الجامعة الإسلامية العالمية ماليزيا
INTERNATIONAL ISLAMIC UNIVERSITY MALAYSIA
يونيورسيتي إسلام انتارابغسا ملايسيا

INFO 2302 Web Technologies

DOM and Events

Marini Othman
Kulliyyah of Information and Communication Techhnology
International Islamic University Malaysia
omarini@iium.edu.my

Contents

- Document Object Model (DOM)
- Events

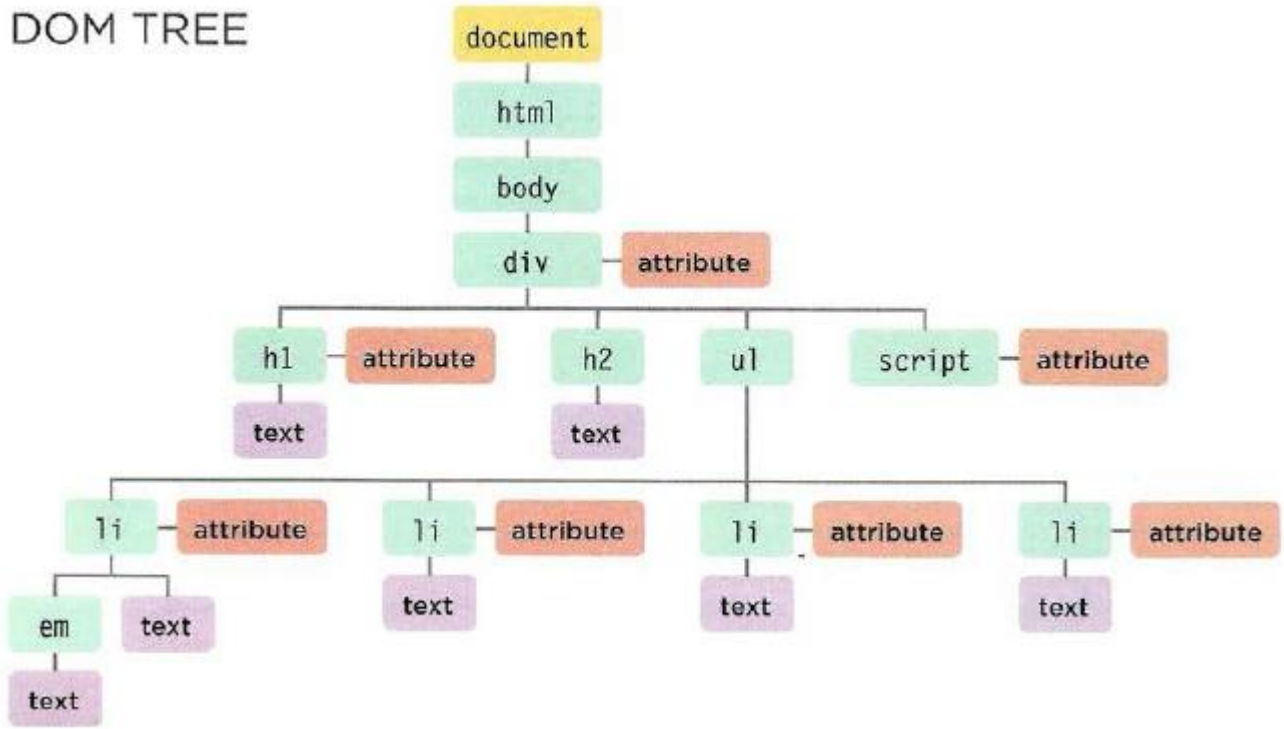
Document Object Model

- Describes how:-
  - browsers should create a model of an HTML using **DOM tree**
  - JavaScript can access and update the contents of a web page while it is in the browser window

DOM Tree

BODY OF HTML PAGE

```
<html>
  <body>
    <div id="page">
      <h1 id="header">List</h1>
      <h2>Buy groceries</h2>
      <ul>
        <li id="one" class="hot"><em>fresh</em> figs</li>
        <li id="two" class="hot">pine nuts</li>
        <li id="three" class="hot">honey</li>
        <li id="four">balsamic vinegar</li>
      </ul>
      <script src="js/list.js"></script>
    </div>
  </body>
</html>
```

