WEB222 - Web Programming Principles

Week 7: DOM and Events

Agenda

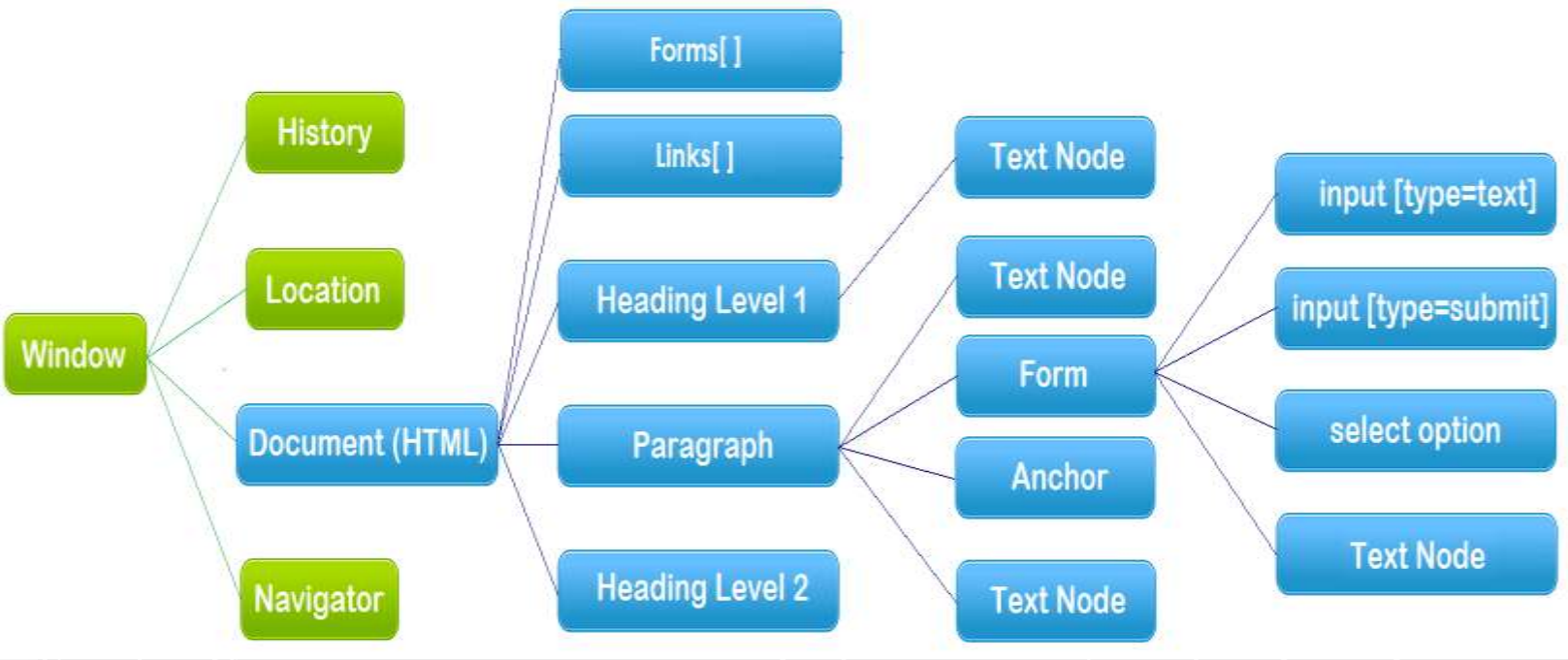
- Document Object Model (DOM)
- DOM Tree, Node
- DOM Events

The 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 updated.
  - It provides a structured representation of the document.
- With the DOM, programmers can build documents, navigate their structure, and add, modify, or delete elements and content using JavaScript or other languages.

The Document Object Model (DOM)

- The hierarchy of Document Object Model (the blue color part)



- Document – HTML Document/file
- API – each node above is an object that has methods and properties

The Document Object

- An HTML document loaded into a browser window becomes a Document object.
- The Document object provides access to all HTML elements/objects in a page, from / within JavaScript.
- The Document object is also part of the Window object, so it can be accessed through window.document or document.

