UTM
UNIVERSITI TEKNOLOGI MALAYSIA

Online Learning

JavaScript - DOM

Sarina Sulaiman

Faculty of Computing

JS: Document Object Model (DOM)

- DOM stands for Document Object Model, and allows programmers generic access to:
  - adding, deleting, and manipulating - of all styles, attributes, and elements in a document.
  - It can be accessed via any language available in the browser, including Java, JavaScript/ECMAScript/JScript, and VBScript (MSIE only).
  - The DOM is supported most completely starting in IE 5 and Gecko (NS6 and upwards, such as Firefox.)
- Every tag, attribute, style, and piece of text is available to be accessed and manipulated via the DOM -- the possibilities are endless:
  - adding and removing tags,
  - attributes and styles,
  - animating existing elements,
  - and hiding/ showing elements on a page.

JS: Document Object Model (DOM)

- The DOM is constantly being revised by the W3C, with browsers at the same time constantly trying to support the latest recommended version of the DOM.
- As of IE6 and Firefox 1.0, DOM 2 best encompasses what the two browsers currently support.
- DOM 3 is the next major version in the works.
- Before we get started, you need to know a few terms that we will use:
  - Node: A reference to an element, its attributes, or text from the document.
  - Element: A representation of a <TAG>.
  - Attribute: A property of an element. HREF is an attribute of <A>, for example.

JS: Document Object Model (DOM)

- The DOM represent an HTML documents as a tree of objects.
- The tree representation of an HTML document contains nodes representing HTML tags or elements, such as <body> and <p>, and nodes representing strings of text.
- HTML document may also contain nodes representing HTML comments.

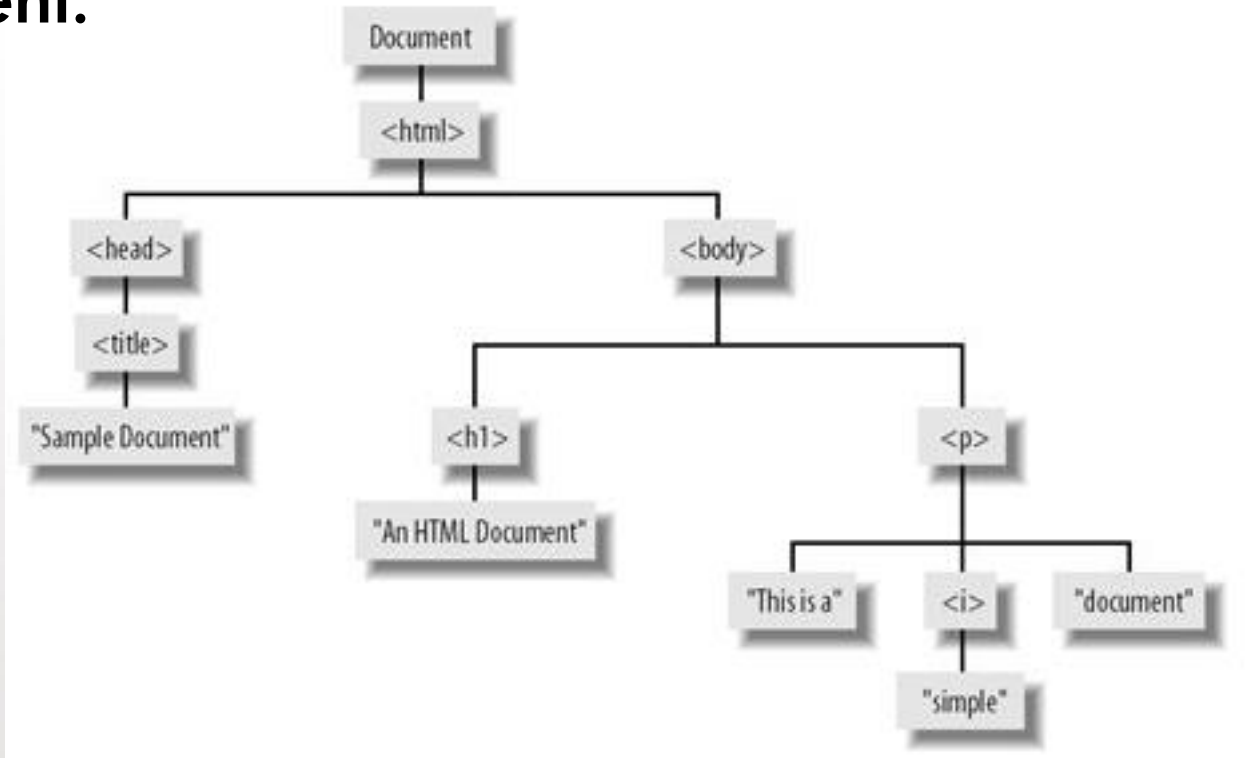
JS: Document Object Model (DOM)

- **Consider the following simple HTML document:**

```
<html>
    <head>
        <title>Sample Document</title>
    </head>
    <body>
        <h1>An HTML Document</h1>
        <p>This is a <i>simple</i> document</p>
    </body>
</html>
```

JS: Document Object Model (DOM)

- **The tree representation of the previous HTML document:**



JS: Document Object Model (DOM)