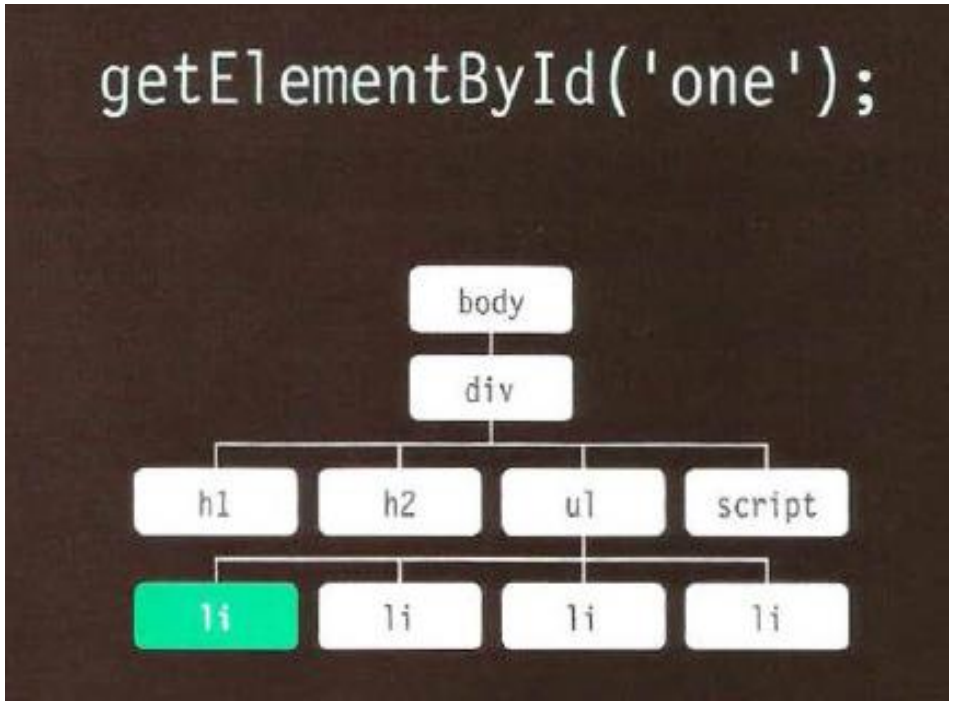


Working with DOM Tree

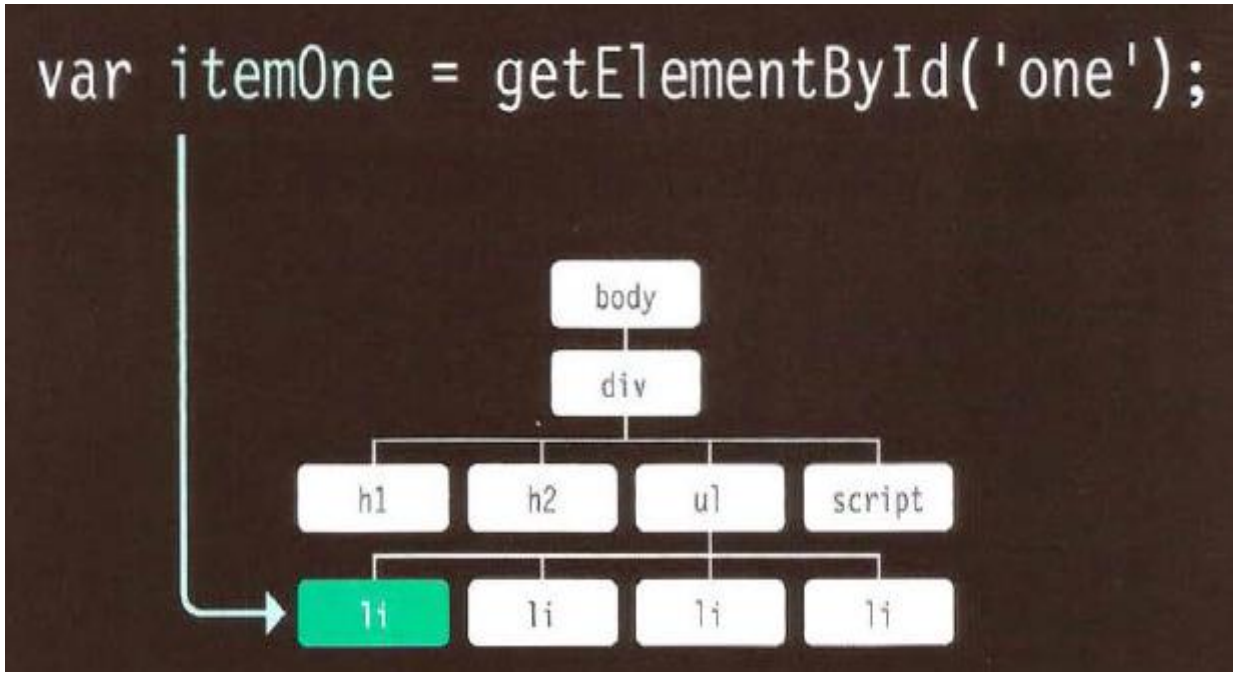
- Accessing and updating the DOM tree involves two steps:
  - 1: Locate the node that represents the element you want to work with.
  - 2: Use its text content, child elements, and attributes.