The 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 object properties

- Examples:

  - Access:

    ```
    console.log(document.title);
    ```

  - Update:

    ```
    document.title = "Seneca College";
    ```

Document object methods

- get/query element(s):
  - 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
  - document.getElementById()
  - document.getElementsByClassName()
  - document.getElementsByTagName()

Examples

```
var elem = document.querySelector("#demo");  // ✓
var elem = document.getElementById("demo"); // equivalent

var example = document.querySelectorAll(".example"); // ✓
var example = document.getElementsByClassName("example"); //equivalent

var example = document.querySelector(".example"); //get the 1st one
```

Document object methods

- Create element and event handler:
  - document.createElement()
  - document.createTextNode()
  - document.createAttribute()
  - document.createComment()

  - document.addEventListener()

- create-element.html

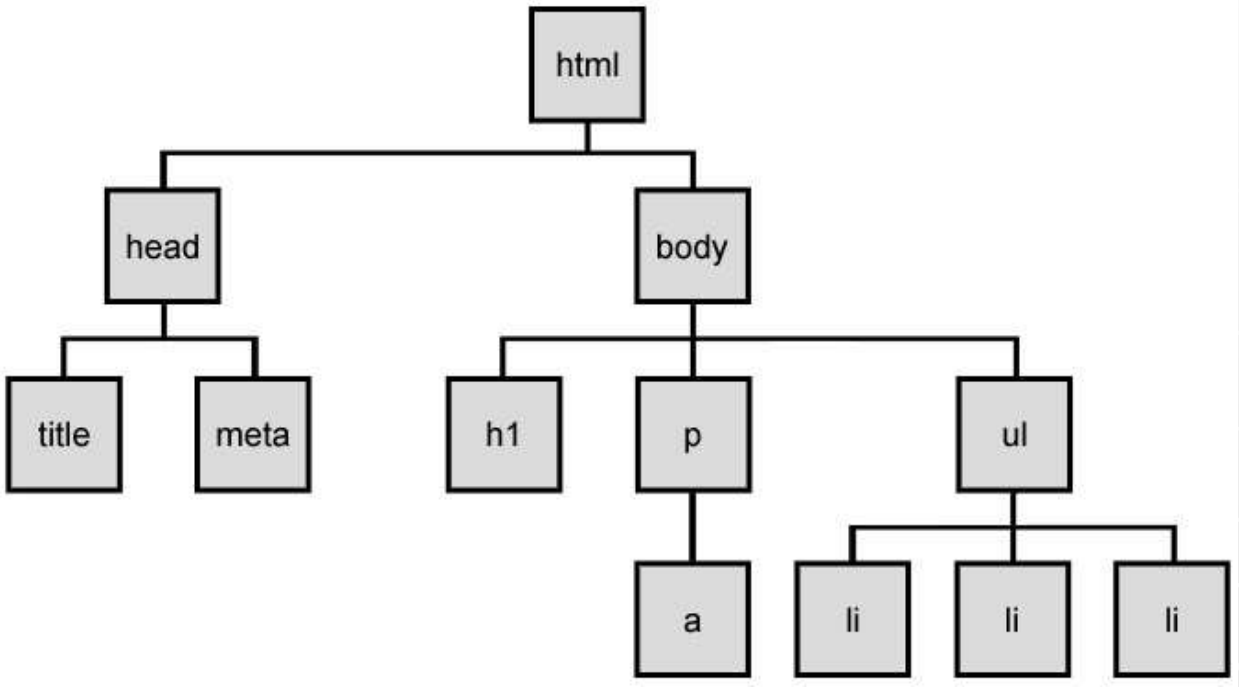
The DOM tree

- When a HTML document is loaded, the browser creates a **D**ocument **O**bject **M**odel.
- The DOM represents a document as a tree.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8" />
        <title>This is a Document!</title>
    </head>
    <body>
        <h3>Welcome!</h3>
        <p>This is a paragraph.</p>
        <p>This is a paragraph with a
            <a href="index.html">link</a> in it.
        </p>
        <ul>
            <li>first item</li>
            <li>second item</li>
            <li>third item</li>
        </ul>
    </body>
</html>
```

The DOM tree

- The HTML elements of the page are nested into a tree-like structure of objects.
- The tree is made up of parent-child relationships, a parent can have one or many children elements.



- dom-tree.html

HTML DOM Nodes

- An HTML document is a structure of nodes.
- Everything in an HTML DOM is a node.
  - window.document, each element,
  - attributes and text inside elements are nodes,
  - DOCTYPE, comments, whitespaces are also nodes.
- All these DOM objects inherit form node object, providing properties and methods – Web API Interfaces
  - DOM document object
  - DOM Element objects
  - DOM Attribute objects
  - DOM Event object

Types of DOM nodes

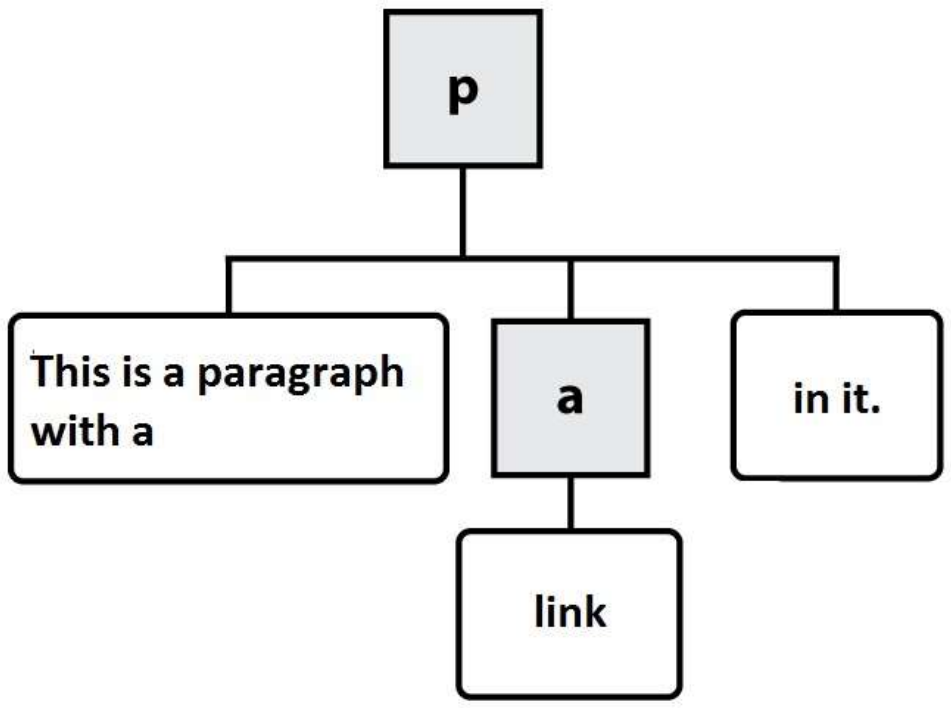
- HTML documents are made up of HTML elements
- In the HTML DOM, every HTML element is represented by an element object (node). In addition, there are other types of nodes.
- Types of DOM nodes
  - element nodes (HTML tag)
    - For each HTML element, there is a node object.
    - It can have children and/or attributes
  - text nodes (text in a block element)
  - attribute nodes (attribute/value pair)
