SWINBURNE UNIVERSITY OF TECHNOLOGY

COS10011/60004
Creating Web Applications

Lecture 7
JavaScript 3

Previously – Linking JavaScript to HTML

HTML - *content*

```
<!DOCTYPE html>
<html lang="en">
<head>
    …
    <script src="my_jsfile.js"></script>
</head>
<body>
    ...
    <p><span id="mymessage"></span></p>
    <p><button type="button" id="clickme">
    Click Me!</button></p>>
    ...
</body>
</html>
```

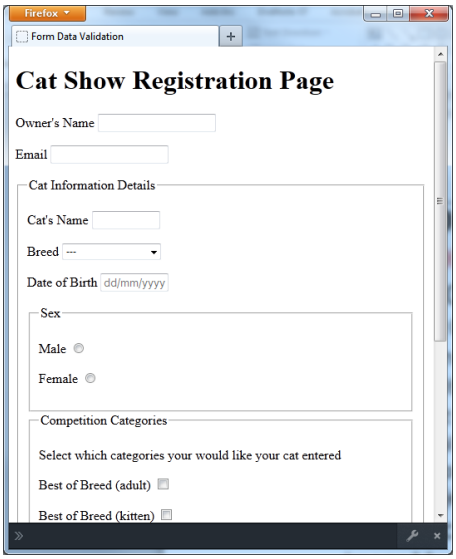
JavaScript - *behaviour*

```
/* Filename: my_jsfile.js
   …
*/

function doSomething()
{
    var myString, outputMessage;     //declare local variables
    myString = prompt("Enter the string", "The string");
    alert("Your output: " + myString);
    outputMessage = document.getElementById("mymessage");
    outputMessage.textContent="Your output: " + myString;
}

function init() {
    var clickme = document.getElementById("clickme");
    clickme.onclick = doSomething;
    }

window.onload = init;
```


© Swinburne University of Technology

Previously – Form Validation

- Regular Expressions
- Input data validation using JavaScript



Contents

- Predefined Objects and JavaScript
  - **JavaScript Core Objects – examples Array, Date, String**
  - **Global Functions**
  - Browser Objects – window navigator
  - Document Object Model
    - General DOM
    - HTML objects
    - CSS objects
- Using JavaScript
  - Image Manipulation: *an Example*
- Storing 'State'
  - Web Storage
  - Cookies
- Multiple files
  - One HTML : many JS
  - One JS : many HTML

