BTI220 - Internet Architecture and Development

Week 7: CSS Properties, Page Layouts and Navigation

Agenda

- CSS Styling
  - lists, tables and hyperlinks
- CSS Centering
- CSS Positioning
- Page Layouts and Navigation
- Introduction to DOM

CSS Lists

- The list-style-type CSS property specifies appearance of a list item element.
- Examples:

```
{ list-style-type:circle; }
{ list-style-type:square; }
{ list-style-type:upper-roman; }
{ list-style-type:lower-alpha; }
```

- Default value: disc

- css_list.html

Property list-style-type Values

Value	Description	e.g.
none	No item marker is shown	
disc	A filled circle (default value)	
circle	A hollow circle	
square	A filled square	
decimal	Han decimal numbers	1, 2,
decimal-leading-zero	Decimal numbers	01, 02, 03, … 98
lower-roman	Lowercase roman numerals	i, ii, iii, iv, v…
upper-roman	Uppercase roman numerals	II, III, IV, V…
lower-greek	Lowercase classical Greek	α, β, γ…
lower-alpha, lower-latin	Lowercase ASCII letters	a, b, c, … z
upper-alpha, upper-latin	Uppercase ASCII letters	A, B, C, … Z

***More list*-style-type values| MDN**

Table Formatting

- CSS properties for table formatting:
  - margin, padding, width, height, text-align, vertical-align, background-color, background-image, border
- Border Collapse
  - Property: border-collapse sets whether the table borders are collapsed into a single border or separated.
  - e.g.:

```
Table { border-collapse:collapse; }

table,th, td { border: 1px solid black; }
```

Table Formatting

- Formatting The Entire Table:

  table { margin: auto; width: 80%; }

  table { border: dotted; }

  table { background-color: yellow; }

- Formatting Table Cells

  td {border: 4px inset #4400FF}

  td {padding:10px 20px}

  td {background-color: aqua}

  td {height: 100px; width: 400px}

  using "text-align" and "vertical-align"

- css_table.html

Border Collapse

- Property: border-collapse sets whether the table borders are collapsed into a single border or separated.
- E.g.:

```
table
{
    border-collapse:collapse;
}
table, th, td
{
    border: 1px solid black;
}
```

Table Sections

- <thead> - group the first one or more rows of a table for formatting
- <tbody> - group the middle rows of a table for formatting
- <tfoot> - group the last one or more rows of a table for formatting

- css_table_section.html

Styling Links

- Properties:
  - color, font-family, background
- Links are styled differently depending on what state they are in.
- The four link states are:
  - a:link - a normal, unvisited link
  - a:visited - a link the user has visited
  - a:hover - a link when the cursor hovers over it
  - a:active - a link the moment it is clicked
- Note: When setting the style for several link states,
  - ✓ a:hover MUST come after a:link and a:visited
  - ✓ a:active MUST come after a:hover

❑ css_link.html css_link-as-button.html

CSS – display Property

- The display CSS property specifies the type of rendering box used for an element.
  - Defaule value: inline
  - The value none lets you turn off the display of an element.
- e.g.

```
p.inline  { display: inline; }
```

- css_display.html

The display Property Values

Value	Description
inline	Default value. Displays an element as an inline element (like)
block	Displays an element as a block element (like <p>)
inline-block	Displays an element as an inline-level block container. The inside of this block is formatted as block-level box, and the element itself is formatted as an inline-level box
inline-table	The element is displayed as an inline-level table
list-item	Let the element behave like a element
table	Let the element behave like a <table> element
table-caption	Let the element behave like a <caption> element
table-column-group	Let the element behave like a <colgroup> element
table-header-group	Let the element behave like a <thead> element
table-footer-group	Let the element behave like a <tfoot> element
table-row-group	Let the element behave like a <tbody> 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
none	The element will not be displayed at all (has no effect on layout)

Centering - Lines Of Text

- Centering lines of text in a paragraph or in a heading.
- p { text-align: center }
- h2 { text-align: center }

