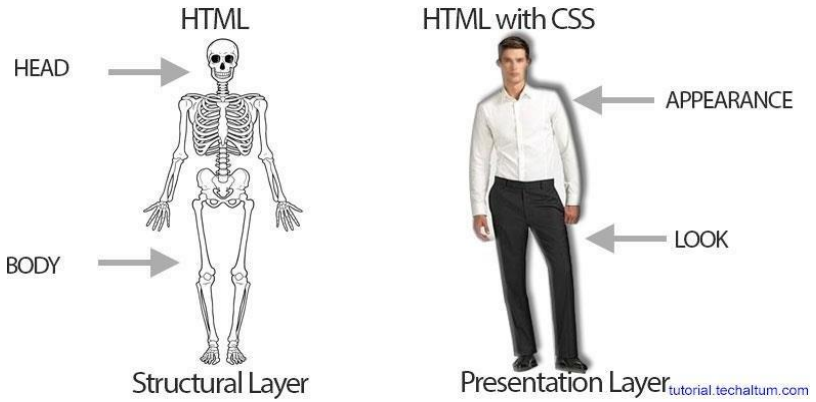
What is CSS?

- CSS stands for Cascading Style Sheets.
- CSS was developed by the W3C.
- CSS is a stylesheet language used to describe the presentation of a document written in a markup language.
- Its most common application is to style web pages written in HTML, XHTML and any kind of XML document.
- Styles define how to display HTML elements (font face, size, color, alignment, …etc)
- Styles are normally stored in Style Sheets
- The term cascading derives from the fact that multiple style sheets can be applied to the same Web page.

Why use CSS?

- The Separation of Structure and Presentation
- Managing Style at Large Sites
- Improved performance
- Decreased production work
- Rich design and layout





CSS Versions

- Cascading Style Sheets 1 (CSS1)
- Cascading Style Sheets 2 (CSS2 & CSS 2.1)
- Cascading Style Sheets 3(CSS3).

How to Link CSS?

- CSS can be linked to an HTML document as:
  - Embedding a style tag <style>
  - Linking to an external stylesheet file
  - Importing a stylesheet
  - Inline style

Inline style

- Inline style loses many of the advantages of style sheets by mixing content with presentation.
- Example:

```
<p style="color: red; font-family: ‘Ariel‘ ">
        This paragraph is styled in red with the Ariel font, if available.
</P>
```

Embedding a style tag

- An internal/embedded style sheet should be used when a single document has a unique style.
- You define internal styles in the head section by using the <style> tag
- An embedded (internal) style sheet should be used when a single document has a unique style.

```
<head>
    <style>
        H1 { color: blue }
        H2 { color: red}
    </style>
</head>
```

H1 header with blue color

H2 header with red color

Linking to an external style sheet file

- An external style sheet is ideal when the style is applied to many pages.
- With an external style sheet, you can change the look of an entire Web site by changing one file.
- Each page must link to the style sheet using the <link> tag.
- The <link> tag goes inside the head section:

```
<head>
        <link rel="stylesheet"  href="style.css"/>
</head>
```

Importing a style sheet

- Importing allows you to import one style sheet into another.
- This is slightly different than the link scenario, because you can import style sheets inside a linked style sheet.
- But if you include an @import in the head of your HTML document, it is written:

```
<STYLE>
                @import url("styles1.css);
                @import  url("style2.css");
                p {color: yellow }
</STYLE>
```

Cascading Order

- Styles will be applied to HTML in the following order:
  - Browser default
  - External style sheet
  - Internal style sheet
  - Inline style
- When styles conflict, the “nearest” (most recently applied) style wins

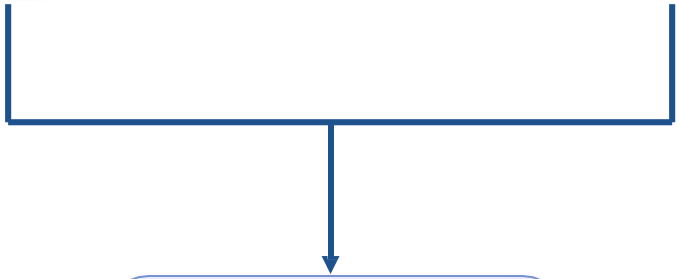
Cascading Order – Example

- **External Style sheet**

```
H3
{
        color: red;
        text-align: left;
        font-size: 8pt
 }
```

- **Internal Style sheet**

```
h3
{
        text-align: right;
        font-size: 20pt
}
```