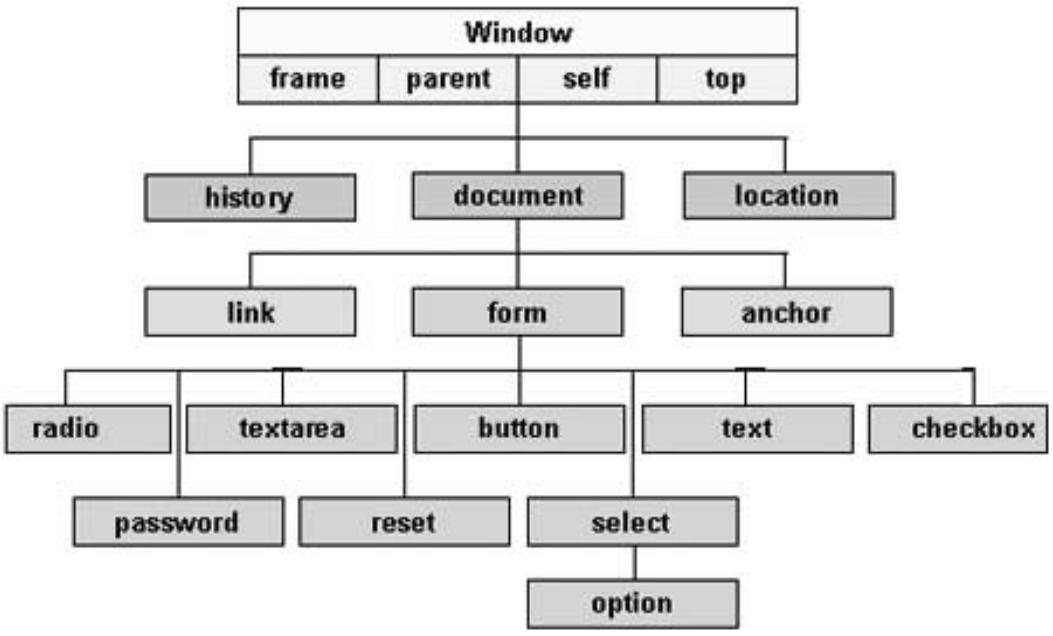
Last week

Contents

Document Object Model and JavaScript

- Predefined Objects
  - Browser Objects – window navigator
  - Document Object Model
    - General DOM
    - HTML objects
    - CSS objects
- Using JavaScript
  - Image Manipulation: *an Example*
- Storing 'State'
  - Web Storage
  - Cookies
- Multiple files
  - One HTML : many JS
  - One JS : many HTML

DOM object hierarchy - examples



Predefined Objects - Browser Objects

- Window
  - document
- Navigator
- Screen
- History
- Location

Examples

```
window.alert("Hello");
var ans=confirm("Are you sure?")
```

document is the main object of the window object.
This will be discussed in detail later

We will not discuss these in detail but you might find them useful.

Window Object – Properties

- The **`window`** object is at the top of the hierarchy, and so its properties and methods may be used without explicitly referring to the "window" object.
  eg. `document` is same as `window.document`

- **Properties**:

`document`	- returns a reference to the document contained in the window
`location`	- gets/sets the location, or current URL, of the window object
`history`	- returns a reference to the history object, an array of visited URLs
`name`	- gets/sets the window's name
`navigator`	- returns a reference to the navigator object
`defaultStatus`	- gets/sets the message in the status bar
`status`	- gets/sets the transient message in the status bar
`self`	- identifies the current window being referenced
`parent`	- identifies the window containing a particular window

Note: *This is **not** a complete list of properties! For more information see:*
https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model

Window Object – Methods

- **Methods** *(this is **not** a complete list of methods)*

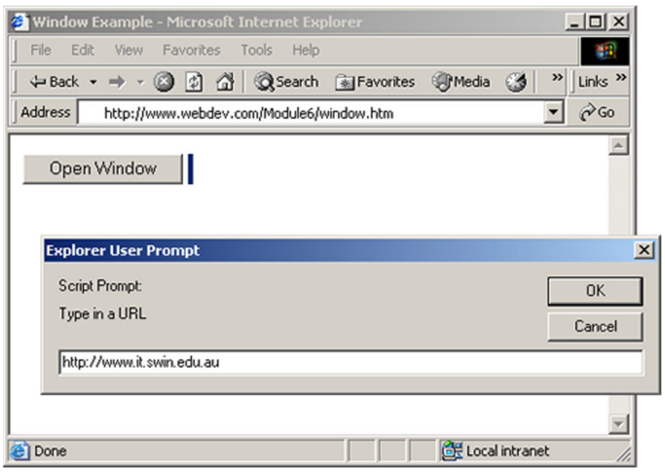
  `alert(text)` - pops up an alert box
  `confirm(text)` - pops up a box with 'OK' or 'Cancel'
  `prompt(text,def)` - retrieves a line of text from the user
  `open(url,[ops])` - opens up a new window
  `close()` - closes a window
  `focus()` - gives focus to a window
  `blur()` - removes focus from a window

- **Window HTML Event Handling**

  `onload` - occurs when the page has completed the loading process.
  `onunload` - occurs just before the document is cleared from the browser window. Usually used for background statistical purposes etc.

