CSC309 *Programming on the Web*

week 4: js, dom, forms

Amir H. Chinaei, Spring 2017

Office Hours: M 3:45-5:45 BA4222

ahchinaei@cs.toronto.edu
http://www.cs.toronto.edu/~ahchinaei/

review

- **design tips**
  - separate **semantics** from **appearance**
    - use semantic elements
  - for **responsive web design**, use
    - hybrid layout (mostly fluid layout)
    - max-width & min-width
    - box model and border-box for sizing
    - viewport, float, grid design, and @media
  - use browser **developer tools** & *html* & *css* **validators**
  - use **frameworks** and **templates**
- **this week**
  - separate **semantics**, **appearance**, **behavior**

javascript

- it's a web programming language
- to define/execute some **behaviour** in a document (web page)

- **brief history**
  - created by Netscape/Mozilla (1995)
    -
  - **XMLHttpRequest JS object** by Mozilla (2000)
  - first w3c specification of **XMLHttpRequest** (2006)

is it **java**?

- it has almost nothing do with **java**
  - it's prototyped-based OO
    -
  - it's dynamically typed
  - its var's are not block scoped
  - runs inside browsers
    -
  - c-like syntax

pros vs. cons

- **fat client** vs **thin client**
  - too thin is not good either!
  - client-side scripting helps
- **advantage**
  - reduce the load from servers
  - faster response by browser
  - more expressive power towards html
  - asynchronous requests
- **disadvantages**
  - client device may not support it, or disabled
  - inconsistencies from one browser to another
  - debugging and maintenance

<noscript>

- its content is seen by other processors
  - such as, web crawlers

- its content is shown if JS is not supported or disabled
  - useful for **fail safe design**

fail safe design

- **graceful degradation**

```
<p id="printIt">
 <a href="javascript:window.print()">Print this receipt.</a>
</p>

<noscript>
 <p>
  Use the print feature of your browser.
</p>
</noscript>
```

fail safe design

- **progressive enhancement**

```
<p id="printIt">Thank you. Please print this receipt for your records.</p>
<script type="text/javascript">
(function(){
 if(document.getElementById){
  var parent = document.getElementById('printIt');
  if(parent && typeof window.print === 'function'){
   var button = document.createElement('input');
   button.setAttribute('type','button');
   button.setAttribute('value','Print it');
   button.onclick = function(){
    window.print();
   };
   parent.appendChild(button);
  }
 }
})();
</script>
```

where js go?

inline js

```
<a href="javascript:window.print()">Print this receipt.</a>
```

embedded js

```
<script>
document.getElementById("demo").innerHTML = "My First JavaScript";
</script>
```

external js

```
<script src="myScript.js"></script>
```

syntax

- *c-like syntax*
  - *assignment, conditionals, loops, exception handling*
- *dynamically typed variable*
- ===
- *!==*
- *alert("hey");*
- *console.log("hey");*

objects

- Array, String, Date, etc.

  var myArray = new Array("orange", "blue");
  myArray=["orange", "blue"];

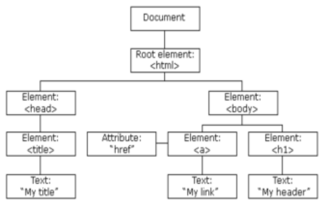
  - push(), pop(), sort(), concat(), join()

- String

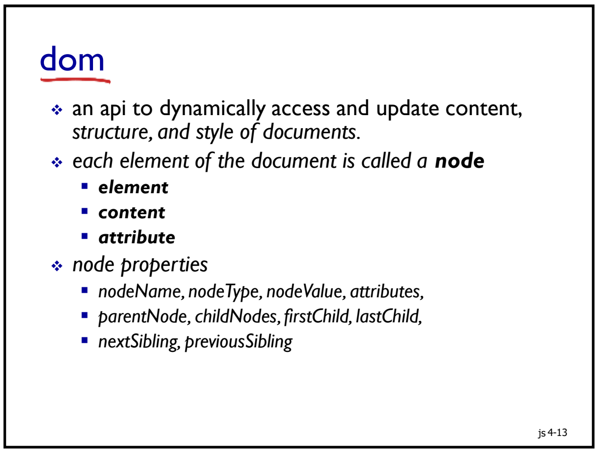
  var myString = "Hello World!";

  - split(), search(), match(), charAt(), indexOf()

dom



```
<!doctype html>
<html>
<head>
  <title>My title</title>
</head>
<body>
     <a href="xyz.html">My link</a>
     <h1>My header</h1>
</body>
</html>
```



dom

- an api to dynamically access and update content, *structure, and style of documents.*
- *each element of the document is called a* ***node***
  - ***element***
  - ***content***
  - ***attribute***
- *node properties*
  - *nodeName, nodeType, nodeValue, attributes,*
  - *parentNode, childNodes, firstChild, lastChild,*
  - *nextSibling, previousSibling*

js 4-13