Client/Server Programming for Internet Applications

TCSS460

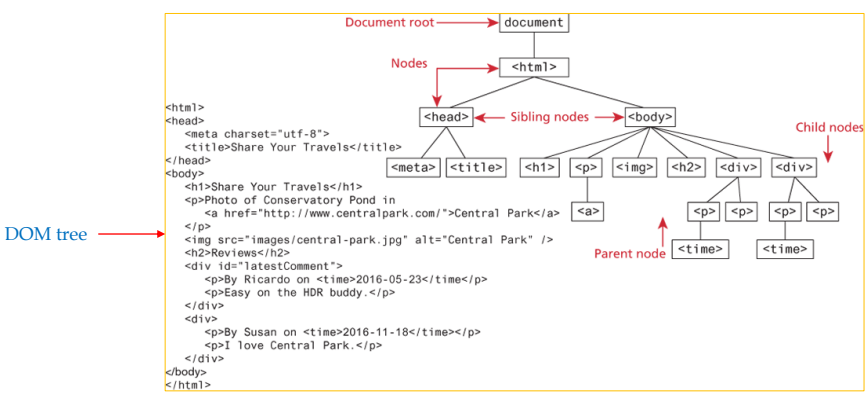
Summer 2020

Lecture 5 JavaScript (Part II) - DOM

TACOMA

Document Object Model (DOM)

- **JavaScript** is generally used to interact with the HTML document in which it is contained. This is achieved via an API called **Document Object Model (DOM)**

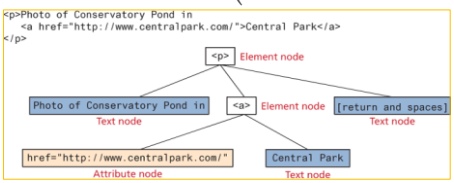


Randy Connolly, Ricardo Hoar, Fundamentals of Web Development (2nd Edition), 2017

2

Nodes and NodeLists

- **node**: an element within HTML document (i.e. an individual branch)
  - element nodes
  - text nodes
  - attributes nodes



  - all nodes share a common set of **properties** and **methods**
    - we can
      - **retrieve** information about the node
      - **manipulate** its properties
        - i.e. changing its CSS properties or retrieving its text content
      - and even **create** new content
- **nodelist**: a collection of nodes

Randy Connolly, Ricardo Hoar, Fundamentals of Web Development (2nd Edition), 2017 3

Nodes and NodeLists (cont'd)

Property	Description
`attributes`	Collection of node attributes
`childNodes`	A `NodeList` of child nodes for this node
`firstChild`	First child node of this node
`lastChild`	Last child of this node
`nextSibling`	Next sibling node for this node
`nodeName`	Name of the node
`nodeType`	Type of the node
`nodeValue`	Value of the node
`parentNode`	Parent node for this node
`previousSibling`	Previous sibling node for this node
`textContent`	Represents the text content (stripped of any tags) of the node

Randy Connolly, Ricardo Hoar, Fundamentals of Web Development (2nd Edition), 2017 4

document object

- **document object**: root JavaScript object representing entire HTML document
  - globally accessible via **document** object reference
  - some ***properties*** are read-only while others are modifiable

```
// retrieve the URL of the current page
var a = document.URL;
// retrieve the page encoding, for example ISO-8859-1
var b = document.inputEncoding;
```

accessed via the dot notation

  - several essential ***methods*** you will use all the time
    - i.e. `document.write` (`.writeln`)
    - **three** categories
      - **selection**
      - **family manipulation**
      - **event**

5

document object
→ selection methods

- most important DOM methods are those that allow the ***selection*** one or more document elements