Window Object – Example

```
function newWindow() {
    theUrl = prompt("Type in a URL",
                                window.location);
    window.open(theUrl);
}
```



Contents

Document Object Model and JavaScript

- Predefined Objects
  - Browser Objects – window navigator
  - Document Object Model
    - General DOM
    - HTML objects
    - CSS objects
- Using JavaScript
  - Image Manipulation: *an Example*
- Storing 'State'
  - Web Storage
  - Cookies
- Multiple files
  - One HTML : many JS
  - One JS : many HTML

In simple terms, the DOM is a programming interface for HTML, XML and SVG documents. It provides a way to represent a document tree, navigate through a document, and reference objects.

Document Object Model (DOM)

- a platform and language neutral interface to allow programs and scripts to dynamically access and update the content, structure and style of a document [W3C]

- a way to represent and navigate an HTML document or any XML document as a tree.

Note: The DOM Core applies to any XML, and any HTML that complies with XML.

DOM Levels

- The W3C has developed DOM "levels" to represent the different features that may be supported
  - DOM Level 0: The earlier *"vendor specific* intermediate" DOMs
  - DOM Level 1: HTML & XML document tree structures, including HTML specific elements and node add / move / delete.
  - DOM Level 2: XML namespaces, styles, views, and events
  - **DOM Level 3:** Divided into specific modular sections
  - ***DOM Level 4:*** Aims at supporting mutimedia, and removing things that haven't been implemented

Different browser provide various levels of support for DOM.

From our demo ....

```
function isCategorySelected(){
    ...
    var categories =
      document.getElementById("categories").getElementsByTagName("input");
    var labels =
       document.getElementById("categories").getElementsByTagName("label");
    var label = "";
    var catList = "";
    for (i=0; i<categories.length; i++){          //for each category element
        selected = selected ||  categories[i].checked;   //see if it is checked
        label = labels[i].firstChild.textContent;        //get its label
        catList = catList + label + "\n";
    }
    ...
}
```

text node content

```
<fieldset id="categories">
        <legend>Competition Categories</legend>
        <p>Select which categories your would like your cat entered</p>
        <p><label for="bestbreed">Best of Breed (adult)</label>
                <input type="checkbox" id="bestbreed" name="categories[] " value="best"/>
        </p>
        <p><label for="kit">Best of Breed (kitten)</label>
                <input type="checkbox" id="kit" name="categories[]" value="kitten"/>
        </p>
```


Contents

Document Object Model and JavaScript

- Predefined Objects
  - Browser Objects – window navigator
  - Document Object Model
    - General DOM (see reference slides below)
    - HTML objects
    - CSS objects
- Using JavaScript
  - Image Manipulation: *an Example*
- Storing 'State'
  - Web Storage
  - Cookies
- Multiple files
  - One HTML : many JS
  - One JS : many HTML

Document Object – as Element

Element properties

`objElement.`

- `id`
- `className`
- `tagName`
- `getElementsByTagName()`
- `getAttribute()`
- `setAttribute()`
- `removeAttribute()`

`objElement`, shown here is just a sample object defined from the DOM

```
objElement = document.documentElement;
```

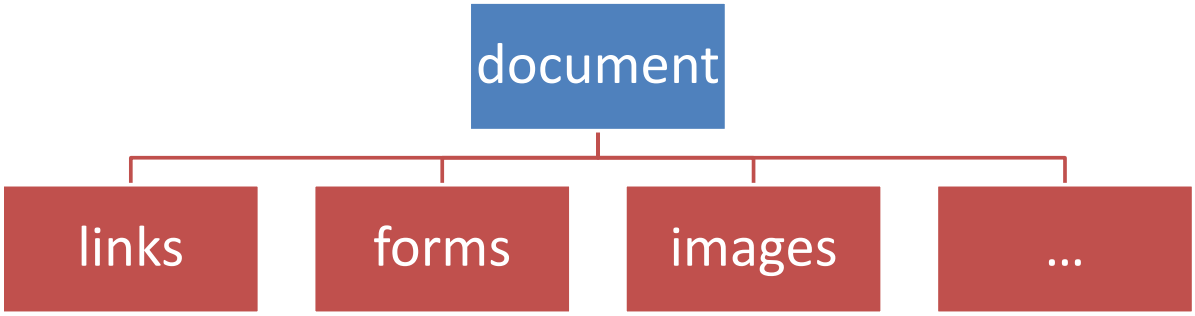
or

```
objElement =
    document.getElementById("pgHead");
```

For example, `myElement.tagName`

Predefined Objects - Document Object

HTML document object and its collection objects



- These are collections of specific objects, e.g. form**s** is a collection of form objects.