DOM queries

DOQ queries:
Methods that find elements in DOM TREE



Storing element in variables



Selecting elements using ID attribute

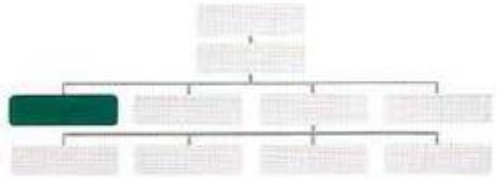
```
<h1 id="header">List King</h1>
<h2>Buy groceries</h2>
<ul>
  <li id="one" class="hot"><em>fresh</em>
    figs</li>
  <li id="two" class="hot">pine nuts</li>
  <li id="three" class="hot">honey</li>
  <li id="four">balsamic vinegar</li>
</ul>
```

```
// Select the element and store it in a variable.
var el = document.getElementById('one');

// Change the value of the class attribute.
el.className = 'cool';
```

Nodelists

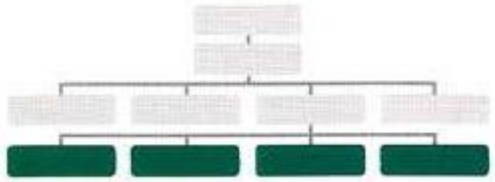
- Nodelist: When a DOM method returns more than 1 element



`getElementsByTagName('h1')`

Even though this query only returns one element, the method still returns a NodeList because of the potential for returning more than one element.

INDEX NUMBER & ELEMENT
0 <h1>



`getElementsByTagName('li')`

This method returns four elements, one for each of the <li> elements on the page. They appear in the same order as they do in the HTML page.

INDEX NUMBER & ELEMENT
0 <li id="one" class="hot">
1 <li id="two" class="hot">
2 <li id="three" class="hot">
3 <li id="four">

Selecting elements using **class** attribute

```
<!DOCTYPE html>
<html>
 <head>
   <title>Chapter 5: Get Elements By Class Name</title>
   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
   <link rel="stylesheet" href="css/c05.css">
  </head>
<body>
  <div id="page">
  <h1 id="header">List</h1>
 <h2>Buy groceries</h2>
      <ul>      <li id="one" class="hot"><em>fresh</em>
figs</li>
                <li id="two" class="hot">pine nuts</li>
               <li id="three" class="hot">honey</li>
               <li id="four">balsamic vinegar</li>
      </ul>
</div>
<script src="js/get-elements-by-class-name.js"></script>
</body></html>
```

```
var elements = document.getElementsByClassName('hot');
// Find hot items

if (elements.length > 2) {                    // If 3 or more are found

 var el = elements[2];          // Select the third one from the NodeList
 el.className = 'cool';         // Change the value of its class attribute

}
```

Finding items in Nodelist using items

Selecting element using **tag** name

```
<!DOCTYPE html>
<html>
   <head>    <title> Get Elements By Tag Name</title>
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <link rel="stylesheet" href="css/c05.css">  </head>
 <body>    <div id="page">
             <h1 id="header">List King</h1>
             <h2>Buy groceries</h2>
                <ul>
                   <li id="one" class="hot"><em>fresh</em> figs</li>
                   <li id="two" class="hot">pine nuts</li>
                   <li id="three" class="hot">honey</li>
                   <li id="four">balsamic vinegar</li>
               </ul>
             </div>
        <script src="js/get-elements-by-tag-name.js"></script>
</body></html>
```

```
var elements = document.getElementsByTagName('li');
// Find <li> elements

if (elements.length > 0) {  // If 1 or more are found

  var el = elements[0];     // Select the first one using array syntax
  el.className = 'cool';    // Change the value of the class attribute

}
```

Selecting elements using CSS selectors

```
<!DOCTYPE html>
<html>
    <head>    <title> Query Selector</title>
            <meta name="viewport" content="width=device-
width, initial-scale=1.0">
            <link rel="stylesheet" href="css/c05.css">
    </head>
<body>
     <div id="page">
          <h1 id="header">List King</h1>
           <h2>Buy groceries</h2>
           <ul>
              <li id="one" class="hot"><em>fresh</em> figs</li>
              <li id="two" class="hot">pine nuts</li>
              <li id="three" class="hot">honey</li>
              <li id="four">balsamic vinegar</li>
           </ul>
         </div>
<script src="js/query-selector.js"></script>
</body>
</html>
```

```
// querySelector only returns the first match.
var el = document.querySelector('li.hot');
el.className = 'cool';

// querySelectorAll returns a NodeList.
// The third li element is then selected and changed.
var els = document.querySelectorAll('li.hot');
els[1].className = 'cool';
```

Looping through a Nodelist

```
<!DOCTYPE html>
<html>
<head>
	<title>Node List</title>
	<meta name="viewport" content="width=device-width, initial-
scale=1.0">
	<link rel="stylesheet" href="css/c05.css">
</head>
<body>
	<div id="page">
	<h1 id="header">List</h1>
	<h2>Buy groceries</h2>
	 <ul><li id="one" class="hot"><em>fresh</em> figs</li>
		 <li id="two" class="hot">pine nuts</li>
		<li id="three" class="hot">honey</li>
		<li id="four">balsamic vinegar</li>
	</ul>
	 </div>
    <script src="js/node-list.js"></script>
</body></html>
```