    - text/attributes are children in an element node
    - cannot have children or attributes
    - not usually shown when drawing the DOM tree

At the Ends of DOM Tree

- HTML

```
<p>This is a paragraph with a <a href="index.html">link</a> in it.</p>
```

- DOM nodes
  - element nodes
  - text nodes
  - attribute nodes



DOM Nodes / Element Objects

- Properties and Methods of DOM Nodes / Element Objects – the standard API
- Properties:

  element.className
  element.id
  element.innerHTML

  element.style
  element.parentNode
  element.nextSibling
  … …

- Methods:

  element.querySelector()
  element.querySelectorAll()
  element.getElementsByTagName()
  element.getElementsByClassName()
  element.addEventListener()

  element.appendChild()
  element.insertBefore()
  element.removeChild()
  element.setAttribute()
  element.getAttribute()

- The HTML DOM Element Object

Examples

- Access all paragraphs within a div element

```
<p>Mail to: </p>
<div id="address">
    <p>70 The Pond Road</p>
    <p>Toronto, ON</p>
</div>
```

```
var elem = document.querySelector("#address").;
var addrParas = elem.querySelectorAll("p");

// highlight paragraphs inside the div element
for (var i = 0; i < addrParas.length; i++) {
        addrParas[i].style.backgroundColor = "yellow";
}
```

- Access all paragraphs in the document/web page

```
var allParas = document.querySelectorAll("p");

// highlight all paragraphs in the web page
for (var i = 0; i < allParas.length; i++) {
        allParas[i].style.backgroundColor = "lightgreen";
}
```

- node-elements.html

Modifying DOM with JavaScript

Modifying DOM (HTML Structure and content) with JavaScript

Methods for selecting elements

- Selecting groups of elements within an element object, including document object

name	description
getElementsByTagName()	Returns a collection of all child elements with the specified tag name
getElementsByClassName()	Returns a collection of all child elements with the specified class name
querySelector()	Returns the first element that is a descendent of the element on which it is invoked that matches the specified group of selectors.
querySelectorAll()	Returns a non-live NodeList of all elements descended from the element on which it is invoked that match the specified group of CSS selectors.

Modifying the DOM tree

- Every DOM element object has these methods:

name	description
appendChild(node)	places given node at end of this node's child list
insertBefore(new, old)	places the given new node in this node's child list just before old child
removeChild(node)	removes given node from this node's child list
replaceChild(new, old)	replaces given child with new node

Creating new nodes

- Example 1: Add Node to the end of the web page