Note: collection names are often expressed in plural form

https://developer.mozilla.org/en-US/docs/Web/API/Document

DOM - Document Methods revisited

- Some useful methods of **`document`** object

```
getElementById()
getElements ByName()
getElements ByTagName()
createElement()
createTextNode()
createAttribute()
```

Pre-defined object

HTML DOM Events (revisited)

- Mouse events
  - onclick, ondblclick, onmouseup, on mousedown, onmouseover, onmousemove, …
- Keyboard events
  - onkeydown, onkeyup, onkeypress, …
- Form events
  - onblur, onchange, onfocus, oninvalid, onsubmit, …
- Drag events, animation, clipboard, print, media, transition, …

http://www.w3schools.com/jsref/dom_obj_event.asp

From our demo ….

```
//register onblur events for all the input elements
function registerInputsOnBlur(){
        var inputElements =
                document.getElementById("regForm")
                .getElementsByTagName("input");
    for (var i = 0; i < inputElements.length; i++){
        inputElements[i].onblur = validateInputOnBlur;
    }
}
function validateInputOnBlur(){
    var objectLostFocus_id = this.id;
    var isOk = false;
    switch (objectLostFocus_id){
        case "owner":
        …
```

The keyword "**this**" refers to the object that will fire (trigger) the checkdata function (method). Use an if statement to check the "id" value and perform corresponding tests.

Document Object – as Element

- The following HTML elements have additional properties:
  - Links `<a …>…</a>`
  - Forms `<form …>…</form>`
  - Select / Option elements `<select …>… </select>`
  - Input (text, radio, checkbox, password, hidden, submit) `<input … />`
  - Textarea `<textarea… >… </textarea>`
  - Images `<img … />`

Document Object - Examples

- Get all images from the body element

```
var imgElements =
 document.getElementsByTagName("img");
```

Will return a collection/array. Use a **plural** object name to indicate multiple elements

Document Object - Examples

- Get the element with **id="intro"**

```
var introElement =
  document.getElementById("intro");
```

Use a **singular** object name to indicate 1 element

- Get all **<p>** elements that are descendants of the element with **id="main"**

```
var mainParagraphElements =
  document.getElementById("main")
          .getElementsByTagName("p");
```

plural

Will return a collection/array.

Contents

Document Object (Style)

From our demo ….

```
function chkOwnerName () {
    //check owner name valid
    var owner = document.getElementById("owner").value;
    …
     //highlight the textbox if not valid
    if (!nameOk){
    document.getElementById("owner").style.borderColor = "red";
    }
    return  nameOk;
}
```

Why not *border-color* ????

Document Object (Style)

- **Style** properties are typically hyphenated words, **but this *does not work* in JavaScript**, so CSS style properties are joined together using ‘camelCase’ notation. e.g.

  `some-css-property` becomes

  `someCssProperty`

Document Object (Class and Style)

- **`class`** is often used to associate style with elements. If we change the class in JavaScript, the browser changes the associated presentation

```
objElement.className = "styleRule2";
```

- Usually element ***attribute names*** are directly matched to DOM property names.
  For example the **`href`** attribute

```
<a href="page1.htm" class="button">
```

  is mapped to `objElement.href`

- But the **`class`** attribute
  is mapped to `objElement.className`
  NOT `".class"` as "class" is a ***reserved word*** in JavaScript

Document Object (Class and Style)

- **`objElement.style.`**

```
background
backgroundAttachment
backgroundColor
backgroundImage
backgroundPosition
backgroundPositionX
backgroundPositionY
backgroundRepeat
```

```
border
borderCollapse
borderColor
borderSpacing
borderSpacing
borderStyle
border[side]
border[side]Color
border[side]Style
border[side]Width
```

