DOM with JavaScript

Mohamed Mukthar Ahmed

DOM with JavaScript

Outline

- Doccument Object Model (DOM)
- The Node Interface
- The Document & Element Interfaces
- The FAQs Application
- Usability & Accessibility
- Cancel Default Action of an Event
- Preload Images
- Image Swap Application
- Using Timers
- FAQs Application with Timers

Mohamed Mukthar Ahmed

Document Object Model (DOM)

- As the browser loads an HTML page, it builds a **DOM**
- The DOM is a hierarchical collection of **nodes** in the web browser's memory that represents the current page
- JavaScript can modify the web page in the browser by modifying the DOM. Whenever the DOM is changed the browser displays the result
- To modify the DOM, we can use the properties and methods that are defined by the DOM Core specification

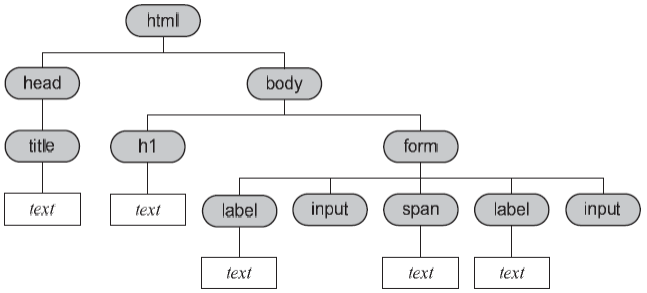
Document Object Model (DOM)

The code for a web page

```
<!DOCTYPE html>
<html>
<head>
    <title>Join Email List</title>
</head>
<body>
    <h1>Please join our email list</h1>
    <form id="email_form" name="email_form"
          action="join.html" method="get">
        <label for="email_address">Email Address:</label>
        <input type="text" id="email_address">
        <span id="email_address_error">*</span><br>
        <label> </label>
        <input type="button" id="join_list" value="Join our List">
    </form>
</body>
</html>
```

Document Object Model (DOM)

The DOM for the web page



The DOM nodes commonly use

Type	Description
Document	Root node of the DOM. It can only have one Element node as a child node.
Element	An element in the web page. It can have Element, Text, and Comment nodes as child nodes.
Attr	An attribute of an element. Although it is attached to an Element node, it isn't considered a child node. It can have a Text node as a child node.
Text	The text for an element or attribute. It can't have a child node.

Mohamed Mukthar Ahmed

Document Object Model (DOM)

- DOM can also contain **attribute nodes**, and each attribute node can have a **text node**
- If the HTML includes comments, the DOM will include the **comment nodes**
- The properties and methods for working with DOM nodes are defined by a specification called an **interface**
- As we work with the interface we come across terms like **parent**, **child**, **sibling** and **descendant**. Used just as they are in the family tree

Mohamed Mukthar Ahmed

The Node Interface

- The six properties that are defined by the Node interface for working with the nodes are:

Property	Description
`nodeValue`	For a Text, Comment, or Attribute node, this property returns the text that's stored in the node. Otherwise, it returns a null value.
`parentNode`	Returns the parent node of the current node if one exists. Otherwise, this property returns a null value.
`childNodes`	Returns an array of Node objects representing the child nodes of the current node.
`firstChild`	Returns a Node object for the first child node. If this node doesn't have child nodes, this property returns a null value.
`lastChild`	Returns a Node object for the last child node. If this node doesn't have child nodes, this property returns a null value.
`nextElementSibling`	Returns a Node object for the next sibling. If this node doesn't have a sibling element that follows it, this property returns a null value.