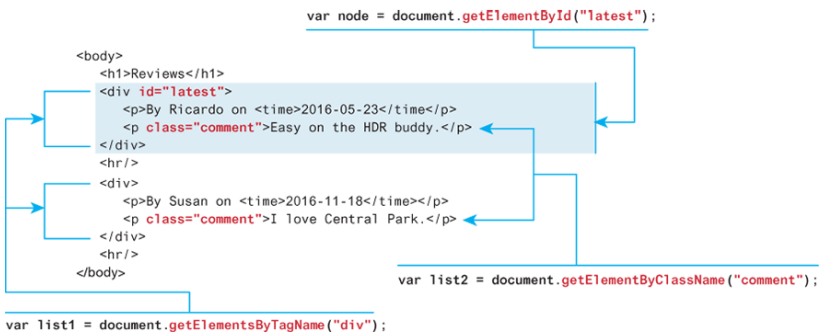
Method	Description
`getElementById(id)`	Returns the single element node whose `id` attribute matches the passed `id`.
`getElementsByClassName(name)`	Returns a `NodeList` of elements whose class name matches the passed `name`.
`getElementsByTagName(name)`	Returns a `NodeList` of elements whose tag name matches the passed `name`.
`querySelector(selector)`	Returns the first element node that matches the passed CSS selector.
`querySelectorAll(selector)`	Returns a `NodeList` of elements that match the passed CSS selector.

Randy Connolly, Ricardo Hoar, Fundamentals of Web Development (2nd Edition), 2017

6

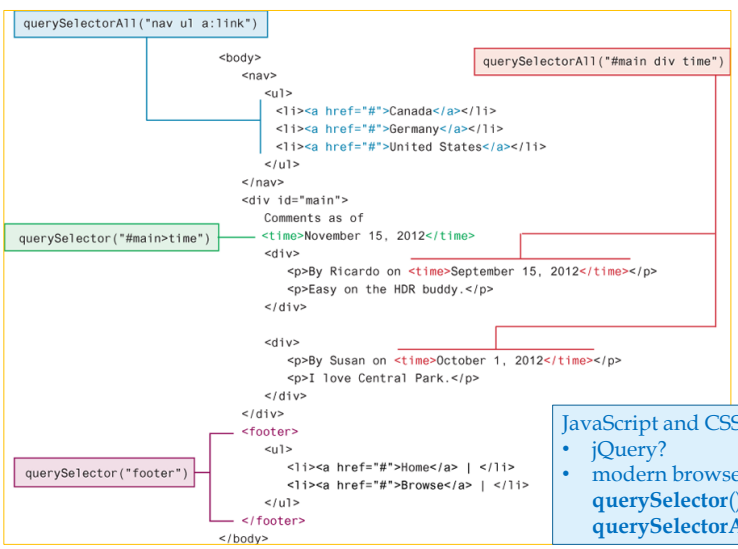
document object → selection methods (cont'd)

- **getElementById**
  - returns single DOM element (i.e. node)
- **getElementsByClassName** and **getElementsByTagName**
  - return an "array" of node elements (i.e. NodeList)



Randy Connolly, Ricardo Hoar, Fundamentals of Web Development (2nd Edition), 2017 7

document object → selection methods (cont'd)



JavaScript and CSS selectors?
- jQuery?
- modern browsers now support **querySelector**() & **querySelectorAll**() methods

Randy Connolly, Ricardo Hoar, Fundamentals of Web Development (2nd Edition), 2017 8

document object → element node object

- represents an HTML element in the hierarchy

Some Essential Element Node Properties

Property	Description
`classList`	A read-only list of CSS classes assigned to this element. This list has a variety of helper methods for manipulating this list.
`className`	The current value for the `class` attribute of this HTML element.
`id`	The current value for the `id` of this element.
`innerHTML`	Represents all the content (text and tags) of the element.
`style`	The style attribute of an element. This returns a `CSSStyleDeclaration` object that contains sub-properties that correspond to the various CSS properties.
`tagName`	The tag name for the element.

Randy Connolly, Ricardo Hoar, Fundamentals of Web Development (2nd Edition), 2017 9

document object → element node object

- represents an HTML element in the hierarchy

Some Specific HTML DOM Element Properties for Certain Tag Types

Property	Description	Tags
`href`	Used in `<a>` tags to specify the linking URL.	`a`
`name`	Used to identify a tag. Unlike `id` which is available to all tags, `name` is limited to certain form-related tags.	`a, input, textarea, form`
`src`	Links to an external URL that should be loaded into the page (as opposed to `href` which is a link to follow when clicked).	`img, input, iframe, script`
`value`	Provides access to the `value` attribute of input tags. Typically used to access the user's input into a form field.	`input, textarea, submit`