```
// create a new <h2> node
var newHeading =   document.createElement("h2");

//add text to using text node
var t = document.createTextNode(" This is a heading ");
newHeading.appendChild(t);

// append the node to the end of the web page
document.body.appendChild(newHeading);
```

- Note:
  - The create…() methods merely create a node but does not add it to the page/node.
  - You must add the new node as a child of an existing element on the page...

Adding new nodes to DOM tree

- Example 2: Add Node into another Node

```
var newNode = document.createElement("p");

//add text to using the innerHTML
newNode.innerHTML = "This a new paragraph";

// add the new element to the end of the "demo" div element.
var demo_div = document.querySelector("#demo");
demo_div.appendChild(newNode);
```

- node_appendChild.html

More examples

- More on modifying the DOM tree
  - node_insertBefore.html
  - node_removeChild.html
  - node_replaceChild.html

Modifying element / node attributes

- DOM nodes provide access to HTML attributes using the following standard methods:

name	description
hasAttribute(name)	checks if the attribute exists
getAttribute(name)	gets an attribute value
setAttribute(name, value)	sets an attribute
removeAttribute(name)	removes an attribute

Modifying element / node attributes

- Example 1:

```
var elem = document.querySelector("#d1");
elem.setAttribute("class", "notes");
//  or just simply:
//   elem.class = "notes";
```

- Example 2:

```
var image =   document.createElement("img");
image.height = "50";
image.src="http://www.senecacollege.ca/images/seneca-logo2.svg";

document.body.appendChild(image);
```

Using the innerHTML Property

- innerHTML: 2-way API for text and/or HTML tags in a node:
  - Get: `var txt = elem.innerHTML;`
  - Set: `elem.innerHTML = "<p>Paragraph changed!</p>";`
  - Append: `elem.innerHTML += "<p>Paragraph appended!</p>";`

  - innerHTML2.html

- Note: innerHTML property can be used to add elements / objects into a web page. But we typically use it create text/paragraph only, ∵
  - bad style on many levels (e.g. JS code embedded within HTML)
  - error-prone: must carefully distinguish
  - innerHTML should be mainly used to add plain text.
- Note: the "value" property of an HTML form control is similar to the innerHTML, but the "value" property can only accept plain text and only for form controls.

Modifying DOM with JavaScript

Modify the DOM (CSS formatting and appearances) with JavaScript

- Will be covered in week 9

Example: populates table using DOM API

```
function generateTable(){
   // get the reference for the body and creates a <tbody> element
   var tbl = document.querySelector("#outputTable");
   var tblBody = document.createElement("tbody");

   for (var i = 0; i < myData.length; i++) { // creating all table rows
      var row = document.createElement("tr");
      … …
}

function getTdElement(text) { // Create a <td> element and a text
   var cell = document.createElement("td");
    … …
}

function getTdLinkElement(text, href) { // Create a <td> element of a hyperlink
   var cell = document.createElement("td");
   … …
}
```

User	Address	Email
James	123 Keele St. Toronto, ON.	james12@myseneca.ca
Mary	92 Appleby Ave. Hamilton, ON.	mary356@myseneca.ca
Paul	70 The Pond Rd. North Youk, ON.	paul345@myseneca.ca
Catherine	1121 New St. Burlington, ON.	catherine89@myseneca.ca

- Note: Without using innerHTML property – better coding style.
- populate-table-dom.html

HTML DOM Events

DOM Events

- In general everything that happens in a browser may be called an event.
  - e.g. an event occurs when a user clicks on a link or a button in a form
- Every element on a web page has certain events which can trigger a JavaScript function.
  - JavaScript needs a way of detecting user actions so that it knows when to react.
  - It also needs to know which functions to execute.

Common Events

- Events triggered by user actions.
  - e.g.
    - When a user clicks the mouse
    - When a user strokes a key
    - When the mouse moves over an element
    - When an input field is changed
    - When an HTML form is submitted
- Events that are not directly caused by the user.
  - e.g.
    - When a web page has finished loading
    - When an image has been loaded
- Events category
  - Mouse events, keyboard events, HTML frame/object events, HTML form events, user interface events, touch events, …

Event Handlers

- Event Handlers are used to manipulate documents.
- An event handler is used in order to execute a script when an event occurs.
- The event handler has a prefix "on" followed by the event name.
- For example, the event handler for the click event is onclick.