For example,

```
objElement.style.display
```


Contents

Document Object Model and JavaScript

- Predefined Objects
  - Browser Objects – window navigator
  - Document Object Model
    - General DOM (see reference slides below)
    - HTML objects
    - CSS objects
- Using JavaScript
  - Image Manipulation: *an Example*
- Storing 'State'
  - Web Storage
  - Cookies (see reference slides below)
- Multiple files
  - One HTML : many JS
  - One JS : many HTML

Content and JavaScript

JavaScript can enrich user experiences by changing content and providing:

- slideshows,
- **cycling images,**
- 'drag and drop' interfaces,
- re-sorting / re-displaying page information,
- hiding /showing page information,

… and lots more …

Cycling Images - Example

Given the following HTML page segment,

*take note of the **IDs***

```
<h1>Cycling an image</h1>
<button id="startSlides">Start</button>
<button id="stopSlides">Stop</button>
<figure>
   <img src="img1.jpg" id="picImage"
      alt="Swinburne" />
   <figcaption id="picText">Swinburne Pics
   </figcaption>
</figure>
```


Cycling Images - Example (continued)

Using the JavaScript template:

```
function cycleImage() {
 // see next slide
}
function startSlideShow(){
     running = setInterval("cycleImage()", 2000);  //2 secs
}
function stopSlideShow(){
     window.clearInterval(running);
}
function init() {
   document.getElementById("startSlides").onclick
      = startSlideShow;
   document.getElementById("stopSlides").onclick
      = stopSlideShow;
}
window.onload = init;
```


Cycling Images - Example (continued)

```
var currentImg = 0;  // set start position as global
var theImages
      = new Array("img1.jpg","img2.jpg","img3.jpg");
var theTexts  = new Array("text1","text2","text3");
var numImgs = theImages.length;

function cycleImage() {
   var figImg = document.getElementById("picImage");
   var figCap = document.getElementById("picText");
   if(document.images) {
       currentImg++;
       currentImg = currentImg%numImgs;
       figImg.src = theImages[currentImg];
       figCap.textContent = theTexts[currentImg];
   }
```


Contents

Document Object Model and JavaScript

- Predefined Objects
  - Browser Objects – window navigator
  - Document Object Model
    - General DOM
    - HTML objects
    - CSS objects
- Using JavaScript
  - Image Manipulation: *an Example*
- Storing 'State'
  - Web Storage
- Multiple files
  - One HTML : many JS
  - One JS : many HTML

Web Storage

Web storage

- allows HTML5 web pages to store data ***locally*** *within the browser*
- stores data in key/value pairs
- is more secure and faster compared to cookies
  (data is not included as part of the HTTP header)
- can only be used to access data by the web site that created it
- allows the storage of a *large amounts* of data
  (at least 5mb per origin depending on browser)
- can only by accessed by client scripts

Web Storage (continued)

Two objects for storing data

- **localStorage**
  - stores data with no expiration, even when the browser is closed
- **sessionStorage**
  - stores data for one session, defined by the lifetime of the current window.

Web Storage (continued)

Can check if Web Storage is supported

```
if(typeof(Storage)!=="undefined"){
 // localStorage and
       sessionStorage supported

 }else {
   // No web storage supported.
 }
```


Web Storage (continued)

Setting and reading sessionStorage

Store value on browser only for the session

```
sessionStorage.setItem('key', 'value');
```

Examples

sessionStorage.uname = document.getElementById("username").value;

or

sessionStorage.setItem(keyname,valuename);

Retrieve value for the session

```
var a = sessionStorage.getItem('key');
```

Examples

var a = sessionStorage. uname;

or

var a =sessionStorage.getItem("uname");

Web Storage (continued)

Setting and reading localStorage

Store value on the browser

```
localStorage.setItem('someKey', 'somevalue');
```

or

```
localStorage.someKey = 'someValue';
```

Retrieve value, even after re-opening browser

```
var someValue = localStorage.getItem('someKey');
```

or

```
var someValue = localStorage.someKey;
```


Storage interface

Properties

sessionStorage.length / localStorage.length (Read only)

Returns an integer representing the number of data items stored in the Storage object.

