Javascript and the DOM

Goes with examples in javascript2 directory

The Browser DOM (from w3schools.com)

The DOM is a W3C (World Wide Web Consortium) standard.

- The DOM defines a standard for accessing documents:

  "The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

- The W3C DOM standard is separated into 3 different parts:
  1. Core DOM - standard model for all document types
  2. XML DOM - standard model for XML documents
  3. HTML DOM - standard model for HTML documents

The HTML DOM is a standard for how to get, change, add, or delete HTML elements. It is a standard **object** model and **programming interface** for HTML. It defines:

- The HTML elements as **objects**
- The **properties** of all HTML elements
- The **methods** to access all HTML elements
- The **events** for all HTML elements

Javascript W3C DOM API Overview

Document class

- Represents top-level document
  - Also a specialized version representing the HTML page

Element class

- Represents XML/HTML element
- Inherits Node methods plus has some extras

Node class

- Represents node in DOM tree
  - Element is main node type, but there are also text nodes, CDATA notes, and a few others
- Most Element methods inherited from here

W3C DOM Syntax Summary

Document class

- Properties
  - `documentElement`
- Methods
  - `getElementById (HTML only), getElementsByTagName`

Element class

- Methods
  - `getAttribute, getElementsByTagName, hasAttribute`

Node

- Properties
  - `attributes, childNodes, firstChild, lastChild, nextSibling, nodeName, nodeType, nodeValue, parentNode, previousSibling`
- Methods
  - `hasAttributes, hasChildNodes, normalize, write`

The W3C DOM Document Class

`documentElement` property

- The root Element of the document

`getElementById` method

- Returns the Element with the specified ID.
  - For HTML documents only!
    - Refers to attribute that the DTD defines as an "id attribute", not necessarily named "id". Does not match attributes named "id" in regular XML docs.

`getElementsByTagName`

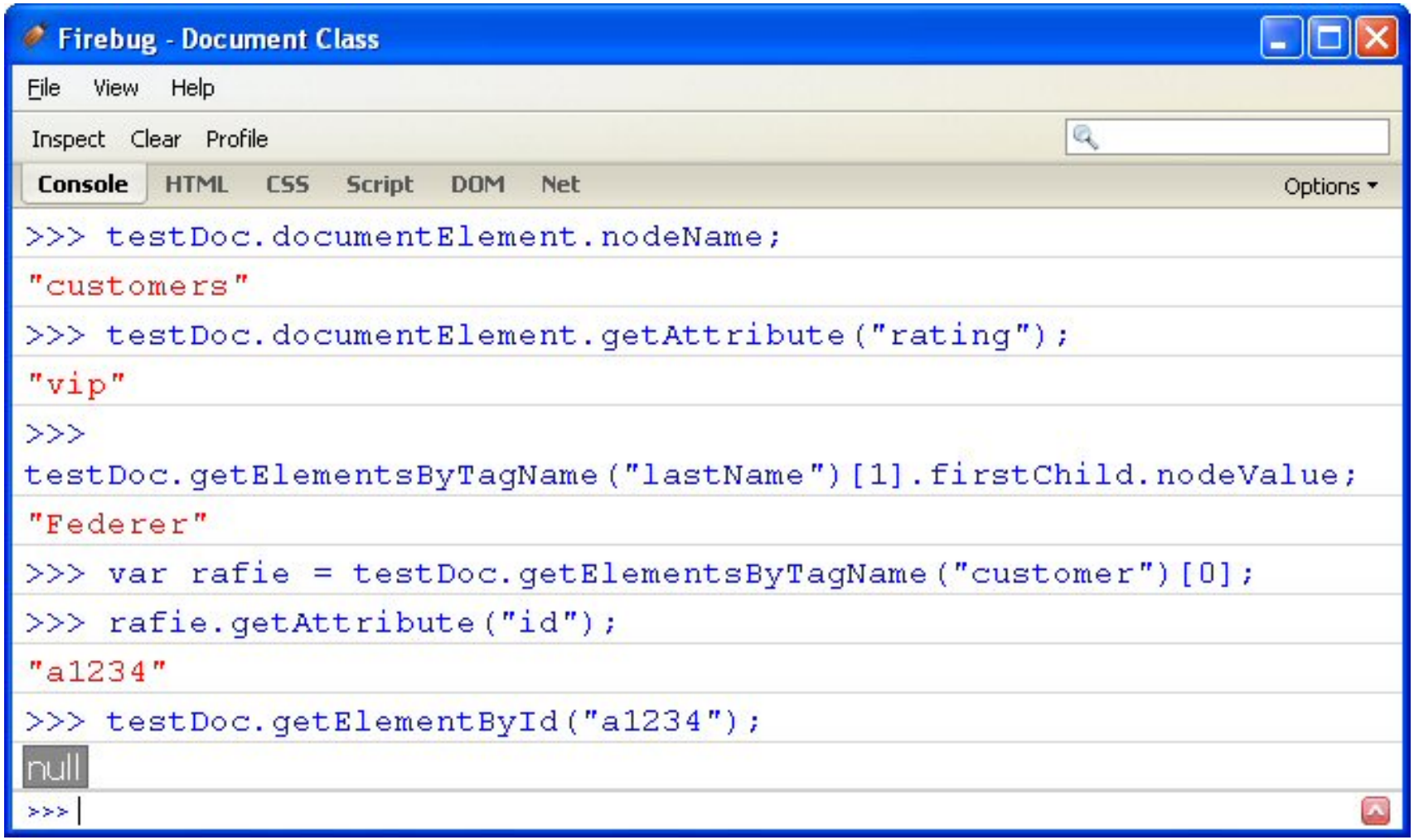
- Returns an array of Elements that have that tag name
  - Can use "*" for all Elements in document
  - Is case-sensitive for regular XML documents
  - Is case-insensitive for HTML documents
    - Even when using XHTML