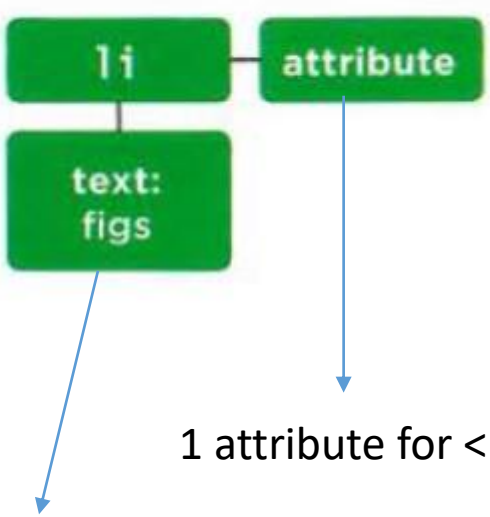
```
var hotItems = document.querySelectorAll('li.hot');
// Store NodeList in array
if (hotItems.length > 0) {	// If it contains items
   for (var i = 0; i < hotItems.length; i++) {
    // Loop through each item

hotItems[i].className = 'cool';
// Change value of class attribute  }}
```

Looping through a nodelist: play-by-play

Get/update element content

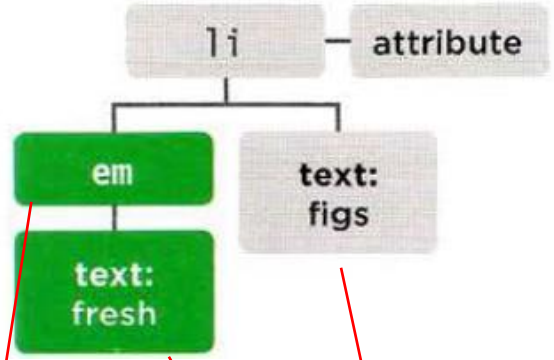
```
<li id="one">figs</li>
```



1 attribute for

1 child node for

```
<li id="one"><em>fresh</em> figs</li>
```



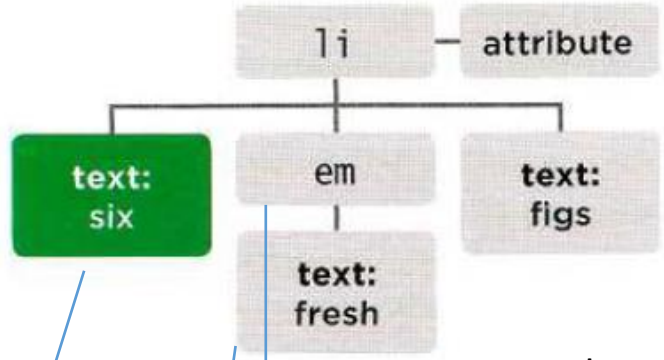
Sibling to

1st child for

Child text for

Text added before

```
<li id="one">six <em>fresh</em> figs</li>
```



Text node is 1st child for

 is sibling of text node: six

Another sibling for the first text node and

Child text node for

Access and change text node

```
<!DOCTYPE html>
 <html>
    <head>   <title> Document Object Model - Node Value</title>
            <meta name="viewport" content="width=device-width,
initial-scale=1.0">
           <link rel="stylesheet" href="css/c05.css">  </head>
<body>
      <div id="page">
              <h1 id="header">List</h1>
              <h2>Buy groceries</h2>
              <ul>       <li id="one" class="hot"><em>fresh</em> figs</li>
                         <li id="two" class="hot">pine nuts</li>
                         <li id="three" class="hot">honey</li>
                         <li id="four">balsamic vinegar</li>
             </ul>
       </div>
       <script src="js/node-value.js"></script>
</body>
</html>
```

```
var itemTwo = document.getElementById('two');
// Get second list item

var elText  = itemTwo.firstChild.nodeValue;
// Get its text content

elText = elText.replace('pine nuts', 'kale');
// Change pine nuts to kale

itemTwo.firstChild.nodeValue = elText;        /
/ Update the list item
```

Accessing text only: (textContent vs. innerText)

```
<!DOCTYPE html>
<html>
<head>    <title>Inner Text &amp;amp; Text Content</title>
        <meta name="viewport" content="width=device-width, initial-
scale=1.0">
        <link rel="stylesheet" href="css/c05.css">
        <style>    /* This page hides the <em> elements to demonstrate
difference between innerText and textContent */
            em {display: none;}
        </style>
</head>
<body> < div id="page">  <h1 id="header">List</h1>
                        <h2>Buy groceries</h2>
            <ul>
                <li id="one" class="hot"><em>fresh</em> figs</li>
                <li id="two" class="">pine nuts</li>
                <li id="three" class="hot">honey</li>
                <li id="four">balsamic vinegar</li>
            </ul>
     <div id="scriptResults"></div>    </div>
<script src="js/inner-text-and-text-content.js"></script>
</body></html>
```


```
var firstItem = document.getElementById('one');
// Find first list item

var showTextContent = firstItem.textContent;
// Get value of textContent

var showInnerText = firstItem.innerText;
// Get value of innerText

// Show the content of these two properties at the end of
//the list
var msg = '<p>textContent: ' + showTextContent + '</p>';
    msg += '<p>innerText: ' + showInnerText + '</p>';
var el = document.getElementById('scriptResults');
el.innerHTML = msg;

firstItem.textContent = 'sourdough bread';
 // Update the first list item
```