Methods

sessionStorage.key() / localStorage.key()

When passed n, this method will return the name of the nth key in the storage.

sessionStorage.getItem() / localStorage.getItem()

When passed a key name, will return that key's value.

sessionStorage.setItem() / localStorage.setItem()

When passed a key name and value, will add that key to the storage, or update that key's value if it already exists.

sessionStorage.removeItem() / localStorage.removeItem()

When passed a key name, will remove that key from the storage.

sessionStorage.clear() / localStorage.clear()

When invoked, will empty all keys out of the storage.

https://developer.mozilla.org/en-US/docs/Web/API/Storage

Contents

Document Object Model and JavaScript

- Predefined Objects
  - Browser Objects – window navigator
  - Document Object Model
    - General DOM
    - HTML objects
    - CSS objects
- Using JavaScript
  - Image Manipulation: *an Example*
- Storing 'State'
  - Web Storage
- **Multiple files**
  - **One HTML : many JS**
  - **One JS : many HTML**

Reference Slides

- Navigator and other browser objects
- Document Object properties
- Document Objects as Node
- Cookies

Navigator Object

- The **`navigator`** object does not fall within the normal Browser window object hierarchy.
  (It relates to the 'environment' in which the window sits)
- The **`navigator`** object ***may*** be used to gather information about the **client platform**. eg. if it has GPS
- The **`navigator`** object ***was*** often used to identify **browser dependent** features that a script may need to use.

```
if (navigator.appName == "Netscape") {
  // insert code here for Netscape
} else {
  // insert code here for other
  // browsers
}
```

Now best to use other DOM methods
http://www.w3.org/TR/html5/webappapis.html#the-navigator-object

Navigator Object – Properties/Methods

Properties

`appCodeName`	The coded name of the browser
`appName`	The name of the browser
`appVersion`	The version of the browser
`language`	The language supported by the browser
`mimeTypes[]`	An array of the MIME types recognised
`platform`	The platform the browser is running on
`plugins[]`	An array of the plugins installed

Many of these properties are superseded.
See HTML5 spec., device guides.