Document Class: Examples (*document.js*)

```
function getXmlDoc(xmlString) {
  var parser = new DOMParser();
  var xmlDocument =
    parser.parseFromString(xmlString, "application/
   xml");
  return(xmlDocument);
}
var test =
  "<customers rating='vip'>" +
    "<customer id='a1234'>" +
      "<firstName>Rafael</firstName>" +
      "<lastName>Nadal</lastName>" +
    "</customer>" +
    "<customer id='a1235'>" +
      "<firstName>Roger</firstName>" +
      "<lastName>Federer</lastName>" +
    "</customer>" +
  "</customers>";
var testDoc = getXmlDoc(test);
```

Note this example is XML and is used On the following slides!

Document Class: Examples (Results)



The W3C DOM Element Class

`getAttribute`

- Gets value of designated attribute.
- E.g., if element refers to `<foo bar="a" baz="b">...</foo>`, `element.getAttribute("baz")` returns `"b"`

`getElementsByTagName`

- Returns an array of subelements that have this tag name
- Subelements can be arbitrarily nested

hasAttribute

- Tests if element has attribute of given name

Also inherits from Node class

- See next slides
- All Elements are Nodes, but not vice versa

The W3C Node Class: Properties

`attributes`

- An array of the attribute names of the Node

`childNodes`

- An array of direct child nodes. 0-length if no children.

`firstChild, lastChild, parentNode`

- Specific child nodes. Parent node (null for top element).

`nextSibling, previousSibling`

- Related children of the parent node

`nodeName`

- For Element nodes, the XML element name

`nodeType`

- `Node.ELEMENT_NODE, Node.TEXT_NODE, Node.CDATA_SECTION_NODE`, and a few other options

`nodeValue`

- For Text nodes, the body content.
- Call `normalize` first. See next slide.

The W3C Node Class: Methods

`hasAttributes`

- Does this Node have any attributes at all?

`hasChildNodes`

- Does this Node have any children at all?

`normalize`

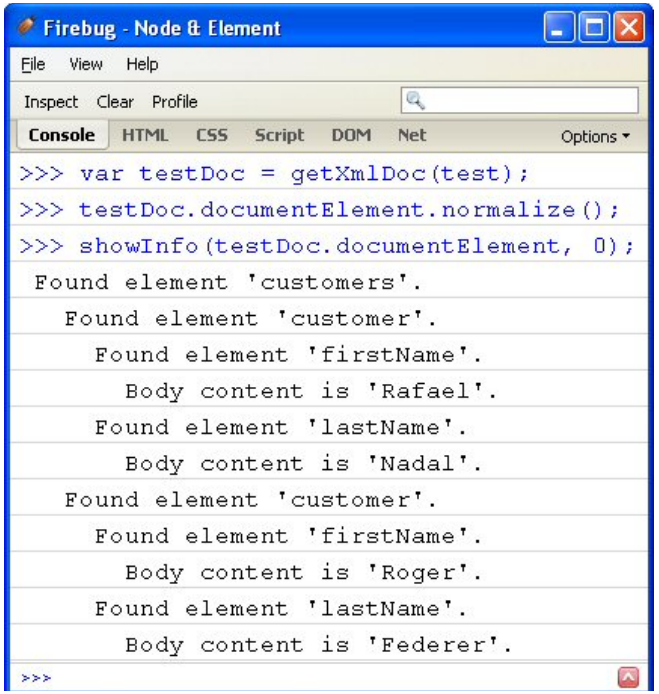
- Merges empty text nodes and removes empty ones.
- Important if element has body content that spans multiple lines. Body content is nodeValue of the child text node.

```
// Element is <foo>bar</foo>
var element = someElement();
element.normalize();
var bodyContent = element.firstChild.nodeValue;
// bodyContent is now "bar"
```