Mohamed Mukthar Ahmed

The Node Interface

HTML that contains element and text nodes

```
<body>
    <h1>Please join our email list</h1>
    <form id="email_form" name="email_form" action="join.html" method="get">
        <label for="email_address">Email Address:</label>
        <input type="text" id="email_address">
        <span id="email_error">*</span><br>
        <label> </label>
        <input type="button" id="join_list" value="Join our List">
    </form>
</body>
```

How to get the text of an HTML element with "email_error" as its id

```
var errorText = $("email_error").firstChild.nodeValue;
```

How to set the text of an HTML element with "email_error" as its id

```
$("email_error").firstChild.nodeValue = "Entry is invalid.";
```

How to set the text for the span tag to an empty string without using its id

```
$("email_address").nextElementSibling.firstChild.nodeValue = "";
```

Mohamed Mukthar Ahmed

The Document & Element Interfaces

- The methods of Document and Element interfaces returns an array of elements
- The methods of the Element interface also allows us to work with attributes

Common methods of the Document and Element interfaces

Method	Description
`getElementsByTagName(tagName)`	Returns an array of all Element objects descended from the document or element that have a tag that matches the specified tag.
`getElementsByName(name)`	Returns an array of all Element objects descended from the document or element that have a name attribute that matches the specified name.
`getElementsByClassName(classNames)`	Returns an array of all Element objects descended from the document or element that have a class attribute with a name or names that match the parameter. The classNames parameter can be a single name or a space-separated list of class names.

The Document & Element Interfaces

- The methods of Document and Element interfaces returns an array of elements
- The methods of the Element interface also allows us to work with attributes

Common methods of the Element interface

Method	Description
`hasAttribute(name)`	Returns true if the Element has the attribute specified in name.
`getAttribute(name)`	Returns the value of the attribute specified in name or an empty string if an attribute of that name isn't set.
`setAttribute(name, value)`	Sets the attribute specified in name to the specified value. If the attribute doesn't already exist, it creates the attribute too.
`removeAttribute(name)`	Removes the attribute specified in name.

Mohamed Mukthar Ahmed

The Document & Element Interfaces

How to create an array of all <a> tags in a document

```
var links = document.getElementsByTagName("a");
```

How to create an array of all li tags within a ul element (image_list)

```
var list = document.getElementById("image_list");
var items = list.getElementsByTagName("li");
```

How to test for and get an attribute

```
var list = document.getElementById("image_list");
if ( list.hasAttribute("class") ) {
    var classAttribute = list.getAttribute("class"));
}
```

How to set an attribute

```
var image = document.getElementById("div");
image.setAttribute("class", "open");
```

How to remove an attribute

```
var list = document.getElementById("image_list");
list.removeAttribute("class");
```

Mohamed Mukthar Ahmed

The FAQs Application

- If the user clicks on a heading with a plus sign, the text below it is displayed and the plus sign is changed to minus
- If the user clicks on a heading with a minus sign, the text below it is hidden and the minus sign is changed to plus

Mohamed Mukthar Ahmed

The FAQs Application - HTML

```
<body>
    <section id="faqs">
        <h1>jQuery FAQs</h1>
        <h2 class="plus">What is jQuery?</h2>
        <div class="closed">
            <p>jQuery is a library of the JavaScript functions that you're most likely
               to need as you develop web sites.
            </p>
        </div>
        <h2 class="plus">Why is jQuery becoming so popular?</h2>
        <div class="closed">
            <p>Three reasons:</p>
            <ul>
                <li>It's free.</li>
                <li>It lets you get more done in less time.</li>
                <li>All of its functions are cross-browser compatible.</li>
            </ul>
        </div>
        <h2 class="plus">Which is harder to learn: jQuery or JavaScript?</h2>
        <div class="closed">
            <p>For most functions, jQuery is significantly easier to learn
                and use than JavaScript. But remember that jQuery is JavaScript.
            </p>
```


The FAQs Application - CSS

```
h2 {
    font-size: 120%;
    padding: .25em 0 .25em 25px;
    cursor: pointer;
}
h2.plus {
    background: url(images/plus.png) no-repeat left center;
}
h2.minus {
    background: url(images/minus.png) no-repeat left center;
}
div.closed {
    display: none;
}
div.open {
    display: block;
}
```

The FAQs Application - JavaScript

```
window.onload = function () {
    var faqs = $("faqs");
    var h2Elements = faqs.getElementsByTagName("h2");

    var h2Node;
    for (var i = 0; i < h2Elements.length; i++ ) {
        h2Node = h2Elements[i];
        // Attach event handler
        h2Node.onclick = function () {
            var h2 = this;          // h2 is the current h2Node object
            if (h2.getAttribute("class") == "plus") {
                h2.setAttribute("class", "minus");
            }
            else {
                h2.setAttribute("class", "plus");
            }
            if (h2.nextElementSibling.getAttribute("class") == "closed") {
                h2.nextElementSibling.setAttribute("class", "open");
            }
            else {
                h2.nextElementSibling.setAttribute("class", "closed");
            }
        }
    }
```