Randy Connolly, Ricardo Hoar, Fundamentals of Web Development (2nd Edition), 2017 10

document object → element node object

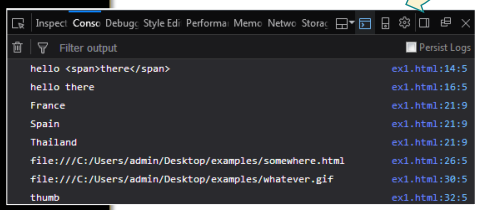
ex1

```
<p id="here">hello <span>there</span></p>
<ul>
  <li>France</li>
  <li>Spain</li>
  <li>Thailand</li>
</ul>
<div id="main">
  <a href="somewhere.html"><img src="whatever.gif" class="thumb"></a>
</div>

<script>
  var node = document.getElementById("here");
  // outputs: hello <span>there</span>
  console.log(node.innerHTML);
  // outputs: hello there
  console.log(node.textContent);
  var items = document.getElementsByTagName("li");
  for (var i=0; i<items.length; i++) {
     // outputs: France, then Spain, then Thailand
     console.log(items[i].textContent);
  }
  var link = document.querySelector("#main a");
  // outputs: somewhere.html
  console.log(link.href);
  var img = document.querySelector("#main img");
  // outputs: whatever.gif
  console.log(img.src);
  // outputs: thumb
  console.log(img.className);
</script>
```

accessing elements and their properties

browser console (F12)



Modifying DOM Elements → changing element's style

change an element's **style** (e.g. background color & border)

```
var node = document.getElementById("someId");
node.style.backgroundColor = "#FFFF00";
node.style.borderWidth = "3px";
```

```
<style>
    .box {
        margin: 2em; padding: 0;
        border: solid 1pt black;
    }
    .yellowish { background-color: #EFE63F; }
    .hide { display: none; }
</style>
<main>
    <div class="box">
    ...
    </div>
</main>
```

```
var node = document.querySelector("main div");

1 node.className = "yellowish";
  This replaces the existing class specification with this one. Thus the <div> no longer has the box class

2 node.classList.remove("yellowish");
  node.classList.add("box");
  Removes the specified class specification and adds the box class

3 node.classList.add("yellowish");
  Adds a new class to the existing class specification

4 node.classList.toggle("hide");
  If it isn't in the class specification, then add it

5 node.classList.toggle("hide");
  If it is in the class specification, then remove it
```

Equivalent to:

```
1 <div class="yellowish">
2 <div class="">
  <div class="box">
3 <div class="box yellowish">
4 <div class="box yellowish hide">
5 <div class="box yellowish">
```

Modifying DOM Elements
→ changing element's style (cont'd)

ex2

change the **style** of an HTML element (body for this example)

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>ex2: changing element's style</title>
</head>

<body>
<script type="text/javascript">
     var colors = ["d6eaf8", "a9cce3", "aed6f1", "85c1e9", "5dade2", "3498db", "2e86c1", "2874a6"];
     document.body.style.backgroundColor = "#" + colors[Math.floor(Math.random() * 7) + 1];
</script>
</body>
</html>
```

13

DOM Timing

- the timing of any DOM code is very important
  - you cannot access or modify the DOM until it has been **loaded**

this example will not work since body is not loaded yet

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>ex2: changing element's style
</head>
<script type="text/javascript">
     var colors = ["d6eaf8", "a9cce3",
     document.body.style.backgroundCo
</script>

<body>
</body>
</html>
```

- need a tool for checking your programming code?
- a **linter** is a program that checks your code for syntactical and stylistic correctness (e.g. **JSLint** and **JSHint**)

14

Modifying DOM Elements
→ changing element's content

ex3

we can programmatically access the content of an element node though its **`innerHTML`** or **`textContent`** properties

change the **content** of an HTML element

```
<p id="Countries">Countries List:</p>
<ul>
    <li>United States</li>
    <li>France</li>
    <li>Spain</li>
    <li>Thailand</li>
</ul>


<script type="text/javascript">
     var items = document.getElementsByTagName("li");
     for (var i=0; i<items.length; i++) {
        items[i].textContent = items[i].textContent + " (" + (i+1) + ')';
     }
</script>
```