```
<button onclick="myFunction()">Click me</button>
```

Creating Event Handler in HTML file

- General syntax:

```
<htmltag id="anid" eventHandler="JavaScript Code">
```

- e.g.

```
<input type="button" value="New Button!"
        onclick="console.log('some text');" />
```

- Note:
  - The event handlers in HTML must be enclosed in quotation marks.
  - Alternate double quotation marks with single quotation marks or \' or \" :

```
<input type="button" value="New Button!"
        onclick="console.log(\"some text\")" />
```

Creating Event Handler in JS file/code

- The HTML DOM allows you to create the event object in JavaScript file.
  - It's the better coding style: separating JavaScript code (behaviour) from HTML code (structure/content).
  - Syntax:

```
// the element is a DOM element object
Element.event = functionName;
```

Creating Event Handler in JS file /code

- Example

```
window.onload = function() { // why use window.onload?
    var elem = document.querySelector("#myBtn");

    elem.addEventListener( "click", displayDate );
    //or:  elem.onclick = displayDate;  // note: no "( )"
    //or:  elem.onclick = function () { displayDate(); };
};

function displayDate() {
    document.querySelector("#demo").innerHTML = (new Date()).toLocaleString();
}
```

- Note: event handler must be set after HTML document/page loaded into browser, so we need help from "window.onload" which is also an event handler
  - Or in HTML: <body onload="functionName()">

- js-event-init.html

Event Handler Examples

- onchange:

  occurs when the content of a field changes.

  - Applies to :

    select, input elements

  - Example:

    - In HTML:

      ```
      <input type='text' name='fullname' id='fullname' onchange='setOutput1()' >
      ```

    - Or in JavaScript:

      ```
      document.querySelector("#fullname").onchange = setOutput1;
      ```

  - js_onchange.html

Event Handler Examples

- onclick / ondblclick
  - Occurs when the user has click / double and released a mouse button (or keyboard equivalent) on an element.
  - Applies to:
    button, document, checkbox, link, radio, reset, submit
  - Example:
    - js_onclick.html

Event Handler Examples

- onfocus
  - Occurs when the user has given focus to an element.
  - Applies to

    button, checkbox, file, password, radio, reset, select, submit, text, textarea, window.
  - Example:
    - js-onfocus.html

Event Handler Examples

- onload
  - Occurs when a document or other external element has completed downloading all data into the browser.
  - Applies to

    image, window.
  - Example:
- js-event-init.html

Event Handler Examples

- onmouseout
  - Occurs when the user has rolled the mouse out of an element.
  - Applies to

    image, window.
  - Example:
    - js-onmouseout.html

Event Handler Examples

- onmouseover
  - Occurs when the user has rolled the mouse on top of an element.
  - Applies to

    image, window.
  - Example:
    - js-onmouseover.html

Event Handler Examples

- onresize
  - Occurs when the user has resized a window or object.
  - Applies to
    window.
  - Example:
    - js-onresize.html

HTML onbeforeunload Event

- onbeforeunload
  - The onbeforeunload event fires when the document is about to be unloaded.

- js-onbeforeunload.html

Timer Events

method	description
setTimeout(*function*, *delayMS*);	arranges to call given function after given delay in ms
setInterval(*function*, *delayMS*);	arranges to call function repeatedly every *delayMS* ms
clearTimeout(*timerID*); clearInterval(*timerID*);	stops the given timer so it will not call its function

- The first 2 methods return timerID which can be used to stop the timer by using the last 2 methods.
- E.g.

```
<script>
    setInterval(function(){
        document.body.innerHTML = new Date().toLocaleString();
    },1000);
</script>
```

Advanced: jQuery

- https://jquery.com/
- Some jQuery Functions And Their JavaScript Equivalents

Resourceful Links

- MDN: Introduction to DOM
- MDN: DOM Examples
- MDN: Creating New Elements
- JavaScript Kit: DOM Reference
- W3C: Handling events with JavaScript
- MDN – Node (interface) reference
- MDN – Element (interface) reference
- MDN – Text (interface) reference

Thank you!

Any Questions?