Adding or removing HTML content

```
<!DOCTYPE html>
<html>  <head>    <title>Inner HTML</title>
       <meta name="viewport" content="width=device-width, initial-
scale=1.0">
       <link rel="stylesheet" href="css/c05.css">
 </head>
 <body>
      <div id="page">
         <h1 id="header">List</h1>
        <h2>Buy groceries</h2>
         <ul><li id="one" class="hot"><em>fresh</em> figs</li>
              <li id="two" class="hot">pine nuts</li>
             <li id="three" class="hot">honey</li>
             <li id="four">balsamic vinegar</li>
         </ul>
  </div>    <script src="js/inner-html.js"></script>
  </body></html>
```

```
// Store the first list item in a variable
var firstItem = document.getElementById('one');

// Get the content of the first list item
var itemContent = firstItem.innerHTML;

// Update the content of the first list item so it is a link
firstItem.innerHTML = '<a href=\"http://example.org\">' + itemContent
+ '</a>';
```

1

CREATE THE ELEMENT

`createElement()`

You start by creating a new element node using the `createElement()` method. This element node is stored in a variable.

When the element node is created, it is not yet part of the DOM tree. It is not added to the DOM tree until step 3.

2

GIVE IT CONTENT

`createTextNode()`

`createTextNode()` creates a new text node. Again, the node is stored in a variable. It can be added to the element node using the `appendChild()` method.

This provides the content for the element, although you can skip this step if you want to attach an empty element to the DOM tree.

3

ADD IT TO THE DOM

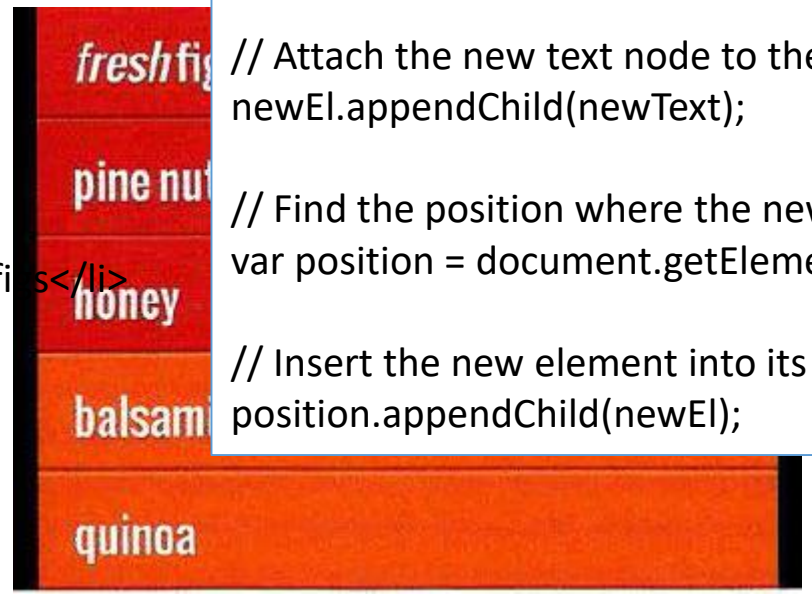
`appendChild()`

Now that you have your element (optionally with some content in a text node), you can add it to the DOM tree using the `appendChild()` method.

The `appendChild()` method allows you to specify which element you want this node added to, as a child of it.

Adding element using DOM manipulation

```
<!DOCTYPE html>
  <html>
    <head>   <title> Add Element</title>
    <link rel="stylesheet" href="css/c05.css" />
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
     </head>
   <body>   <div id="page">
      <h1 id="header">List</h1>
      <h2>Buy groceries</h2>
      <ul id="todo">
         <li id="one" class="hot"><em>fresh</em> fi s</li>
         <li id="two" class="hot">pine nuts</li>
         <li id="three" class="hot">honey</li>
         <li id="four">balsamic vinegar</li>
      </ul>
    </div>
<script src="js/add-element.js"></script>
</body>
</html>
```




```
// Create a new element and store it in a variable.
var newEl = document.createElement('li');

// Create a text node and store it in a variable.
var newText = document.createTextNode('quinoa');

// Attach the new text node to the new element.
newEl.appendChild(newText);

// Find the position where the new element should be added.
var position = document.getElementsByTagName('ul')[0];

// Insert the new element into its position.
position.appendChild(newEl);
```

Removing element using DOM tree

```
<!DOCTYPE html>
    <html>  <head>    <title> Remove Element</title>
     <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <link rel="stylesheet" href="css/c05.css">
     </head>
     <body>
        <div id="page">
        <h1 id="header">List</h1>
        <h2>Buy groceries</h2>
        <ul>
          <li id="one" class="hot"><em>fresh</em> figs</li>
         <li id="two" class="hot">pine nuts</li>
         <li id="three" class="hot">honey</li>
         <li id="four">balsamic vinegar</li>
        </ul>
      </div>
<script src="js/remove-element.js"></script>
 </body></html>
```

```
// Store the element to be removed in a variable.
var removeEl = document.getElementsByTagName('li')[3];

// Find the element which contains the element to be removed.
var containerEl = document.getElementsByTagName('ul')[0];

// Remove the element.container
El.removeChild(removeEl);
```

Comparing techniques

document.write()

+quick and easy way to add content to a page

-only works when page initially loads.
-Using it after the page loads:
  -overwrites the whole page
  - not adding content
  -create a new page
-rarely used technique

element.innerHTML()

+can add more markup with less code than DOM manipulation.
+faster than DOM manipulation

-should not be used with contents coming from user (eg. Username—security issues).
-event handlers may no longer works.