http://www.w3.org/TR/html5/webappapis.html#the-navigator-object

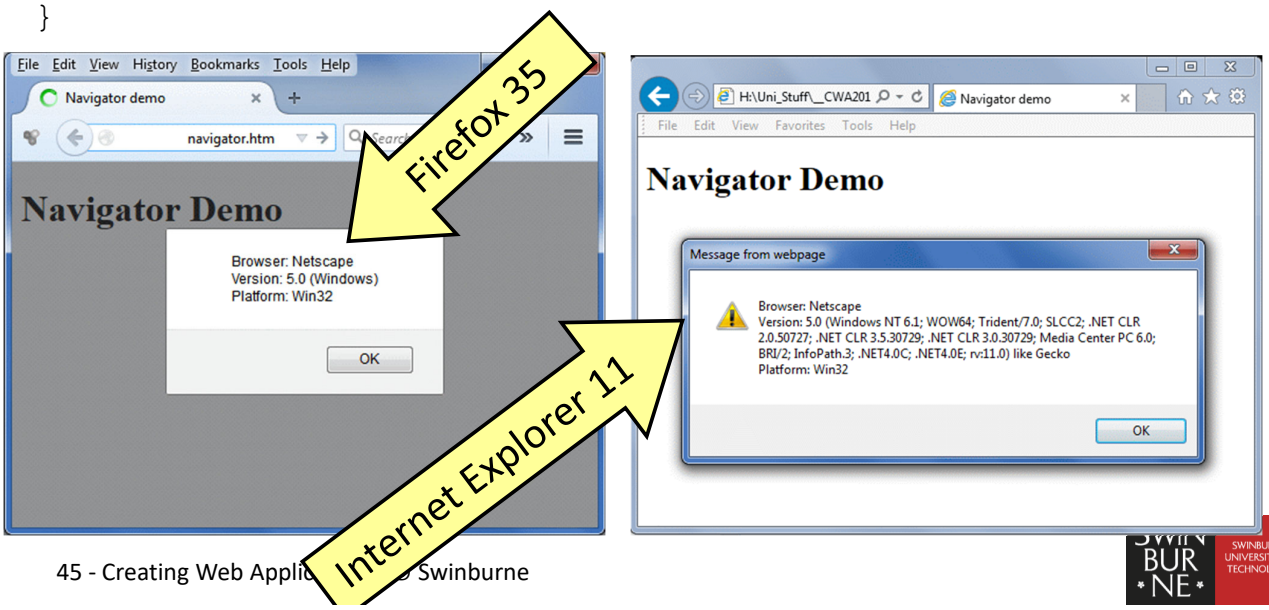
Methods

`javaEnabled()`	Returns true if the browser supports Java applets
`preferences()`	Checks or sets user preferences

Note: *Properties and Methods may differ between browsers.*

Navigator Object – Example

```
function showInfo() {
  var msg="Browser: " + navigator.appName +"\n";
  msg += "Version: " + navigator.appVersion + "\n";
  msg += "Platform: " + navigator.platform + "\n";
  alert(msg);
}
```



Other Browser Objects

history .back(),
.forward(),
.go(n)

Avoid using these. Changing them can confuse users.

location .href,
.host,
.pathname,
.protocol,
.search,
.reload([*force*]),
.replace(URL)

Useful for redirection, and for determining current webpage, so scripts can enhance menus by highlighting the current page.

Reference Slides

- Navigator and other browser objects
- **Document Object properties**
- Document Objects as Node
- Cookies

Document Object - Anchor Element

Anchor Element `<a ></a>`

```
objElement.
      href
      rel
      target
```

For example, `myAnchor.href`

Document Object - Form Element

Form Element `<form …>…</form>`

```
objElement.
      elements[]
      length
      action
      method
      enctype
      target
      submit()
      reset()
```

An array of all the elements in the form

For example, `myForm.length`

Document Object - Select Element

Select Element `<select …>…</select>`

```
objElement.
      type                disabled
      selectedIndex       multiple
      value               name
      length              size
      form                add()
      options[]           remove()  …
```

For example, `mySelect.value`

Document Object - Option Element

Option Element `<option …>…</option>`

```
objElement.
      form
      text
      disabled
      selected
      value, …
```

For example, `myOption.text`

Document Object - Input Element

Input Element `<input … />`

```
objElement.
      form              readOnly
      checked           value
      disabled          select()
      name              click(), …
```

For example, `myInput.checked`

Document Object - Textarea Element

Text Area Element `<textarea …>…</textarea>`

```
objElement.
    form
    disabled
    name
    readOnly
    value
    select(), …
```

For example, `myTextArea.value`

Document Object - Image Element

Image Element `<img … />`

```
objElement.
    name
    src
    alt …
```

For example, `myImage.src`

Reference Slides

- Navigator and other browser objects
- Document Object properties
- **Document Objects as Node**
- Cookies

SWINBURNE UNIVERSITY OF TECHNOLOGY

Document Object – as Node

- Use document property and method to obtain as node

```
node2 = document.getElementById("pgHead");
```

- What are some properties of a node?

```
node2.nodeType
node2.nodeName
node2.nodeValue
```

Number type

String type

String type

SWINBURNE UNIVERSITY OF TECHNOLOGY

Document Object – as Node

- The **nodeType** property returns the type of node.
  - nodeType is *read only*.
- The most important node types are:

Element Type	NodeType
Element	1
Attribute	2
Text	3
Comment	8
Document	9

Document Object – as Node

- The **`nodeName`** property
  - specifies the name of a node
  - is *read-only*
  - of an **element** node is the same as the element name
  - of an **attribute** node is the attribute name
  - of a **text** node is always #text
  - of the **document** node is always #document

For HTML, nodeName always contains the *uppercase* element name of an HTML element.