- **The node relationships of the previous DOM tree:**
  - HTML is an **ancestor** for head, h1, i, … etc
  - head and body are **sibling**
  - h1 and p are **children** of body
  - head is **parent** of title
  - "this is a" is the **first children** of p
  - head is the **firstChild** of html

JS: Document Object Model (DOM)

- The DOM says that:
  - The entire document is a document node
  - Every HTML tag is an element node
  - The texts contained in the HTML elements are text nodes
  - Every HTML attribute is an attribute node
  - Comments are comment nodes

JS: Document Object Model (DOM)

Activity 13:1

- Basic Node Properties (DOM)
  - nodeName is the name of the node (not the ID).
    - The name is the tag name for HTML tag nodes,
    - #document for the document node, and
    - #text for text nodes.
  - nodeType is a number describing the node's type:
    - 1 for HTML tags,
    - 3 for text nodes, and
    - 9 for the document.
  - nodeValue is the text contained within a text node.
  - id is the value of the ID attribute for the node.

JS: Document Object Model (DOM)

Activity 13:2

- Relationship Properties
  - firstChild is the first child node for the current node.
  - lastChild is the last child object for the current node.
  - childNodes is an array of all the child nodes under a node.
  - previousSibling is the sibling before the current node.
  - nextSibling is the sibling after the current node.
  - parentNode is the object that contains the current node.

JS: Document Object Model (DOM)
Activity 13:3 – node editing

- Node Methods
  - appendChild(node) adds a new child node to the node after all its existing children.
  - insertBefore(node, oldnode) inserts a new node before the specified existing child node.
  - replaceChild(node, oldnode) replaces the specified old child node with a new node.
  - removeChild(node) removes an existing child node.
  - hasChildNodes() returns a Boolean value of true if the node has one or more children, or false if it has none.
  - cloneNode() returns a copy of the current node.

JS: Document Object Model (DOM) innerHTML

- innerHTML sets or gets all of the markup and content within a given element.
- var *markup* = element.innerHTML;
  element.innerHTML = *markup*;
- Creating Some text
  - Var p = document.createElement("p");
  - p.innerHTML = "Some text"; //able to include html tag
  - p.innerHTML = "Some text";

  - Also the same (drawback – cannot put html tag)
    - Var p = document.createElement("p");
    - var textNode = document.createTextNode(" Some text");
    - p.appendChild(textNode);

JS: Document Object Model (DOM) innerHTML – notes

- Not actually a part of the W3C DOM specification,
- However can provides a simple way to completely replace the contents of an element or textNode and also table cell content
  - document.body.innerHTML = "";
  - // Replaces body content with an empty string.
- Supported both IE6 and Mozilla

JS: Document Object Model (DOM)

- DOM Methods and Properties
  - document.getElementById(ID) returns the element with the specified ID attribute.
  - document.getElementsByTagName(tag) returns an array of the elements with the specified tag name. You can use the asterisk (*) as a wildcard to return an array containing all of the nodes in the document.
  - document.createElement(tag) creates a new element with the specified tag name.
  - document.createTextNode(text) creates a new text node containing the specified text.
  - document.documentElement is an object that represents the document itself, and can be used to find information about the document.

JS: DOM – Creating HTML doc.

Activity 13:4

- Get the body reference
  - var body = document.getElementsByTagName("BODY").item(0);
- Create the tag and complete the tag properties, children… etc
  - var pElement = document.createElement("p");
  - pElement.innerHTML = "Hello Class, Welcome to DOM world!"
  - Or
    - var textNode = document.createTextNode("Hello Class, Welcome to DOM world!");
    - pElement.appendChild(textNode);
- Add the tag to body
  - body.appendChild(pElement);

JS: DOM – Adding or modifying HTML element properties

- Methods
  - getAttribute("attribute_name")
    - Same method:
      - getAttributeNode("attribute_name")
    - Retrieve the node representation of the named attribute from the current node.
  - setAttribute("attribute_name", "attribute_value")
    - Adds a new attribute or changes the value of an existing attribute on the specified element.
  - hasAttribute(("attribute_name")
    - Return boolean
  - removeAttribute("attribute_name")

JS: DOM – Adding or modifying HTML element properties

Activity 13:5

- Create the element
  - var linkElement = document.createElement("a");
- Set the element content
  - linkElement.innerHTML = "Click me to go to google!";
- Set element property
  - linkElement.setAttribute("href", "http://www.google.com");

- To change attribute value:
  - Get the element
  - Use setAttribute with a new value

JS: DOM – Creating HTML table

Activity 13:6

- Create table
  - var tbl = document.createElement("table");
- Create table body
  - var tblBody = document.createElement("tbody");
- Create row(tr) and cells(td)
  - First create row: var row = document.createElement("tr");
  - Then create col: var cell = document.createElement("td");
    - Create cell content:
      - var cellText = document.createTextNode("cell content");
      - cell.appendChild(cellText);
  - Add cell (td) to row:
    - row.appendChild(cell);
- Add row(tr) to table body(tbody)
  - tblBody.appendChild(row);
- Add table body(tbody) to table(table)
  - tbl.appendChild(tblBody);

JS: DOM – Creating HTML table – cont.

- Creating cell content using innerHTML
  - var cell = document.createElement("td");
  - cell.innerHTML = "cell content";

Online Learning

UTM
UNIVERSITI TEKNOLOGI MALAYSIA

THANK YOU