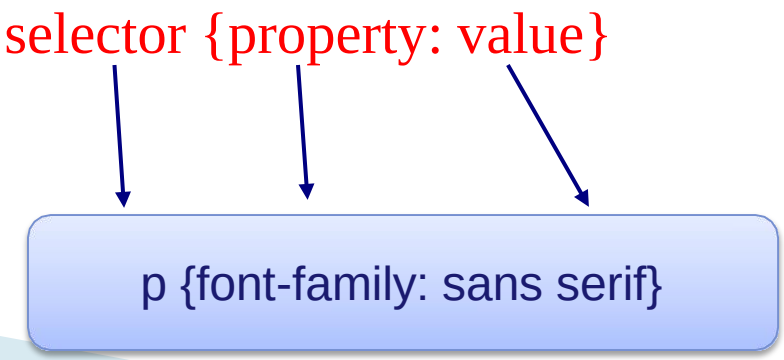


- **Resultant attributes**

```
color: red;
text-align: right;
font-size: 20pt
```

CSS Syntax

- The CSS syntax rule is made up of three parts:
  - selector
  - property
  - value
- selector is the tag to be affected
- property and value describe the appearance of that tag
- Style rules are formed as follows:



CSS Comments

```
<STYLE TYPE="text/css">
 p {
    color: red;
    /* This is a single-line comment */
    text-align: center;
}

/* This is
a multi-line
comment */
 </STYLE>
```

Selector

- **Several types of selectors are defined for use when implementing Style Sheets:**
  - Type Selector
  - Class Selector
  - ID Selector
  - Descendant/Contextual Selector
  - Child Selector
  - Adjacent sibling selectors
  - Attribute selectors

Universal Selector

- **The universal (*) selector selects all elements.**
- **Example:**

```
* {
    background-color: yellow;
}
```

- **The * selector can also select all elements inside another element**

```
div * {
    background-color: yellow;
}
```

Type Selector

- **The STYLE attribute can be added to any HTML element.**
- **Example:**

```
H1 {color: blue;}
```

- **It selects an element of the HTML document: P, H1, BODY, etc.**

Attribute Selector

- Allows you to specify rules that match attributes defined in the source document.
- Syntax :
  - Match when the element sets the "att" attribute, whatever the value of the attribute.

```
element[att] { property:value;}
```

  - Match when the element's "att" attribute value is exactly "val".

```
elemen [att = “val”] {property: value;}
```

Attribute Selector

- Example:
  - Selects “input” element that has the attribute type with value of “button”:

```
Input [type="button"]  {background-color: blue;}
```

  - Selects any element that has the attribute type with value of “button”:

```
[type="button"]  {background-color: blue;}
```

  - Selects all elements with a name attribute containing the word "flower“

```
[name~=flower]{background-color: blue;}
```

  - Selects every <a> element whose href attribute value begins with "https“

```
a[href^=http]{font-size: 12;}
```

  - Selects every <a> element whose href attribute value ends with ".pdf"

```
a[href$=.pdf]{font-size: 16;}
```

IDs

- **The ID attribute is used to define a unique style for an element.**
- **Example:**
  - In the CSS

```
#id1 {color: red}
```

  - In the HTML

```
<div id="id1" >
        This is the div with the id.
</div>
```

Classes

- Classes allow you to define a style which can be applied to multiple elements on your page.
- Example (1):
  - Say that you would like to have two types of paragraphs in your document: one right-aligned paragraph, and one center-aligned paragraph. Here is how you can do it with styles:
    - In the CSS

```
p.righttxt {text-align: right}
p.centertxt {text-align: center}
```

    - In the HTML

```
<p class="righttxt">
        This paragraph will be right-aligned.
</p>
<p class="centertxt">
        This paragraph will be center-aligned.
</p>
```

Classes (Cont.)

- Example (2):
  - To apply more than one class per given element:
    - → In the CSS

```
p.boldtxt { font-weight: bold}
p.largetxt { font-size: xx-large}
```

    - → In the HTML

```
<p class="boldtxt largetxt">
```

    - → This paragraph will be Bold & very large.</p>

- The paragraph above will be styled by the class "bold" AND the class "large".

Classes (Cont.)

- Example (3):
  - → In the CSS

```
p { font-size: 20} /* apply to all p*/
p.c1{color:red}
p.c2{color:blue}
p.c3{ font-weight: bold}
```

  - → In the HTML

```
<p>
    This paragraph will be font size 20.
</p>
<p class="c1">
    This paragraph will be font size 20, and color red.
</p>
<p class="c1 c3">
    This paragraph will be font size 20, and color red, and Bold
</p>
```

Classes (Cont.)