DOM manipulation

+suited to change one element from DOM fragment when there are many siblings.
+does not effect event handlers.

-if making a lot of changes, it is slower than innerHTML.
-need to write more code to achieve same thing with innerHTML.

Cross site scripting attacks (XSS)

XSS: Attacker plans malicious codes.

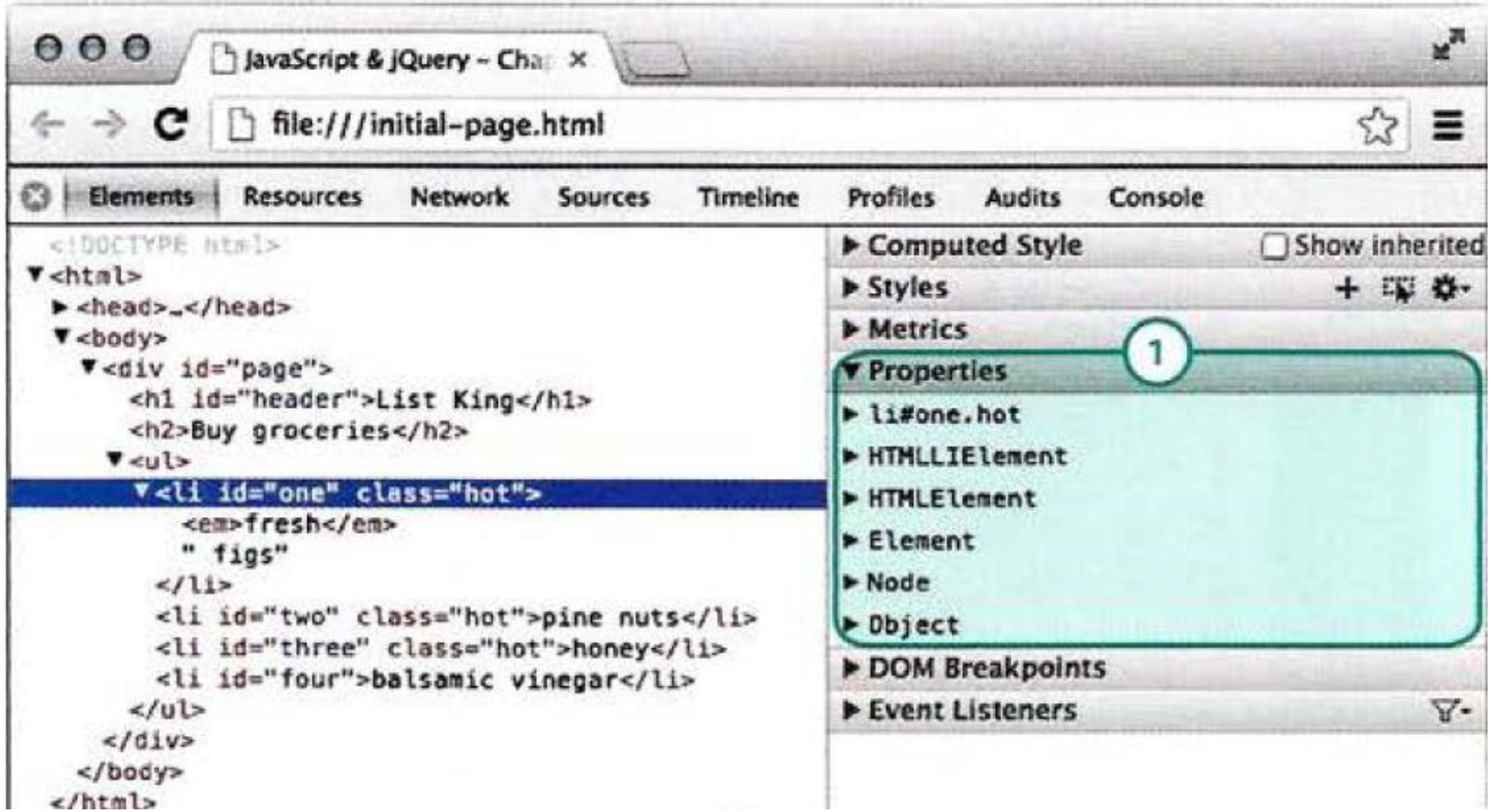
XSS gives access to information in:

- DOM (including data)
- Cookies
- Session tokens (info that tells when user logs in)

This example stores cookie data in a variable, which could then be sent to a third-party server:

```
<script>var adr= 'http : //example.com/xss .php?cookie=' + escape(document . cookie);</script>
```