Node and Element: Example (*node+element.js)*

```
function showInfo(node, indent) {
  if (node.nodeType == Node.TEXT_NODE) {
    console.log("%s Body content is '%s'.",
                spaces(indent), node.nodeValue);
  } else if (node.nodeType == Node.ELEMENT_NODE) {
    console.log("%s Found element '%s'.",
                spaces(indent), node.nodeName);
    var children = node.childNodes;
    for(var i=0; i<children.length; i++) {
      showInfo(children[i], indent+1);
    }
  }
}

function spaces(n) {
  var indentString = "  ";
  var result = "";
  for(var i=0; i<n; i++) {
    result =
     result.concat(indentString);
  }
  return(result);
}
```



HTML specific: HTMLDocument Properties

Obtain with predefined 'document' variable in a browser

- Specialized subclass of Document class discussed earlier

Properties

- `title, domain, URL`
  - Info about the document. URL is same as `window.location.href` unless redirection occurs
- `body`
  - The body element
- `anchors, applets, forms, images, links`
  - Arrays of subelements, in the order they appear in the document. Usually better to find elements by ids.
- `cookie, lastModified, referrer`
  - In Ajax, it is usually better to manipulate these on server
- blah
  - Element that has name="blah" (first one if repeated)

Browse this using the DOM tab in Firebug!

See docprops.js

HTML specific: HTMLDocument Methods

`write, writeln`

- Dynamically insert text into document
- Used from `<script>` tag that has body content
- Not used by Ajax response handlers
  - Use `HTMLElement.innerHTML` property instead

`getElementsByName`

- Returns array of elements that have given name attribute

`getElementsByTagName`

- Returns array of elements that have given element name
  - Case insensitive
  - Inherited from Document class (see earlier slide)

`getElementById` (inherited from W3C DOM Document)

- Finds element with specified id attribute
  - Inherited from Document class (see earlier slide)

HTML specific: HTMLElement

Subclass of W3C DOM Element class. Obtain with

- `document.body`, `document.getElementsByTagName`, `document.getElementsByName`, `document.images`. etc.
- `otherElement.getElementsByTagName`, `otherElement.childNodes`, `otherElement.firstChild`, etc.

Most important properties

- `Id` – The id attribute
- `nodeName` – Element name (inherited from Node class).
- `Name` – The name attribute (for HTML elements with "name" only)
- `innerHTML` – Read/write property giving HTML text inside element
- `Style` – CSS2Properties object representing element styling
- `className` –Space-separated list of CSS class names

Method

- `scrollIntoView` – Scrolls browser so element is visible

HTML: Form Class

Obtaining reference

- `document.forms` array, "form" variable for code invoked by input element
- Any method or property that returns Node (`getElementById`, `childNodes`, etc.)

Properties

- `elements`: Array of all input elements in form
- `action, encType, method, name, target`
  - Corresponds to HTML attributes

Methods

- `submit, reset`

So yes you can programmatically manipulate web forms, reading values for custom checks, or to ensure you are in a state to submit

HTML: Input Class

Obtaining reference

- `theForm.elements` array
- Any method or property that returns Node (`getElementById`, `childNodes`, etc.)

Properties

- `name, id, value, type, disabled, form` (enclosing form)
  - For all input elements
- `defaultValue`
  - Initial value as given in the HTML
- Type-specific properties (see API)
  - `checked, maxLength, useMap`,...

Methods

- `blur/focus` (all), `click` (buttons, checkboxes, radio buttons), `select` (file, password, text)

HTML : Window Class Properties & Methods

Obtaining reference

- Use "window" or "self"

`history`

- History object. Not writable.

`location`

- Location object.
- `location.href` = "new address" redirects browser

`status`

- Status line value. Writable.

Size/position/scrolling

- `innerWidth`, `innerHeight`, `outerWidth`, `outerHeight`, `screenX` (or `screenLeft`), `screenY` (or `screenTop`)

`alert`, `confirm`, `prompt`

- Pops up dialog box

`print`

- Invokes print dialog

`setInterval`, `clearInterval`

- `setInterval(someFunction, milliseconds)`

`setTimeout`, `clearTimeout`

- `setTimeout(someFunction, milliseconds)`

`getComputedStyle`

- Get style info for specified element

Movement

- Lots of methods for opening, closing, resizing windows

Note: This is outside the DOM. Window is a ref to the browser itself! This is usually referred to as the "Browser Object Model", or the "BOM"

Javascript DOM Summary

Javascript on the Web:

- Get a hang for DOM manipulation!
  - Lots of sites do it
  - Lots of frameworks use it underneath the hood
- JSON is a must
  - Particularly when we get to AJAX and API-driven apps
- There are a lot of toolkits, invariably on real sites you will use one, two, … many!
  - JQuery is especially popular, though some argue too popular
  - If you use a toolkit, know what it is for!
    - http://robrich.org/slides/dozen_javascript_libraries/#/