INT222 - Internet Fundamentals

Week 10: DOM and Events

Agenda

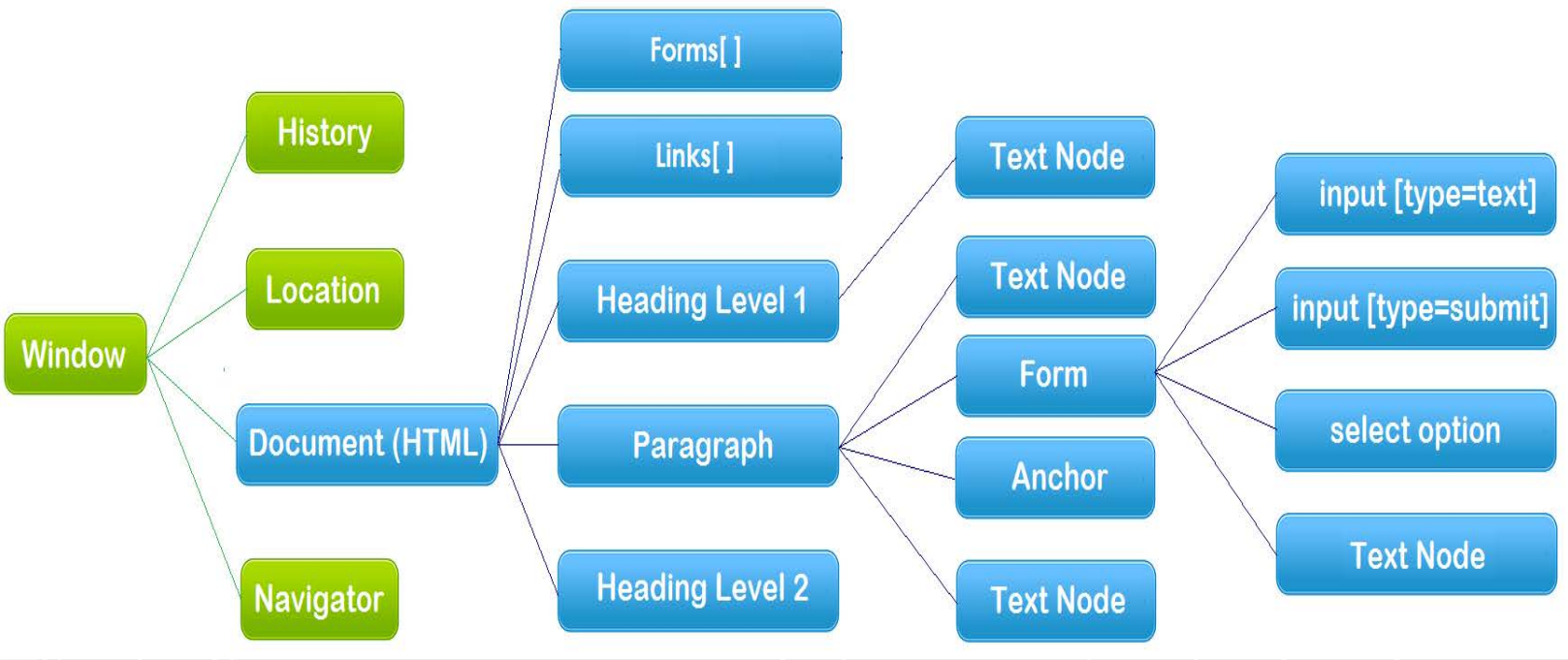
- Document Object Model (DOM)
- DOM Tree, Node
- DOM Events

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 manipulated.
  - 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

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.

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

Examples

- A form input element could be accessed as either
  - document.formName.inputElementName
  - document.forms[0].elements[0]
  - More...
- Two-way API example (run in Scratchpad):

  Get/Read:

```
var ttl = document.title;
alert(ttl);
```

  Set/Write:

```
document.title = "new title";
```

- 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 properties and methods:
  - document.createElement()
  - document.createTextNode()
  - document.createAttribute()
  - document.createComment()

  - document.addEventListener()