- Example (4):
  - To apply one class over more than one different HTML element:
    - → In the CSS

```
.bold { font-weight: bold }
```

    - → In the HTML

```
<p class="bold">
    This paragraph will be Bold.
</p>
<SPAN class="bold">
    This SPAN will be Bold too.
</SPAN>
```

- Both the paragraph & the span elements will be styled by the class "bold".

Descendant/Contextual Selector

- Used when we want selectors to match an element that is the descendant (inside) of another element in the document tree (In any level).

```
<H1>
        This headline is
                <span>very</span>
                important
</H1>
```

- Example:

```
H1 { color: red; }
span { color: green;}
H1 span{ color: blue;}
```

This headline is very important

Child Selector

- The child selector selects all elements that are the **immediate children** of a specified element.
- A child selector is made up of two or more selectors separated by ">“.
- Example:

```
divS>p {background-color: yellow;}
```

  - The following rule sets the style of all P elements that are children of div [that the div is their parent] (Applies only to direct children):

```
<div>
  <p>Paragraph 1 in the div.</p> <!– Direct child, applies-->
  <p>Paragraph 2 in the div.</p> <!– Direct child, applies-->
  <span><p>Paragraph 3 in the div.</p></span> <!-- not Child but Descendant -->
</div>
<p>Paragraph 4. Not in a div.</p>
<p>Paragraph 5. Not in a div.</p>
```

Adjacent Sibling Selector

- Adjacent sibling selectors have the following syntax: E1 + E2, where E2 is the subject of the selector.
- The selector matches if E1 and E2 share the same parent in the document tree and E1 immediately precedes E2.
- Example:
  - The following rule changes the color of an H2 that there's an H1 immediately precedes it:

```
H1+H2{color:red ;}
```

```
<body>
      <h1>text</h1>
      <h2> text</h2> will appear in red
</body>
```

Element1~element2 Selector

- The element1~element2 selector matches occurrences of element2 that are preceded by element1
- Both elements must have the same parent, but element2 does not have to be immediately preceded by element1.
- Example:
  - The following rule changes the color of all H2 that preceded by H1 with the same parent :

```
H1~H2 {color:red }
```

```
<body>
      <h1>text</h1>
      <p>paragraph</p>
      <h2> text</h2> will appear in red
</body>
```

Grouping selector

- Grouping selectors is done by separating each selector with a comma:

```
H1 { font-family: sans-serif }
H2 { font-family: sans-serif }
H3 { font-family: sans-serif }
```

  - is equivalent to:

```
H1,H2,H3 { font-family: sans-serif }
```

Pseudo Classes selector

- CSS pseudo-classes are used to add special effects to some selectors.
- A pseudo-class is similar to a class in HTML, but it's not specified explicitly in the markup.
- Syntax:

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

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

- Example:

Anchor Pseudo-classes:

```
a:link {color:#FF0000;}      /* unvisited link */
a:visited {color:#00FF00;}  /* visited link */
a:hover {color:#FF00FF;} /* mouse over link */
a:active {color:#0000FF;}  /* selected link */
a.menu:active {color:#0000FF;}  /* selected link */
```

CSS Pseudo Classes (cont.)

- More Example:

Selector	example	Description
:first-child	p:first-child	Selects every <p> element that is the first child of its parent
:last-child	p:last-child	Selects every <p> element that is the last child of its parent
:nth-child(n)	p:nth-child(2)	Selects every <p> element that is the second child of its parent
:only-child	p:only-child	Selects every <p> element that is the only child of its parent
:not()	.class1:not(p)	Selects every element that is not a <p> element
:empty	p:empty	Selects every <p> element that has no children (including text nodes)
:focus	input: focus	Selects the input element which has focus.

Pseudo Elements selector

- Pseudo-elements match virtual elements that don’t exist explicitly in the document tree.

- In CSS1 and CSS2, pseudo-elements start with a colon (:) In CSS3, pseudo-elements start with a double colon (::), which differentiates them from pseudo-classes.

- A CSS pseudo-element is used to style specified parts of an element.

Pseudo elements selector (cont.)