Document Object – as Node

- The **`nodeValue`** property
  - specifies the value of a node.
  - for **element** nodes is undefined
  - for **text** nodes is the text itself
  - for **attribute** nodes is the attribute value
  - can be changed

Document Object – as Node

Other node properties

```
theNode.
    nodeType
    parentNode
    firstChild
    lastChild
    previousSibling
    nextSibling
    children[]
    childNodes[]
```

`theNode`, shown here is just a sample object defined from the DOM

```
theNode = document.documentElement;
or
theNode = document.getElementById("pgHead");
```

ref demo example

For example, `myNode.nodeType`

Reference Slides

- Navigator and other browser objects
- Document Object properties
- Document Objects as Node
- **Cookies**

Cookies

- A Cookie is a variable that contains ***a small piece of information*** that can be passed by a **web server** to the client **browser**.
- This variable is ***stored in the client machine*** through the browser.
- The browser may chose not to accept a cookie
- A Cookie:
  - is stored as plain text record (maximum of 4Kb)
  - can be accessed by client and sent back with **HTTP Request** to **web server**
- **Reference:** https://developer.mozilla.org/en-US/docs/DOM/document.cookie

Cookies (continued)

The text record consists of the following variable-length fields:

- **name=*value*** pair used to set cookies
- **domain=*hostName*** is the domain name where the cookie can be used.
- **path=*directoryPath*** is the path to the directory where the cookie can be used.
  This is usually the path to the web page that set the cookie. Webpages from a different directory can access the cookie if left blank.

Cookies (continued)

… Cookie text record continued

- **expires=*stringDate*** is the date when the cookie will expire. If blank, the cookie will expire when browser is closed.
- **secure** is use to restrict the retrieval of the cookie from a secure server. If left blank, no such restriction exists.

Cookies – Checking

Can check if Cookies are enabled

```
if(navigator.cookieEnabled)){
  // cookies enabled

}else {
  // cookies disabled
}
```


Cookies – Setting

Setting a cookie record with no expiration:

```
document.cookie =
 "lname=Smith;fname=Jack;"
```

Setting a cookie record with expiration (session)

```
now = new Date();
document.cookie =
 "lname=Smith;fname=Jack; expires="
 + now.toUTCString()
 + ";domain=.swinburne.edu.au;
    path=/;secure;"
```


Cookies – Setting

Syntax to manage cookies

```
document.cookie = "field=value;";
```

Document object

Setting field values

Note:
Cookie *values* may not include semicolons, commas, or whitespace, use the JavaScript escape() and unescape() functions to encode and decode the value respectively

Cookies – Setting (continued)

Wrong way, there are 6 Cookie records here

```
document.cookie = "lname=Smith;";
document.cookie = "fname=Jack;";
document.cookie = "expires=" +
        now.toUTCString() + ";"
document.cookie =
        "domain=.swinburne.edu.au;"
document.cookie = "path=/;"
document.cookie = "secure;"
```


Cookies – Deleting

Setting expiration date (deleting a cookie)

```
expireDate = new Date();
expireDate.setTime(expireDate.getTime()
     + 3600000*24* ______);
```

Replace with – to delete cookies

Replace with number of days

```
document.cookie = "key=value;expires=" +
     expireDate.toUTCString() + ";"
```


Cookies – Reading

```
// Get all the cookies pairs
var allCookies = document.cookie;
// Split each pair as an element in an array
cookieArray  = allCookies.split(';');
// Access each pair as an element
for(var i=0; i<cookieArray.length; i++){
  // split each element into name and value
  name = cookieArray[i].split('=')[0];
  value = cookieArray[i].split('=')[1];
  alert("Key is " + name +
        " and value is " + value);
}
```


What's Next?
- Introduction to Server-Side Processing