- center_text.html

Centering – a Block

- Rephrase: left and right margin to be equal.
  - set the margins to 'auto'.
  - used with a block of fixed width.

```
div.center { border: 2px solid red;
             margin-left: auto;
             margin-right: auto;
             width: 400px; }

<div class ="center"> … </div>
```

- center_block.html

Centering – Vertically

- Specify the outer block as a table cell, the contents of a table cell can be centered vertically.

```
div { height: 100px; width: 500px; }
div.center { border: 10px dotted red;
                    display: table-cell;
                    vertical-align: middle;
                    text-align: center;}

<div class="center"> This div is centered </div>
```

- center_vertical.html

Positioning

- CSS position property
  - can be used to position elements precisely in HTML pages.
- a browser renders html statements in the order that they are in the html file - this is called normal flow

Positioning

- Values for property: position
  - absolute - position precisely within the containing element
  - relative - position precisely relative to normal flow
  - fixed - position precisely within the browser window, and does not move when the page is scrolled
  - static - position using normal flow (default)
- Values for Properties: "left", "right", "top", and "bottom" can be given offsets in px or %
- tags, with various classes, are used to create the different divisions of the document.

- position.html
- Position_relative.html

Positioning

- What if the text takes more than the allotted space?
- use the property "overflow" to specify an action:
  - { overflow:scroll; } - include scroll bars
  - { overflow:auto; } - scroll if required
  - { overflow:hidden; } - hide overflow
  - { overflow:visible; } – default