- Examples:

Selector	Example	Example description
::after	p::after	Insert content after every <p> element **Example**: p::after {content: " - Remember this";} http://www.w3schools.com/cssref/tryit.asp?filename=trycss_sel_after_style
::before	p::before	Insert content before every <p> element
::first-letter	p::first-letter	Selects the first letter of every <p> element
::first-line	p::first-line	Selects the first line of every <p> element
::selection	p::selection	Selects the portion of an element that is selected by a user http://www.w3schools.com/cssref/tryit.asp?filename=trycss3_selection

CSS measurement Units

- **Physical Measurements**
  - inches (in)
  - points (pt)
- **Screen Measurements**
  - pixels (px)
- **Relative Measurements**
  - %
  - em
- **1em = 12pt = 16px = 100%.**

CSS Properties

Font Styles

CSS and DOM Reference	Values
font-family:*name*	Font *name* can be any system font; multiple names can be specified in order of preference, separated by commas.
font-size:*size*	Font *size* is specified as in a unit of measurement, normally point size (12pt).
font-style:*style*	Font *style* specified as normal italic
font-weight:*weight*	Font *weight* specified as normal bold
font-variant:*variant*	Font *variant* specified as normal small-caps

Text Styles

CSS and DOM Reference	Values
text-align:*alignment*	Sets the horizontal *alignment* of text within an element. The *alignment* can be: left center right justify
line-height:*height*	Sets the *height* of lines of text in an element; specify a measurement (px, pt, *n*%, em, en) normal
letter-spacing:*spacing*	Sets the *spacing* between letters in an element; specify a measurement (px, pt, *n*%, em, en) normal
text-indent:*size*	Sets the *size* of indentation of the first line of a block of text; specify units of measurement (px, pt, *n*%, em, en)
text-transform:*case*	Sets the *case* of words in a text block using capitalize lowercase uppercase none
text-decoration:*style*	Sets a *style* using: underline overline line-through none

Text and Background Colors

CSS and DOM Reference	Values
color:*color*	Foreground color specified as a color name, hexadecimal value, or RGB value: color:red color:#FF0000 color:rgb(255,0,0)
background-color:*color*	Background color specified as a color name, hexadecimal value, or RGB value: background-color:red background-color:#FF0000 background-color:rgb(255,0,0)

Borders, Padding, and Margins (Cont.)

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 HTML elements, and it consists of: margins, borders, padding, and the actual content.
- The image below illustrates the box model:
  - **Content** - The content of the box, where text and images appear
  - **Padding** - Clears an area around the content. The padding is transparent
  - **Border** - A border that goes around the padding and content
  - **Margin** - Clears an area outside the border. The margin is transparent

Borders, Padding, and Margins (cont.)

CSS and DOM Reference.	Values
border-style:*style* **border-top-style:*style*** **border-right-style:*style*** **border-bottom-style:*style*** **border-left-style:*style***	Sets the *style* of a border surrounding a page element. The *style* can be applied to all borders (border-style, borderStyle) or to selected borders. Style types can be dashed dotted double groove inset none outset ridge solid

Borders, Padding, and Margins (Cont.)

CSS and DOM Reference.	Values
border-width:*width* **border-top-width:*width*** **border-right-width:*width*** **border-bottom-width:*width*** **border-left-width:*width***	Sets the *width* of a border surrounding a page element. The *width* can be applied to all borders (border-width, borderWidth) or to selected borders. Widths can be thin medium thick *n*px

Borders, Padding, and Margins (Cont.)

CSS and DOM Reference	Values
border-color:*color* **border-top-color:*color*** **border-right-color:*color*** **border-bottom-color:*color*** **border-left-color:*color***	Sets the *color* of a border surrounding a page element. The *color* can be applied to all borders (border-color, borderColor) or to selected borders. The *color* is specified as a color name, hexadecimal value, or RGB value.

Borders, Padding, and Margins (Cont.)

CSS and DOM Reference	Values
border:*style width color*	Border styles, widths, and colors can be set with the single border specification by coding these values separated by a blank space: border:solid 1px red

Background Images

CSS and DOM Reference	Values
background-image:url(*url*)	Sets the URL of a background image; *url* can be set to none to prevent an image from loading.
background-position:*location*	Sets the *location* of the left and top edges of the background image with a pair of values separated by a space. Values are left|center|right paired with top|center|bottom OR *x% y%* Locations can also be specified as pairs of percentages or pixels for the left and top values. https://www.w3schools.com/cssref/pr_background-position.asp
background-repeat:*axes*	Sets whether a background image should repeat along the horizontal and/or vertical axes. *Axes* values are: no-repeat repeat repeat-x repeat-y
background-attachment:*value*	Describes whether a background image remain fixed in place or scrolls with the document. *Values* are: fixed scroll

Background Images (Cont.)