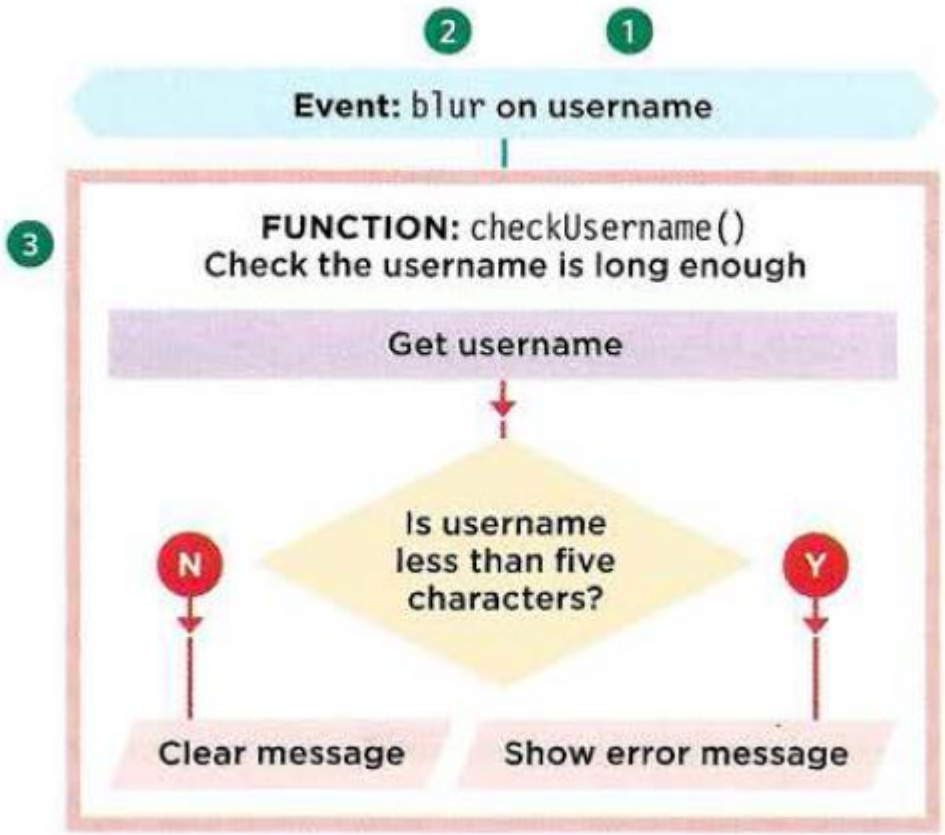
Examining DOM in Chrome



More tools > Developer tools

Events

How events trigger Javascript codes



1

SELECT ELEMENT

The element that users are interacting with is the text input where they enter the username.

2

SPECIFY EVENT

When users move out of the text input, it loses focus, and the `blur` event fires on this element.

3

CALL CODE

When the `blur` event fires on the username input, it will trigger a function called `checkUsername()`. This function checks if the username is less than 5 characters.

Event handlers

- HTML Event Handlers Attribute
- Traditional DOM Event Handlers
- DOM Level 2 Event Listeners

HTML Event Handlers Attribute

TML

c06/event-attributes.html

```
<form method="post" action="http://www.example.org/register">
  <label for="username">Create a username: </label>
  <input type="text" id="username" onblur="checkUsername()" />
  <div id="feedback"></div>

  <label for="password">Create a password: </label>
  <input type="password" id="password" />

  <input type="submit" value="Sign up!" />
</form>
...
<script type="text/javascript" src="js/event-attributes.js"></script>
```

Bad practice!!
No longer used because it is better to separate the JavaScript from the HTML.

VASCRIPT

c06/js/event-attributes.js

```
function checkUsername() {                                  // Declare function
  var elMsg = document.getElementById('feedback');          // Get feedback element
  var elUsername = document.getElementById('username');// Get username input
  if (elUsername.value.length < 5) {                        // If username too short
    elMsg.textContent = 'Username must be 5 characters or more'; // Set msg
  } else {                                                   // Otherwise
    elMsg.textContent = '';                                  // Clear message
  }
}
```

Traditional DOM Event Handlers

JAVASCRIPT

c06/js/event-handler.js

```
function checkUsername() {                                    // Declare function
  var elMsg = document.getElementById('feedback');            // Get feedback element
  if (this.value.length < 5) {                                // If username too short
    elMsg.textContent = 'Username must be 5 characters or more';  // Set msg
  } else {                                                    // Otherwise
    elMsg.textContent = '';                                   // Clear message
  }
}

var elUsername = document.getElementById('username');  // Get username input
elUsername.onblur = checkUsername;  // When it loses focus call checkuserName()
```

1. If you use a named function when the event fires on your chosen DOM node, write that function first. (You could also use an anonymous function.)

2. The DOM element node is stored in a variable. Here the text input (whose `id` attribute has a value of `username`) is placed into a variable called `elUsername`.

3. On the last line of the code sample above, the event handler `elUsername.onblur` indicates that the code is waiting for the `blur` event to fire on the element stored in the variable called `elUsername`.