15

Modifying DOM Elements
→ changing element's content (cont'd)

ex4

toggle **celsius** to **fahrenheit** example

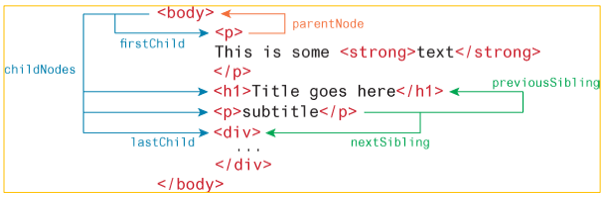
see complete example in ex4.html

```
if (status == 1) {
      document.getElementById("linkC").className = 'disabledLink';
      document.getElementById("linkC").onclick = "return false";
      document.getElementById("linkF").classList.remove("disabledLink");
      document.getElementById("linkF").href = "#";
      document.getElementById("linkF").onclick = function () { convert(0) };
      document.getElementById("unit").innerHTML = "&#8451;";
      for (var i = 0; i < elements.length; i++)
         elements[i].textContent = ((parseInt(elements[i].textContent) - 32) * 0.556).toFixed(0);
}
else {
      document.getElementById("linkF").className = 'disabledLink';
      document.getElementById("linkF").onclick = "return false";
      document.getElementById("linkC").classList.remove("disabledLink");
      document.getElementById("linkC").href = "#";
      document.getElementById("linkC").onclick = function () { convert(1) };
      document.getElementById("unit").innerHTML = "&#8457;";
      for (var i = 0; i < elements.length; i++)
         elements[i].textContent = ((parseInt(elements[i].textContent) * 1.8) + 32).toFixed(0);
}
```

16

Creating DOM Elements
→ DOM Manipulation Methods (cont'd)

- innerHTML approach can output badly formed HTML
- each node in the DOM has a variety of "family relations" **properties** and **methods** for navigating between elements and for adding or removing elements from the document hierarchy.




```
var node = document.getElementsByTagName("body").firstChild.firstChild;
```

Randy Connolly, Ricardo Hoar, Fundamentals of Web Development (2nd Edition), 2017 17

Creating DOM Elements
→ DOM Manipulation Methods

Method	Description
`appendChild`	Adds a new child node to the end of the current node. `aParentNode.appendChild(newNode)`
`createAttribute`	Creates a new attribute node. `var newAttribute = document.createAttribute("name");`
`createElement`	Creates an HTML element node. `var newElement = document.createElement("tag");`
`createTextNode`	Creates a text node. `var newText = document.createTextNode("text content");`
`InsertBefore`	Inserts a new child node before a reference node in the current node `aParentNode.insertBefore(newNode, referenceNode)`
`removeChild`	Removes a child from the current node. `aParentNode.removeChild(child)`
`replaceChild`	Replaces a child node with a different child. `aParentNode.replaceChild(newChild, oldChild)`

Randy Connolly, Ricardo Hoar, Fundamentals of Web Development (2nd Edition), 2017 18

Creating DOM Elements
→ DOM Manipulation Methods

ex5

FIFA World Cup 2018 Round 16

Saturday June 30th

- 7:00AM: France versus Argentina
- 11:00AM: Uruguay versus Portugal

Sunday July 1st

- 7:00AM: Spain versus Russia
- 11:00AM: Croatia versus Denmark

Monday July 2nd

Tuesday July 3rd

```
<div id="Round16">
    <h1>FIFA World Cup 2018 Round 16</h1>
    <p>Saturday June 30th</p>
    <ul id="Sat"> ... </ul>
    .
    .
    .
</div>
```

```
<body>
 <div id="Round16">
  <h1>FIFA World Cup 2018 Round 16</h1>
  <p>Saturday June 30th</p>
  <ul id="Sat">
   <li> 7:00AM: France versus Argentina</li>
   <li>11:00AM: Uruguay versus Portugal</li>
  </ul>
   ...
  <p>Monday July 2nd</p>
  <ul id="Mon">
  </ul>
  <p>Tuesday July 3rd</p>
  <ul id="Tue">...</ul>
</div>
</body>
```

```
<script type="text/javascript">
  var monMatch1 = document.createTextNode(" 7:00AM: Brazil?
                                          versus Belgium?");
  var listMon = document.createElement("li");
  listMon.appendChild(monMatch1)
  console.log(listMon);
  var monMatches = document.getElementById("Mon");
  monMatches.appendChild(listMon);
</script>
```

Randy Connolly, Ricardo Hoar, Fundamentals of Web Development (2nd Edition), 2017 19