- position.html

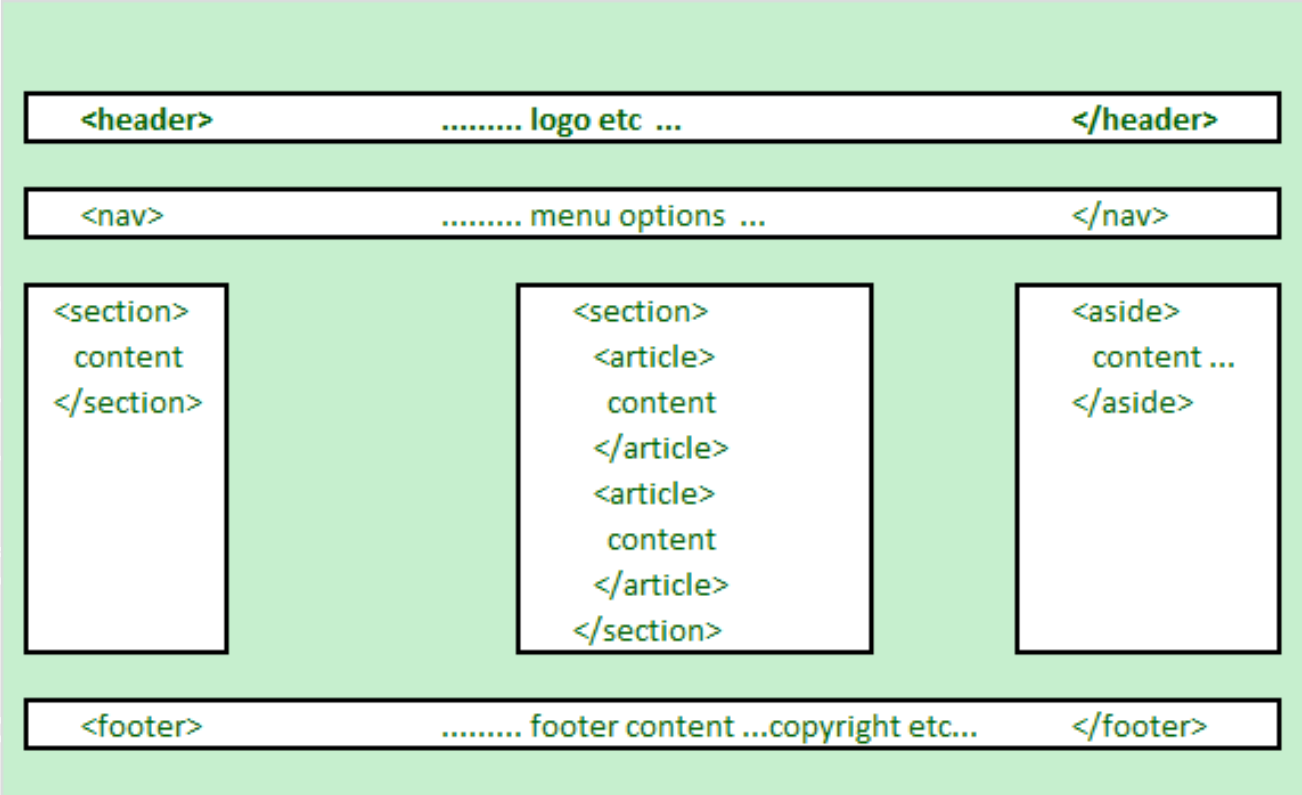
Positioning

- graphics and titles can be positioned in a similar fashion:

- position_graphic.html

HTML5 Structural Elements

- HTML 5 defines a number of new container elements for constructing documents.
  - header, nav, section, aside, article and footer



HTML5 Structural Elements

Elements	Description
<header>	the web page/site header content
<nav>	the navigation functionality for the page/site.
<section>	the grouping of related subjects on the web page.
<main>	the main content on the web page.
<article>	contains a standalone content on the web page.
<aside>	used for content that's not central to the web page.
<footer>	the web page/site footer content

Notes: The element is the generic container for flow content, which does not inherently represent anything. It should be used only when no other semantic element (such as above elements) is appropriate.

HTML5 Structural Elements

```
<body>
  <header>..... logo etc  …
  </header>

  <nav>  ..... menu options  …
  </nav>

  <section id="sidebar1">..... section 1  …
  </section>

  <section id="main"><!-- may be replaced by main element -->
    <article>article within the section </article>
    <article>another article within the section </article>
  </section>

  <aside>..... aside content …
  </aside>

  <footer> ..... footer content ...copyright etc…
  </footer>
 </body>
```

- html5_structure.html

<article> Tags Can Contain Others

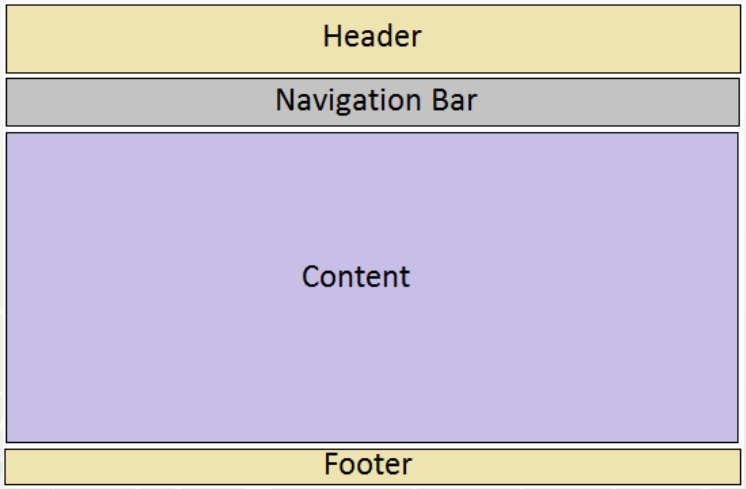
```
<article>
   <header>
   </header>

   <section id="introduction">
   </section>

   <section id="content">
   </section>

   <section id="summary">
   </section>

   <footer>
   </footer>
 </article>
```

The HTML4 Structural Elements:

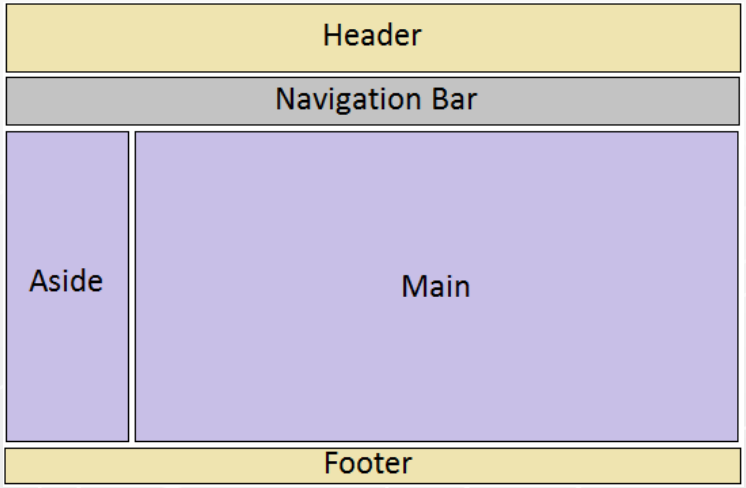
```
<body>
  <div class="header">..... logo etc  …
  </div>

  <div class="navigation">  ..... menu options  …
  </div>

  <div class ="sidebar1">..... Column 1  …
  </div>

  <div class="main">
    ……..main column content goes in here……
  </div >

  <div class="aside">..... aside content …
  </div>

  <div class="footer"> ..... footer content ...copyright etc…
  </div>
 </body>
```

Web Pages Layouts

- One-column layout



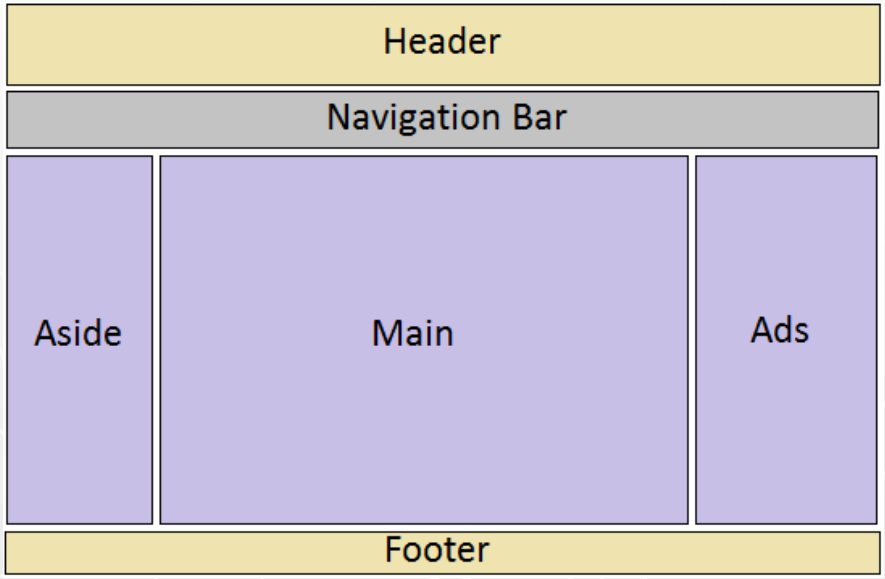
- Two-column layout



- layout-1-column.html
- layout-2-column.html

Web Pages Layouts

- Three-column layout



- layout-3-column.html

Create Layouts with HTML5 and CSS3

- Creating 2-column fluid (float-based) layouts with CSS
  - HTML5 document without CSS:
    - html5_structure.html