DOM Level 2 Event Listeners

JAVASCRIPT c06/js/event-listener.js

```
function checkUsername() {                                    // Declare function
  var elMsg = document.getElementById('feedback');            // Get feedback element
  if (this.value.length < 5) {                                // If username too short
    elMsg.textContent = 'Username must be 5 characters or more'; // Set msg
  } else {                                                    // Otherwise
    elMsg.textContent = '';                                   // Clear msg
  }
}

var elUsername = document.getElementById('username');  // Get username input
// When it loses focus call checkUsername()
elUsername.addEventListener('blur', checkUsername, false);
```

① ② (i) (ii) (iii)

Using parameters with event listeners

JAVASCRIPT

c06/js/event-listener-with-parameters.js

```
var elUsername = document.getElementById('username');  // Get username input
var elMsg = document.getElementById('feedback');       // Get feedback element

function checkUsername(minLength) {                    // Declare function
  if (elUsername.value.length < minLength) {           // If username too short
   // Set the error message
   elMsg.textContent = 'Username must be ' + minLength + ' characters or more';
  } else {                                             // Otherwise
   elMsg.innerHTML = '';                               // Clear msg
  }
}

elUsername.addEventListener('blur', function() {       // When it loses focus
  checkUsername(5);                                     // Pass arguments here
}, false);
```

The Event Object

Event Listeners

Event listener without parameter

```
function checkUsername(e) {
③ var target = e.target; // get target of event
}

var el = document.getElementById('username');
el.addEventListener('blur', checkUsername, false);
```

Event listener with parameters

```
function checkUsername(e, minLength) {
④ var target = e.target; // get target of event
}

var el = document.getElementById('username');
el.addEventListener('blur', function(e){ ①
  checkUsername(e, 5);
}, false);
```

Using Event Listeners with Event Objects

```
function checkLength(e, minLength) {          // Declare function
  var el, elMsg;                              // Declare variables
  if (!e) {                                   // If event object doesn't exist
    e = window.event;                         // Use IE fallback
  }
  el = e.target || e.srcElement;              // Get target of event
  elMsg = el.nextSibling;                     // Get its next sibling

  if (el.value.length < minLength) {          // If length is too short set msg
    elMsg.innerHTML = 'Username must be ' + minLength + ' characters or more';
  } else {                                    // Otherwise
    elMsg.innerHTML = '';                     // Clear message
  }
}
```

```
var elUsername = document.getElementById('username');// Get username input
if (elUsername.addEventListener) {                  // If event listener supported
  elUsername.addEventListener('blur', function(e) {  // On blur event
    checkUsername(e, 5);                            // Call checkUsername()
  }, false);                                        // Capture in bubble phase
} else {                                            // Otherwise
  elUsername.attachEvent('onblur', function(e){     // IE fallback onblur
    checkUsername(e, 5);                            // Call checkUsername()
  });
}
```

User interface event

User interface event (load)

User interface (UI) events occur as a result of interaction with the browser window rather than the HTML page contained within it, e.g., a page having loaded or the browser window being resized.

```
function setup() {                          // Declare function
var textInput;                             // Create variable
textInput = document.getElementById('username');
// Get username input
textInput.focus();
// Give username focus
}

window.addEventListener('load', setup, false);
// When page loaded call setup()

/* LONGER VERSION WITH IE8 (and lower) compatibility
if (el.addEventListener) {
el.addEventListener('click', function(e)
{       itemDone(e);
}, false);}
 else {    el.attachEvent('onload', function(e){
    itemDone(e);    });}*/
```

Focus and blur event

```
function checkUsername() {                          // Declare function
  var username = el.value;                          // Store username in variable
  if (username.length < 5) {                        // If username < 5 characters
    elMsg.className = 'warning';                    // Change class on message
    elMsg.textContent = 'Not long enough, yet...';// Update message
  } else {                                          // Otherwise
    elMsg.textContent = '';                         // Clear the message
  }
}
function tipUsername() {                            // Declare function
    elMsg.className = 'tip';                        // Change class for message
    elMsg.innerHTML = 'Username must be at least 5 characters'; // Add message
}

var el = document.getElementById('username');       // Username input
var elMsg = document.getElementById('feedback');    // Element to hold message

// When the username input gains / loses focus call functions above:
el.addEventListener('focus', tipUsername, false); // focus call tipUsername()
el.addEventListener('blur', checkUsername, false);// blur call checkUsername()
```

Mouse event

The mouse events are fired when the mouse is moved and also when its buttons are clicked

```
// Create the HTML for the message
var msg = '<div class=\"header\"><a id=\"close\" href="#">close X</a></div>';
msg += '<div><h2>System Maintenance</h2>';
msg += 'Our servers are being updated between 3 and 4 a.m. ';
msg += 'During this time, there may be minor disruptions to service.</div>';

var elNote = document.createElement('div');          // Create a new element
elNote.setAttribute('id', 'note');                   // Add an id of note
elNote.innerHTML = msg;                              // Add the message
document.body.appendChild(elNote);                   // Add it to the page

function dismissNote() {                             // Declare function
  document.body.removeChild(elNote);                 // Remove the note
}

var elClose = document.getElementById('close');    // Get the close button
elClose.addEventListener('click', dismissNote, false);// Click close-clear note
```

Thank you.