CSS and DOM Reference	Values
background-size	The background-size property specifies the size of the background images. There are four different syntaxes you can use with this property: the keyword syntax ("auto", "cover" and "contain"), the one-value syntax (sets the width of the image (height becomes "auto"), the two-value syntax (first value: width of the image, second value: height), and the multiple background syntax (separated with comma). https://www.w3schools.com/cssref/css3_pr_background-size.asp

- Background image properties reference: https://www.w3schools.com/cssref/css3_pr_background.asp

Positioning styles

- **The CSS positioning properties allow you to position an element.**

- **CSS syntax:**

```
position: static|absolute|fixed|relative|initial|inherit;
```

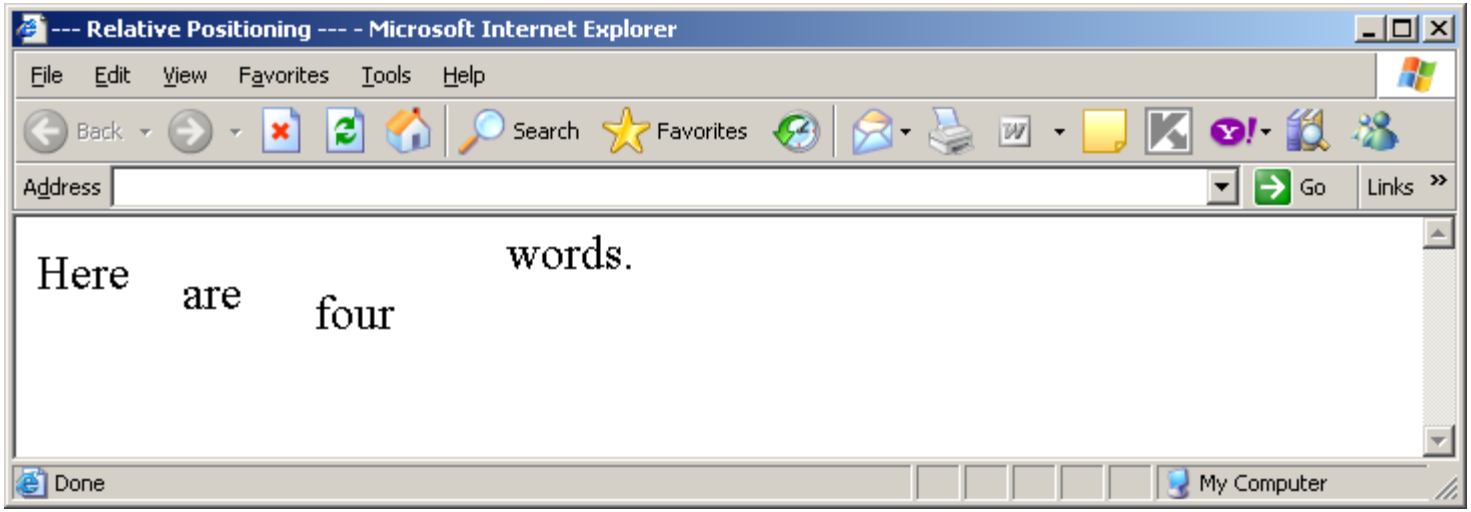
Positioning styles (cont.)

- **Elements can be positioned as:**
  - **Static Positioning**
    - HTML elements are positioned static by default.
    - A static positioned element is always positioned according to the normal flow of the page.
    - Static positioned elements are not affected by the top, bottom, left, and right properties.
  - **Fixed Positioning**
    - An element with fixed position is positioned relative to the browser window.
    - It will not move even if the window is scrolled.
  - **Sticky Positioning**
    - A sticky element toggles between relative and fixed, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like position:fixed).
    - https://www.w3schools.com/howto/howto_css_sticky_element.asp

Positioning styles (cont.)