1. Set the width of the page (e.g. 960px) and center the page:

```
.container { width: 960px; margin: auto; }
```

2. Set the width of the "aside" block and float it to left:

```
aside { width: 192px; float: left; }
```

3. Set the width of the "main" section and float it to left:

```
section.main { width: 768px; float: left; }
```

Create Layouts with HTML5 and CSS3

3. Set the clear property of the footer to 'both':

```
footer { clear: both; background-color: #aaa; }
```

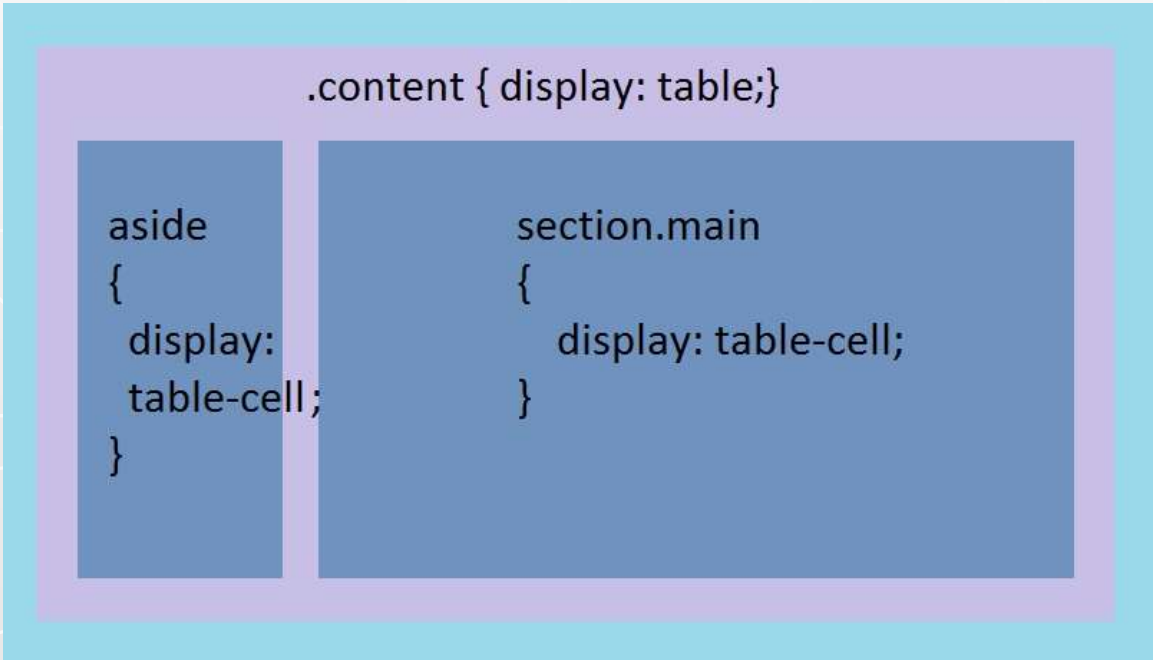
4. Set margin, border, padding, background-color, … to each structural element, e.g.:

```
aside, section.main {margin-top: 58px; margin-right: 10px;
margin-left: 10px; }
```

Note: You may use relative width for the page and columns.

Create Layouts with HTML5 and CSS3

- Creating 2-column tabular layouts with CSS
  - HTML5 document without CSS:
    - html5_structure.html
  - Set CSS:



Create Layouts with HTML5 and CSS3

- Creating 2-column tabular layouts with CSS (cont')
  - CSS code for 2-column tabular layouts :

```
.container { display: table; width: 960px; margin: auto; }
aside { display: table-cell; width: 192px; }
section.main { display: table-cell; width: 720px; }
footer { background-color: #aaa; }
aside, section.main {margin-top: 58px; margin-right: 10px; margin-left: 10px;
```

- ✓ Note: using HTML table to create page layouts is obsolete and not allowed in INT222 assignments

Navigation and Menus

- Web page navigation

```
<nav>
  <ul>
     <li><a href="#top">Home</a></li>
     <li><a href="#timetable">Timetable</a></li>
     <li><a href="#standards">Standards</a></li>
     <li><a href="ibc233/ibc233.html">IBC233</a></li>
     <li><a href="int222/int222.html">INT222</a></li>
     <li><a href="bti220/bti220.html">BTI220</a></li>
   </ul>
</nav>
```

- convert the unordered the list a navigation bar or menu.