- create-element.html

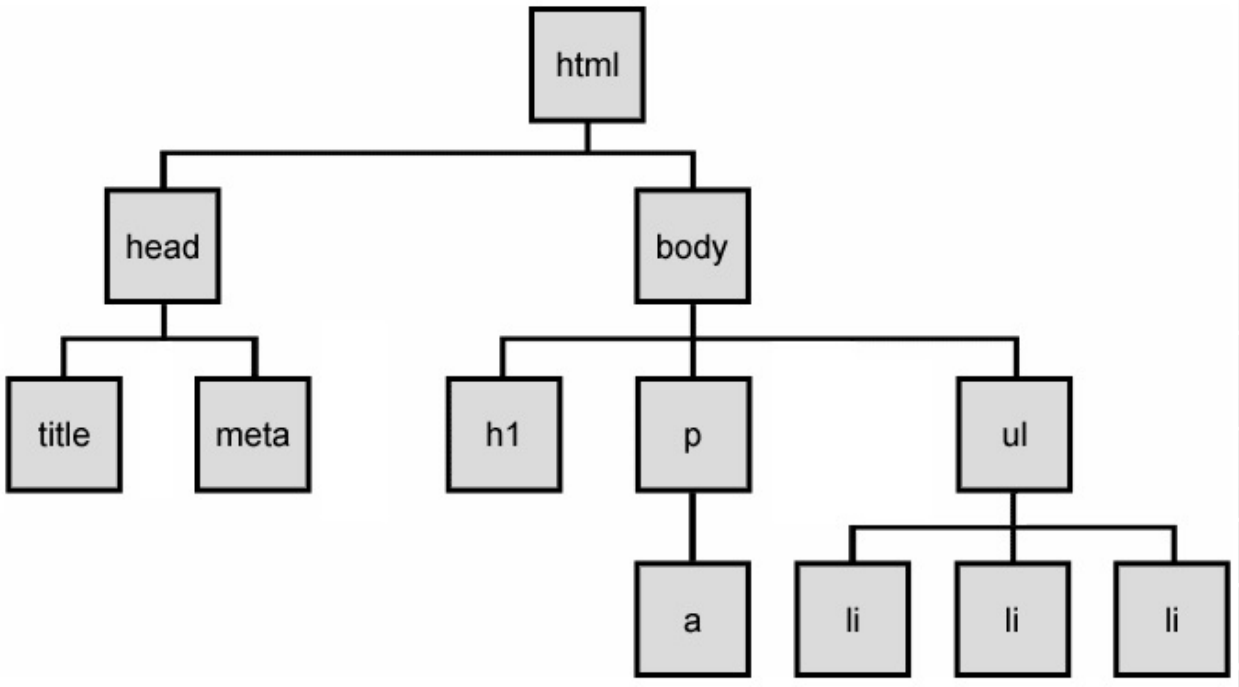
The DOM tree

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

```
<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

- A 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

Document Object Model (DOM)

- HTML documents are made up by HTML elements. In the HTML DOM, every HTML element is represented by an element object or node.

- In the HTML DOM (Document Object Model), everything is a node:

Types of DOM nodes

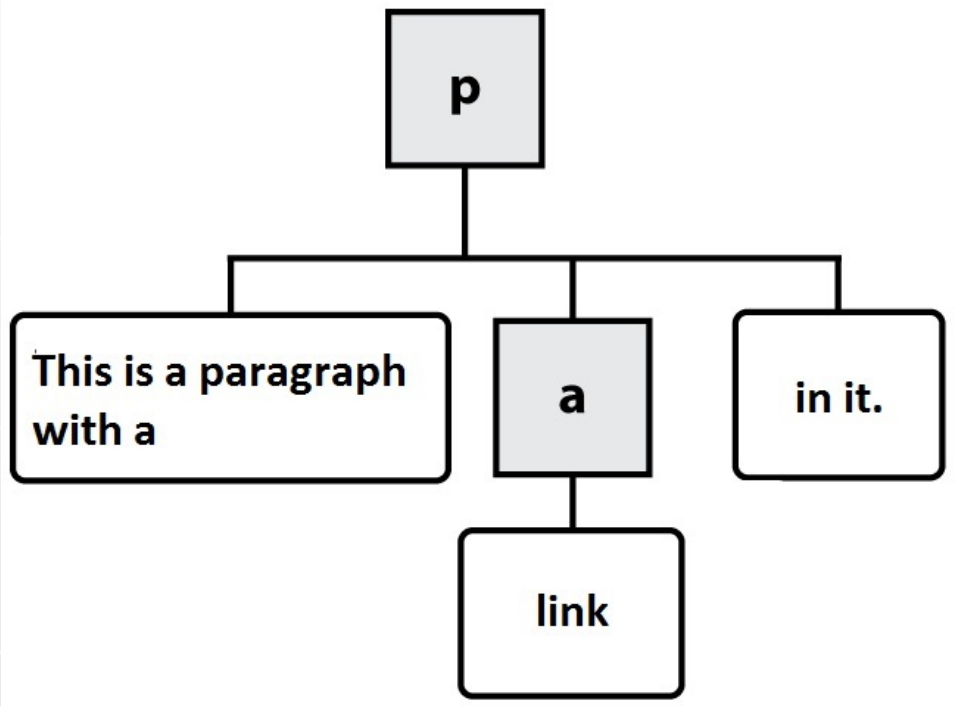
- element nodes (HTML tag)
  - For each HTML element, there is a node object.
  - It can have children and/or attributes
- text nodes (text in a block element)
- attribute nodes (attribute/value pair)
  - text/attributes are children in an element node
  - cannot have children or attributes
  - not usually 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



Element Nodes/Objects

- Properties and Methods of element nodes/objects – the standard API
- Properties:

  element.className
  element.id
  element.style

  element.innerHTML
  element.parentNode
  element.nextSibling
  ... ...

- Methods:

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

  element.addEventListener()

  element.appendChild()
  element.insertBefore()
  element.removeChild()

  element.setAttribute()
  element.getAttribute()

- The HTML DOM Element Object

Example

- Highlight all paragraphs within a div element:
  - HTML

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

  - JavaScript

```
var elem = document.getElementById("address").;
var addrParas = elem.getElementsByTagName("p");

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

- node-elements.html

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.