The FAQs Application - HTML

```
<body>
    <section id="faqs">
        <h1>jQuery FAQs</h1>
        <h2>What is jQuery?</h2>
        <div>
            <p>jQuery is a library of the JavaScript functions that you're most likely
               to need as you develop web sites.
            </p>
        </div>
        <h2>Why is jQuery becoming so popular?</h2>
        <div>
            <p>Three reasons:</p>
            <ul>
                <li>It's free.</li>
                <li>It lets you get more done in less time.</li>
                <li>All of its functions are cross-browser compatible.</li>
            </ul>
        </div>
        <h2>Which is harder to learn: jQuery or JavaScript?</h2>
        <div>
            <p>For most functions, jQuery is significantly easier to learn
                and use than JavaScript. But remember that jQuery is JavaScript.
            </p>
```

The FAQs Application - CSS

```css
h2 {
    font-size: 120%;
    padding: .25em 0 .25em 25px;
    cursor: pointer;
    background: url(images/plus.png) no-repeat left center;
}
h2.minus {
    background: url(images/minus.png) no-repeat left center;
}
div {
    display: none;
}
div.open {
    display: block;
}
```


The FAQs Application - JavaScript

```javascript
window.onload = function () {
    var faqs = $("faqs");
    var h2Elements = faqs.getElementsByTagName("h2");

    var h2Node;
    for (var i = 0; i < h2Elements.length; i++ ) {
        h2Node = h2Elements[i];
        // Attach event handler
        h2Node.onclick = function () {
            var h2 = this;          // h2 is the current headingNode object
            if (h2.hasAttribute("class")) {
                h2.removeAttribute("class");
            }
            else {
                h2.setAttribute("class", "minus");
            }
            if (h2.nextElementSibling.hasAttribute("class")) {
                h2.nextElementSibling.removeAttribute("class");
            }
            else {
                h2.nextElementSibling.setAttribute("class", "open");
            }
        }
    }
```

Usability & Accessibility

- **Usability** refers to how easy it is use a web-site, and usability is critical requirement for an effective web-site
- Guidelines
  - Underlined text is always a link.
  - A small symbol in front of a text phrase is clickable.
  - Images that are close to short text phrases are clickable.
  - Buttons should look like buttons and should always be clickable.

Usability & Accessibility

- **Accessibility** refers to the qualities that make a web page access to users, especially disabled users
- Guidelines
  - For the visually-impaired, all of the essential information should be presented in text that's easy to read because some users may not be able to read the text that's in images.
  - For the motor-impaired, all of the essential content and features should be accessible with the keyboard because some users may not be able to use a mouse.
- When developing a JavaScript application make sure, it also meets the best standards of usability and accessibility

Cancel Default Action of an Event

- In some applications we need to cancel the default action of an event
- All DOM-compliant browsers pass an ***event object*** to the event handler. The **preventDefault** method of the event object when called/invoked prevents the default action of the event from occurring

Common default actions for the click event

Tag	Default action for the click event
a	Load the page or go to the placeholder in the href attribute.
input	Submit the form if the type attribute is set to submit.
input	Reset the form if the type attribute is set to reset.
button	Submit the form if the type attribute is set to submit.
button	Reset the form if the type attribute is set to reset.

```
var eventHandler = function (evt) {
    evt.preventDefault();
}
```


Preload Images

- When an application preloads images, it loads all the images need by the page and stores them in the browsers cache
- If the images are not preloaded, they are retrieved from the server as needed causing a noticeable delay
- To preload an image, create a **Image** object, then set the **src** attribute of the created image object.