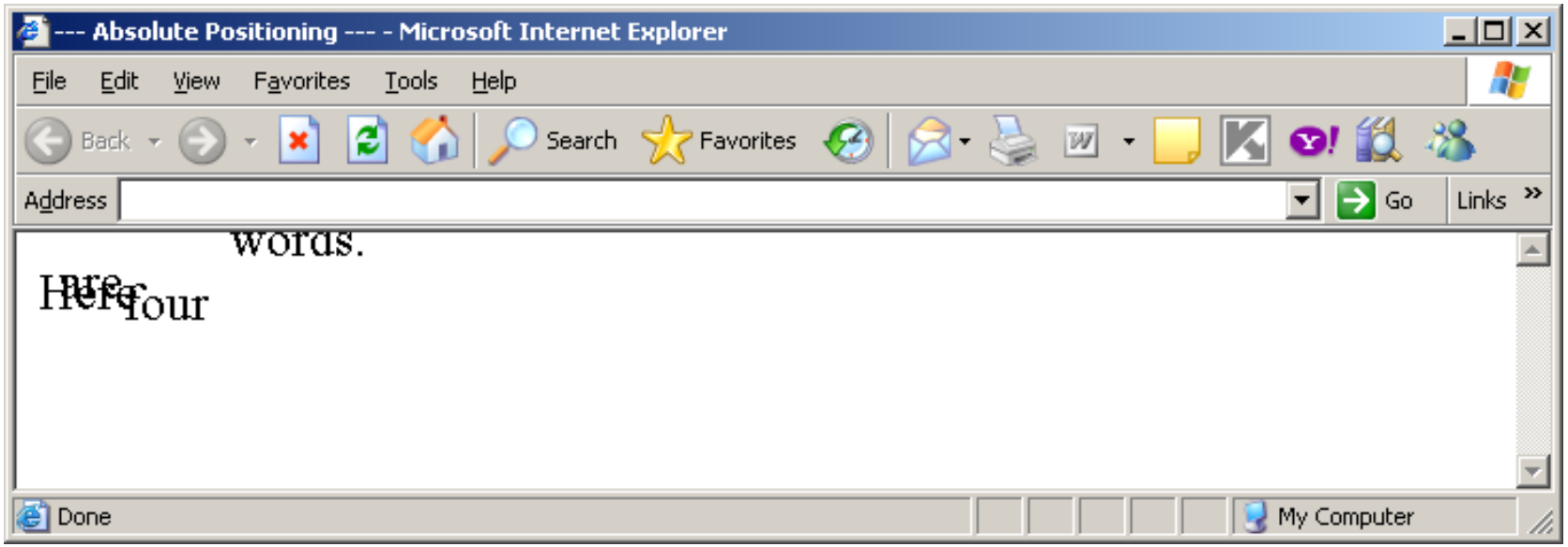
- **Elements can be positioned as:**
  - **Relative Positioning**
    - A relative positioned element is positioned relative to its normal position.
  - **Absolute Positioning**
    - An absolute position element is positioned relative to the first parent element that has a position other than static.
    - Absolutely positioned elements are removed from the normal flow. The document and other elements behave like the absolutely positioned element does not exist.
    - Absolutely positioned elements can overlap other elements.

Position:relative




```
<p style="font-size:18pt">
        <span>Here</span>
        <span style="position:relative; left:20px; top:10px">are</span>
        <span style="position:relative; left:50px; top:20px">four</span>
        <span style="position:relative; left:100px; top:-10px">words.</span>
</p>
```

Position:absolute




```
<p style="font-size:18pt">
      <span>Here</span>
      <span style="position: absolute; left:20px; top:10px">are</span>
      <span style="position: absolute; left:50px; top:20px">four</span>
      <span style="position: absolute; left:100px; top:-10px">words.</span>
</p>
```

Top, left

CSS and DOM Reference	Effects
left:*n px*	Sets the left edge of the element relative to its container element; *n* is a ***string*** measurement unit, e.g., left:100px.
top:*n px*	Sets the top edge of the element relative to its container element; *n* is a ***string*** measurement unit, e.g., top:100px
right:*n*	Sets the right edge of the element relative to its container element; *n* is a ***string*** measurement unit, e.g., right:100px.
bottom:*n*	Sets the bottom edge of the element relative to its container element; *n* is a ***string*** measurement unit, e.g., bottom:100px

Sizing

CSS and DOM Reference	Effects
width:*value*	Sets the width of the element; *n* is a **string** measurement, either in pixels or percentages.
height:*n*	Sets the height of the element; *n* is a **string** measurement, either in pixels or percentages.

Display

- The display property determines how the element id displayed.
- CSS syntax

```
display:block|inline|none ...
```

Display (cont.)