Navigation and Menus

- Single Level Menu Options
  - Horizontal Single Level Menu Example
    - Navigation bar
  - Vertical Single Level Menu Example
- Multi Level Menu Options
  - Horizontal Multi Level Menu Example
  - Vertical Multi Level Menu Example

Introduction to DOM

Document Object Model (DOM)

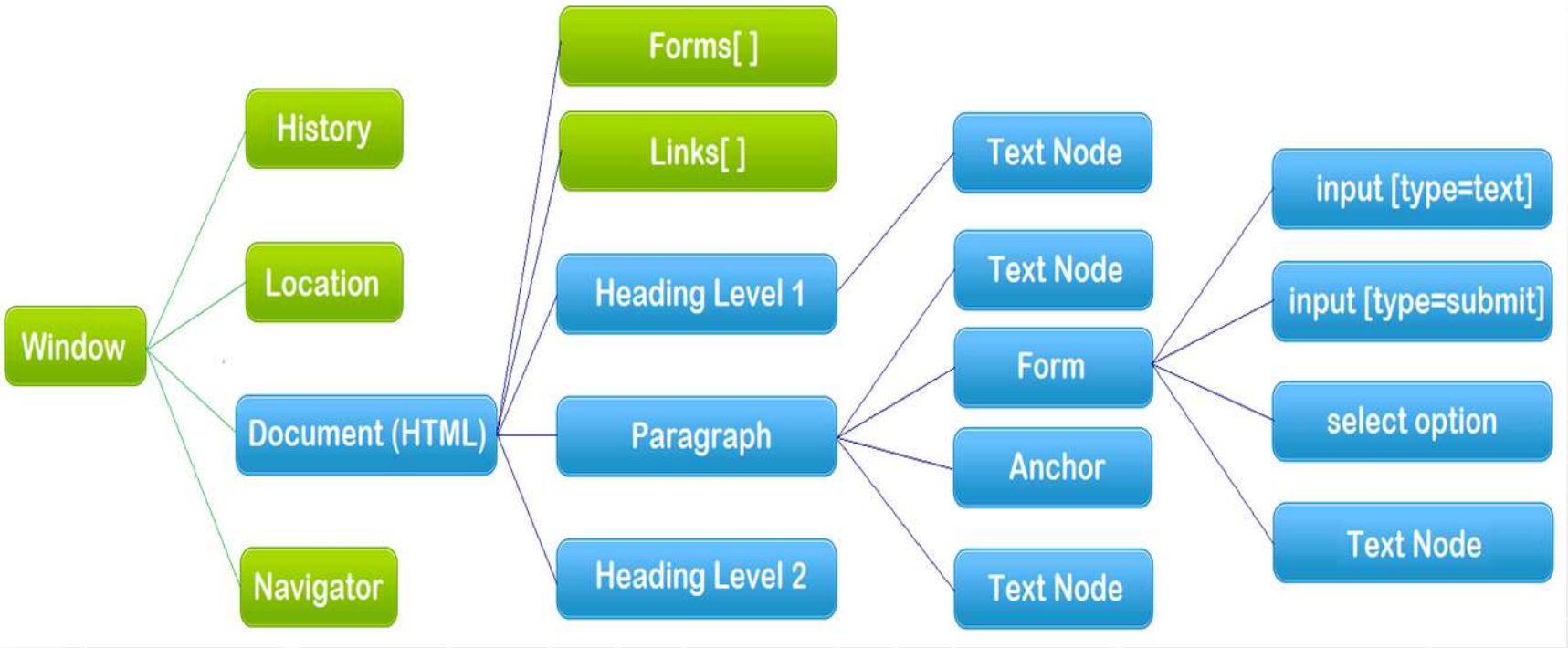
- The Document Object Model (DOM) is an 2-way application programming interface (API) for HTML (and XML) documents.
  - It defines the way how the document structure, style and content can be accessed and manipulated.
  - It provides a structured representation of the document.
  - It's cross-platform and language-independent.

- With the DOM, programmers can build documents, navigate their structure, and add, modify, or delete elements and content using JavaScript or other languages.

Document Object Model (DOM)

- The document's structure is defined by the W3C Document Object Model (DOM) specification.
- A web browser provides and implements the DOM.
- The DOM provides a representation of the document as a structured group of nodes and objects that have properties and methods.
- It connects web pages to scripts or programming languages.