How to create an Image object

```
var image = new Image();
```

How to preload an image in an Image object

```
image.src = "image_name.jpg";
```

Image Swap Application

- In this application, the main image is swapped whenever the user clicks on one of the small (thumbnail) images

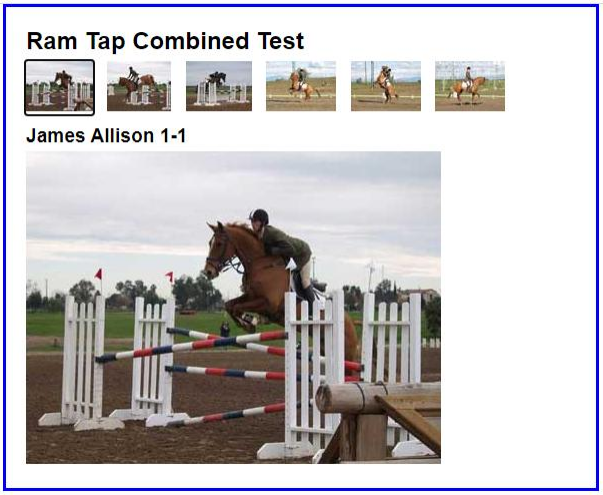


Image Swap Application - HTML

```
<body>
    <section>
        <h1>Ram Tap Combined Test</h1>
        <ul id="image_list">
            <li><a href="images/h1.jpg" title="James Allison: 1-1" id="first_link">
                <img src="thumbnails/t1.jpg" alt=""></a></li>
            <li><a href="images/h2.jpg" title="James Allison: 1-2">
                <img src="thumbnails/t2.jpg" alt=""></a></li>
            <li><a href="images/h3.jpg" title="James Allison: 1-3">
                <img src="thumbnails/t3.jpg" alt=""></a></li>
            <li><a href="images/h4.jpg" title="James Allison: 1-4">
                <img src="thumbnails/t4.jpg" alt=""></a></li>
            <li><a href="images/h5.jpg" title="James Allison: 1-5">
                <img src="thumbnails/t5.jpg" alt=""></a></li>
            <li><a href="images/h6.jpg" title="James Allison: 1-6">
                <img src="thumbnails/t6.jpg" alt=""></a></li>
        </ul>
        <h2 id="caption">James Allison 1-1</h2>
        <p><img src="images/h1.jpg" alt="" id="image"></p>
    </section>
</body>
```

Image Swap Application - CSS

```
5  body {
6      font-family: Arial, Helvetica, sans-serif;
7      margin: 0 auto;
8      padding: 20px;
9      width: 550px;
10     border: 3px solid blue;
11 }
12 h1, h2, ul, p {
13     margin: 0;
14     padding: 0;
15 }
16 h1 {
17     padding-bottom: .25em;
18 }
19 h2 {
20     font-size: 120%;
21     padding: .5em 0 .25em;
22 }
23 li {
24     padding-right: 10px;
25     display: inline;
26 }
```

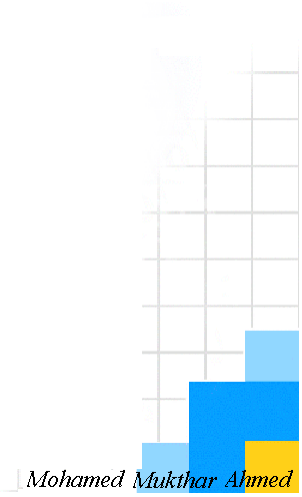
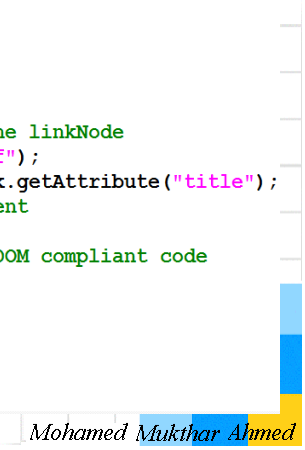