- ✓ Try: use querySelectorAll() for the last example ...

Modifying DOM with JavaScript

Modifying HTML Structure and content with JavaScript

Creating New Nodes

name	description
document.createElement("tag")	creates and returns a new empty DOM node representing an element of that type
document.createTextNode("text")	creates and returns a text node containing given text

- It merely creats a node does not add it to the page
- you must add the new node as a child of an existing element on the page...
- e.g. (run in Firefox Scratchpad)

```
// create a new <h2> node
var newHeading =   document.createElement("h2");
var t = document.createTextNode(" This is a heading ");
newHeading.appendChild(t);

newHeading.style.color = "blue";
document.body.appendChild(newHeading);
```

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

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

- Example 2:

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

- Example 3:

```
document.getElementById("image").height = "200";
document.getElementById("image").src="pic2.jpg";
```

innerHTML Property

- Example – 2-way API:
  - innerHTML2.html
- Note: innerHTML property can be used to add elements / objects into a web page. e.g.

```
document.getElementById("xy12").innerHTML +=
        "<p style='color: red; " +
       "margin-left: 50px;' " +
       "onclick='myOnClick();'>" +
       "A paragraph!</p>";
```

  - 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.
- For inserting elements/objects into a page, use DOM (other properties and methods).

Modifying DOM with JavaScript

- Modify HTML formatting and appearances

Changing HTML style with style object

- style object can be used to set any CSS style properties of HTML elements.
- Syntax:
  document.getElementById(*id*).style.*property = new style;*
- e.g.

```
// highlight all paragraphs in the document
var allParas = document.getElementsByTagName("p");

for (var i = 0; i < allParas.length; i++) {
    allParas[i].style.backgroundColor = "yellow";
}
```

HTML DOM Style Object

- Style Object Properties

Property	Description
backgroundColor	Sets or returns the background-color of an element
border	Sets or returns borderWidth, borderStyle, and borderColor in one declaration
borderColor	Sets or returns the color of an element's border (can have up to four values)
color	Sets or returns the color of the text
display	Sets or returns an element's display type
fontSize	Sets or returns the font size of the text
fontWeight	Sets or returns the boldness of the font
margin	Sets or returns the margins of an element (can have up to four values)
padding	Sets or returns the padding of an element (can have up to four values)
textDecoration	Sets or returns the decoration of a text
minHeight	Sets or returns the minimum height of an element
minWidth	Sets or returns the minimum width of an element
visibility	Sets or returns whether an element should be visible

Changing HTML style setAttribute()

- Syntax:

  document.getElementById(*id*).setAttribute("style", new styles);

- e.g.

```
var elem = getElementById("xy1");
// set multiple style properties in one statement
elem.setAttribute("style", "width:200;background:blue; ");
```

HTML DOM Events

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

- General syntax:

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

- e.g.

```
<input type="button" value="New Button!"
        onclick="alert('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="alert(\"some text\")" />
```

Creating Event Handler – Using DOM

- The HTML DOM allows you to create the event object to the element using JavaScript
  - better coding style: separation of structure and behaviour

```
<script>
  window.onload = function() { // why use window.onload?
      var elem = document.getElementById("myBtn");

      elem.onclick = displayDate;
      // elem.addEventListener( "click", displayDate );  //  also works
  };

  function displayDate() {
     document.getElementById("demo").innerHTML = (new Date()).toLocaleString();
  }
</script>
```

- js-event-init.html

Event Handler Examples

- onchange:

  occurs when the content of a field changes.

  - Applies to :

    select, text, input elements

  - Example:

  - js_onchange.html

Event Handler Examples

- onclick
  - Occurs when the user has pressed 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

- ondblclick
  - Occurs when the user has double-clicked a mouse button on an element.
  - Applies to the following HTML tags:
    document, image button elements, link
  - Example:
    - js-ondblclick.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-onload.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.

```
<button onclick="setInterval(function(){alert('Hello')},3000);">
Try it</button>
<!-- this code alert “hello” every 3 seconds. -->
```

Resourceful Links

- MDN: Introduction to DOM
- MDN: DOM Examples
- MDN: Creating New Elements
- JavaScript Kit: DOM Reference
- W3C: Handling events with JavaScript

Thank You!