The Hierarchy of DOM



- Everything in the Hierarchy of DOM is an object.
- The Document Object Model: the blue color part

The Hierarchy of DOM

- On the top the Hierarchy is the window object.
  - When JavaScript is hosted in a browser, the browser provides the window host object to JavaScript.
  - The window object possesses the properties of document, history, location and navigation objects.
  - The global functions are belong to the window object.

- The DOM, strictly speaking, covers the document object and the objects underneath.
  - When an HTML document is loaded into a web browser, it becomes a document object.

Document Object

- An HTML document loaded into a browser window becomes a Document object.
- The Document object provides access to all HTML elements in a page, from within a script.
- The Document object is also part of the Window object, and can be accessed through the window.document property.
- The document object is the root node of the HTML document and the "owner" of all other nodes:
  - element nodes, text nodes, attribute nodes, and comment nodes.

Document Object

- Document object properties and methods (legacy DOM):

document.anchors	Returns a collection of all the anchors in the document
document.body	Returns the body element of the document
document.domain	Returns the domain name of the server that loaded the document
document.forms	Returns a collection of all the forms in the document
document.images	Returns a collection of all the images in the document
document.links	Returns a collection of all the links in the document
document.referrer	Returns the URL of the document that loaded the current document
document.title	Sets or returns the title of the document
document.URL	Returns the full URL of the document
document.write()	Writes HTML expressions or JavaScript code to a document
document.writeln()	write() with a newline character after each statement

Document Object

- The document object provides properties and methods to access all node/element objects, from within JavaScript.
- Examples
  - A form input element could be accessed as either

    document.formName.inputName

    or

    document.forms[0].elements[0]

    // (if the input element is first element in the first form)
  - An image in a web page can be accessed via

    document.images[0] // (the 1st image in the page)
- document.html

Document Object

- Document object methods – get/query element(s):
  - document.getElementById()
  - document.getElementsByName()
  - document.getElementsByClassName()
  - document.getElementsByTagName()
  - document.querySelector()
    - Returns the first element that matches a specified CSS selector(s) in the document
  - document.querySelectorAll()
    - Returns a static NodeList containing all elements that matches a specified CSS selector(s) in the document
- Examples:
  - var elem = document.getElementById("demo");
  - var paras = document.getElementsByTagName("p");
  - var example = document.querySelector("p.example");

Document Object

- Document object methods – create element(s), …
  - document.createElement()
  - document.createTextNode()
  - document.createAttribute()
  - document.createComment()

  - document.addEventListener()

- create-element.html

update elements

- Example 1:

```
document.getElementById("demo").innerHTML = "Hello World";
```

- Example 2:

```
var i, elems = document.getElementsByName("pets");
for (i = 0; i < elems.length; i += 1) {
   // elems[i].type = "checkbox";
   elems[i].setAttribute('type', 'checkbox');
}
```

- Example 3:

```
document.querySelector(".example").style.backgroundColor = "red";
```

- update-elements.html

Element innerHTML property

- An element's innerHTML property sets or gets the HTML text content of the element.
- Example 1:

```
HTML:   <p id="demo">Welcome to Seneca College!</p>
JS:     var elem = document.getElementById("demo");
        alert("The text in the paragraph: " + elem.innerHTML);
        elem.innerHTML = prompt("Please input some text");
```

- Example 2:

```
HTML:   <div id=display></div>
JS:     var elem = document.getElementById("display");
        elem.innerHTML = "<p>Paragraph added/changed!</p>";

        elem.innerHTML += '<ul><li>Coffee</li><li>Tea</li></ul>';
        // Writing HTML within JavaScript is not a good practice. Use DOM instead if possible.
```

- innerHTML.html

Resourceful Links

- CSS Library
- Making a Sphere in CSS

- MDN - Document Object Model (DOM)
- MDN – Node (interface) reference
- MDN – Element (interface) reference
- MDN – Text (interface) reference

Thank You!