❑ Display values:

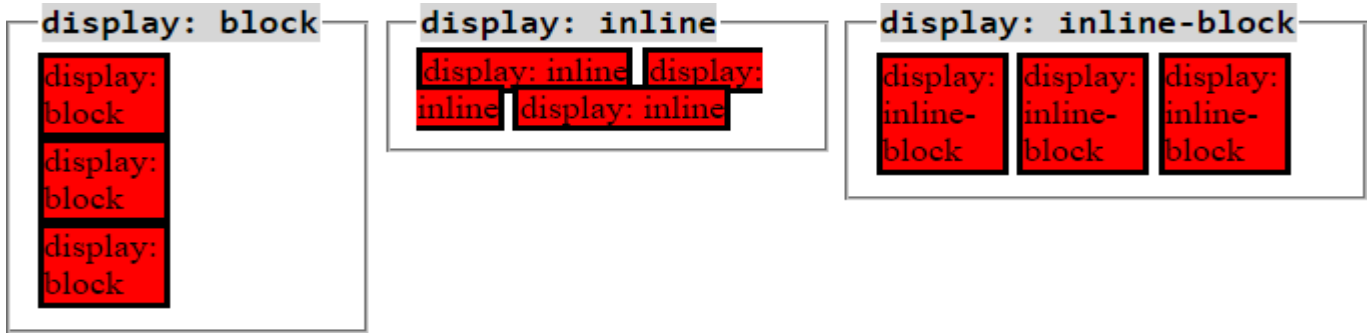
Value	Description
inline	Default value. Displays an element as an inline element (like)
block	Displays an element as a block element (like <p>)
none	The element will not be displayed at all (has no effect on layout)
list-item	Let the element behave like a element
table	Let the element behave like a <table> element
table-cell	Let the element behave like a <td> element
table-column	Let the element behave like a <col> element
table-row	Let the element behave like a <tr> element

- **More properties:**
  - http://www.w3schools.com/cssref/pr_class_display.asp
  - http://www.w3schools.com/jsref/prop_style_display.asp

Display (cont.)

The display: inline-block Value

- Compared to display: inline, the major difference is that display: inline-block allows to set a width and height on the element.
- Also, with display: inline-block, the top and bottom margins/paddings are respected, but with display: inline they are not.
- Compared to display: block, the major difference is that display: inline-block does not add a line-break after the element, so the element can sit next to other elements.



- **More:**

https://www.w3schools.com/css/css_inline-block.asp

https://stackoverflow.com/questions/9189810/css-display-inline-vs-inline-block

http://dustwell.com/div-span-inline-block.html

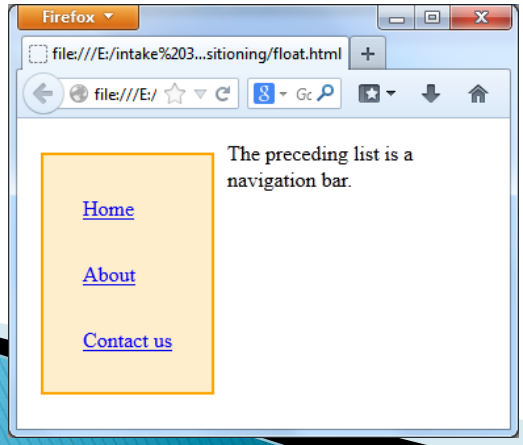
https://alligator.io/css/display-inline-vs-inline-block

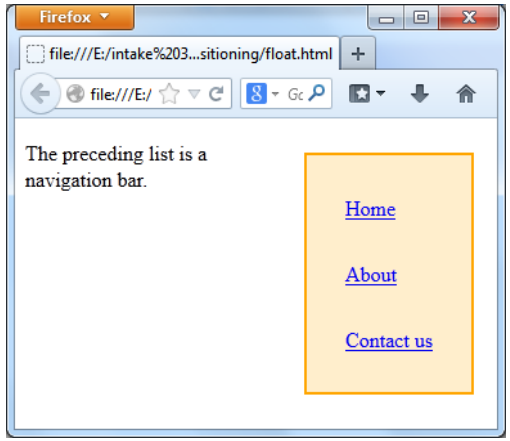
http://learnlayout.com/inline-block.html

Float

- With CSS float, an element can be pushed to the left or right, allowing other elements to wrap around it.
- CSS

```
float:left|right|none|inherit|initial
```





Float (cont.)

- **Turning off Float - Using Clear**
  - Elements after the floating element will flow around it. To avoid this, use the clear property.
  - The clear property specifies which sides of an element other floating elements are not allowed.