Image Swap Application - JavaScript

```
4  window.onload = function () {
5      var listNode = $("image_list");
6      var captionNode = $("caption");
7      var imageNode = $("image");
8
9      var imageLinks = listNode.getElementsByTagName("a");
10     // Process image links
11     var i, linkNode, image;
12     for ( i = 0; i < imageLinks.length; i++ ) {
13         linkNode = imageLinks[i];
14         // Attach event handler
15         linkNode.onclick = function (evt) {
16             var link = this;          // link is the linkNode
17             imageNode.src = link.getAttribute("href");
18             captionNode.firstChild.nodeValue = link.getAttribute("title");
19             // Cancel the default action of the event
20             if ( evt.preventDefault ) {
21                 evt.preventDefault();          // DOM compliant code
22             }
23         }
24         // Preload image
25         image = new Image();
26         image.src = linkNode.getAttribute("href");
27     }
```



Using Timers

- **Timers** let us execute function after a specified period of time
- Two types
  - One-Time Timer
  - Interval Timer
- The **setTimeout** method create a timer that calls a function (once) after the specified delay in millisecond. Moreover it returns a reference to the new timer
- The **clearTimeout** method cancels the timer that was created by the setTimeout method

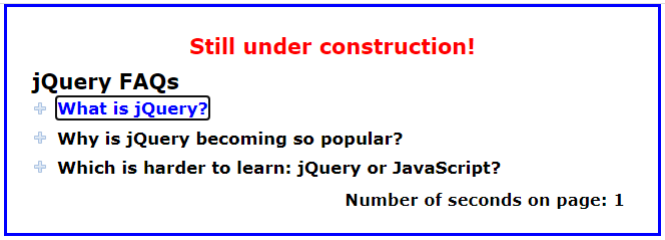
```
setTimeout( function, delayTime )       // creates a timer
clearTimeout ( timer )                  // cancels a timer
```


Using Timers

- Two types
  - One-Time Timer
  - Interval Timer
- The **setInterval** method create a timer that calls a function each time after the specified interval in millisecond. Moreover it returns a reference to the new timer
- The **clearInterval** method cancels the timer that was created by the setInterval method

```
setInterval( function, intervalTime )     // creates a timer
clearInterval ( timer )                   // cancels a timer
```

FAQs Application - Timers

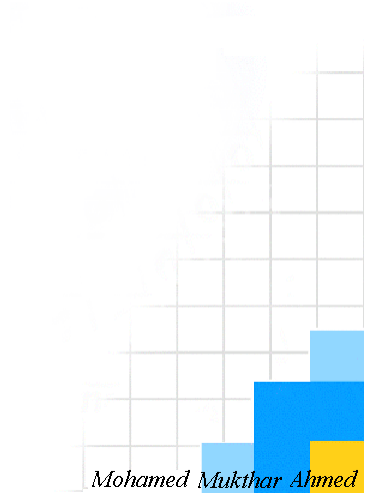


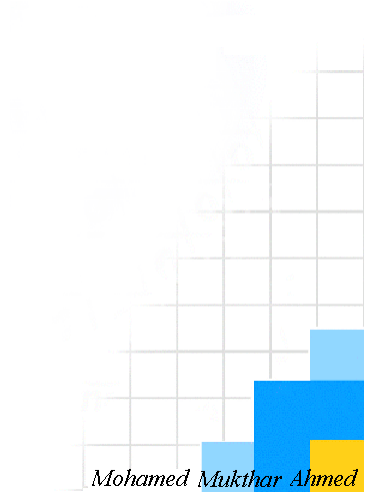

```
<section id="faqs">
    <h1 id="startup_message">Still under construction!</h1>
    <h1>jQuery FAQs</h1>
    :
    <h3>Number of seconds on page: <span id="counter">0</span></h3>
</section>
```


FAQs Application - Timers

```
#startup_message {
    padding: .5em;
    color: red;
    text-align: center;
}
#startup_message.closed {
    display: none;
}
h3 {
    padding: .5em;
    text-align: right;
```

```
// The function for the one-time timer
var hideMessage = function() {
    $("startup_message").setAttribute("class", "closed");
    clearTimeout(timer1);
}
// The function for the interval timer
var counter = 0;
var updateCounter = function() {
    counter++;
    document.getElementById("counter").firstChild.nodeValue = counter;
}
var timer1, timer2;
timer1 = setTimeout(hideMessage, 5000);          // calling one-time fun
timer2 = setInterval( updateCounter, 1000 );     // calling the interval
```


Mohamed Mukthar Ahmed


Mohamed Mukthar Ahmed