- CSS syntax

```
Clear: left|right|both|none|inherit|initial
```

Overflow

- specifies what happens if content overflows an element's box

- CSS syntax

```
overflow: visible|hidden|scroll|auto|initial|inherit;
```

Overflow (cont.)

- Property values:

Value	Description
visible	The overflow is not clipped. It renders outside the element's box. This is default
hidden	The overflow is clipped, and the rest of the content will be invisible
scroll	The overflow is clipped, but a scroll-bar is added to see the rest of the content
auto	If overflow is clipped, a scroll-bar should be added to see the rest of the content
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element.

z-index

- The z-index property is used to place an element "behind" another element.
- Default z-index is 0.
- The higher number the higher priority. z-index: -1 has lower priority.
- CSS syntax

```
z-index:n
```

Visibility

- The visibility property determines if an element is visible or not.
- CSS syntax

```
visibility:hidden|visible|collapse
```

- Collapse: Only for table elements. collapse removes a row or column, but it does not affect the table layout. The space taken up by the row or column will be available for other content.

- If collapse is used on other elements, it renders as "hidden“

http://www.w3schools.com/cssref/playit.asp?filename=playcss_visibility_collapse

Opacity

- **Example: Image Opacity**

- CSS Format:

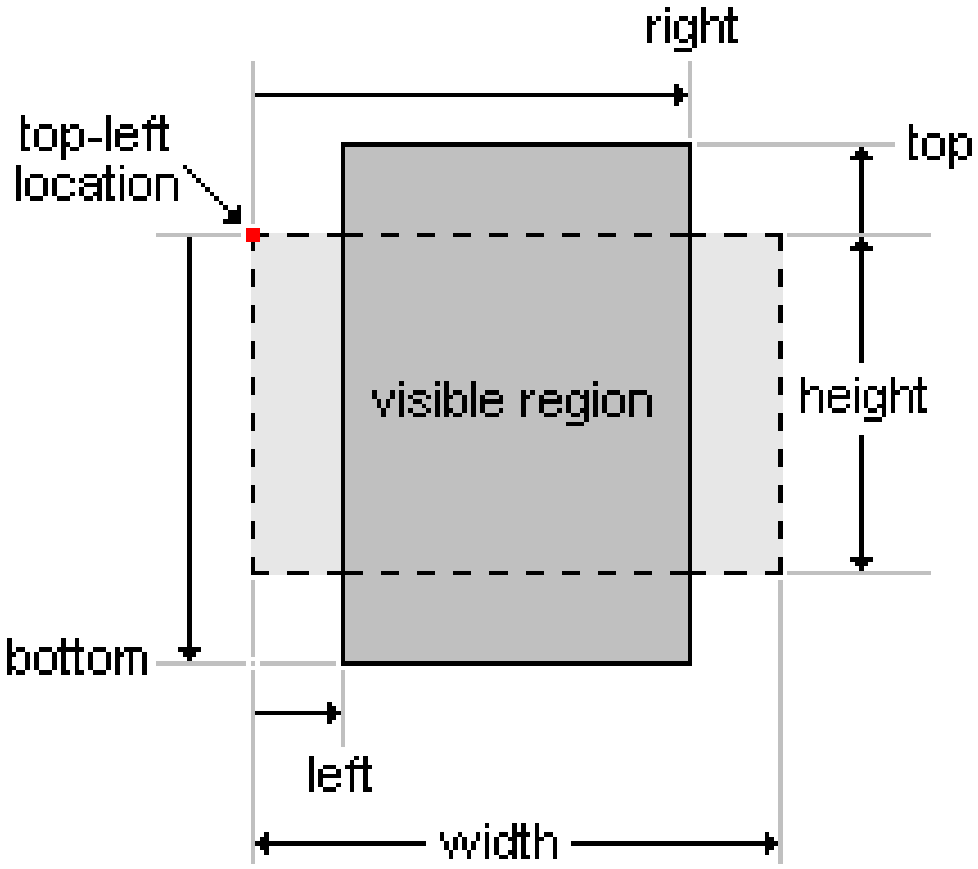
```
opacity: 0.4; /* from 0 to 1*/
```

Clip

- Specifies how an element is clipped for display; i.e. which part is visible.
- The clipping region is defined as a rectangle by setting the clip value for each of the 4 edges (top, right, bottom, left).
- For each edge you can clip a portion of the viewing space away, or to add extra viewing space.
- CSS syntax

```
clip:rect(top,right,bottom,left)
```

Clip(Cont.)



clip:rect(top,right,bottom,left)

Any Questions ?

Thank you