Csci 4131 Internet Programming Spring 2024 Lecture 5 January 31st

Instructor: Dr. Dan Challou

Logistics – Csci 4131 Lecture 5, January 31st

- HW Assignment 2 out, available in week 3 module on the Homepage of the class Canvas site (and in the assignments section). Due Saturday 2/10 at 11:59pm

- ***Weekly readings and exercises are in your zyBook*** and additional readings, tutorials, programming homework due dates and exam dates are on the Class Canvas HomePage:
  - **Course Schedule: Weekly Class Readings and Tutorials, Exam Dates, and Programming Assignment Due Dates**

  in the Resources Module at the top of the Home

  Page on the class Canvas site

zyBooks Required and Optional/Bonus assignments!!!

Zybooks Homework 3 (Required) - Due 2/4

Optional / Bonus

HW1, HW2 Prep (Optional, Bonus) – Due 2/1
HW2 Prep, Part 1 (Optional, Bonus) – Due 2/3
HW2 Prep, Part 2 (Optional, Bonus) – Due 2/7

Suggested (Optional Tutorials): www.w3schools.com –

CSS Tutorial, JavaScript Tutorials:
https://www.w3schools.com/css/default.asp
http://www.w3schools.com/js/

Optional Reading, Sebesta Chapters 3,4

Questions?

Agenda

- Last Time:
  - HW2 Demo
  - Forms Revisited
  - Http methods (Get, Post) and URLS
  - Introduction to CSS
- Today:
  - CSS revisited
  - Into to the DOM / JavaScript

Recall, Units of measure for a Webpage

- https://www.w3.org/Style/Examples/007/units.en.html
- https://www.w3schools.com/cssref/css_units.asp

A CSS File for Styling the BOX

```
        /* A CSS Style file for formatting "the box" around some block elements  */


body {
        border: 3px dotted black;
        margin: 10px;
        }

section{
        border: 2px solid black;
        width: 500px;
        margin: 20px;           /* all four sides */
        padding: 10px;  /* all four sides */
        }
h1,p {
        border: 1px dashed black;
        padding: 10px;
        }
h1{
        margin: .5em 0 .25em; /* .5em top,0 right and left, .25em bottom */
        padding-left: 15px;
}
p{
        margin: 0; /* all four sides */
        padding-left: 15px;
}
```

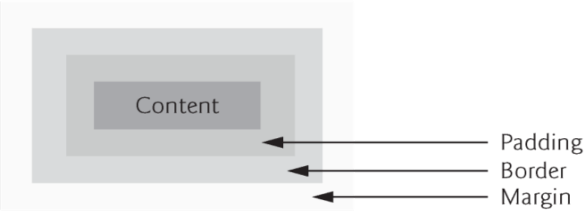


Fig. 4.13 | Box model for block-level elements.

HTML File That Uses our CSS Block Element Style File

```
<!DOCTYPE html>

<!-- Box Model Example -->
<html>
  <head>
    <meta charset = "utf-8">
    <title>Box Model Example</title>

    <!-- this begins the style sheet section -->
    <link rel="stylesheet" type="text/css" href="mystyle8.css">

  </head>
  <body>
              <section>
                            <h1>The University of Minnesota</h1>
                            <p>Educating the Leaders of Tomorrow for over 100 years.
                                There is always something happening at the U!</p>
              </section>

  </body>
</html>
```

Box example

Drawbacks of our Approach?

- Not Responsive Web Design
  - https://www.w3schools.com/html/html_responsive.asp

Lecture 5, Exercise 1: (Think, Pair, Share) – **submit the css file**

- Update the CSS file that I just did so it uses relative styling that enables the previous example seamlessly scales to the window size

  Rel_Box_Example.html

- You can use the online book, your phone or computer **for reference**

- **You can download the files: Box_Example.html and mystyle8.css from the Week 3 module on the class Canvas site**

- ***Note, there is one styling command that you must change to make sure the relative styling works – most of your task is to find it and fix it!!!!*** (Hint see: https://www.w3schools.com/html/html_responsive.asp)

- **Submit Your updated CSS file via the lecture 5, exercise 1 link on Canvas**
- ***Thumbs-up, close computing device when done.***

Questions?

Building a Web Page

- HTML – structure
- CSS – style
- JavaScript – behavior

- ***But what does JavaScript affect?***

The Document Object Model (DOM)

- **What is the HTML DOM?**
  - The HTML DOM is:
  - The data structure that holds the structure (HTML) specified by a Webpage and the style applied to it via CSS, and is what your browser uses to render (display) a web page.
  - A standard object model for HTML
  - A standard programming interface for HTML
  - Platform- and language-independent
- A W3C standard
- The HTML DOM defines the **objects and properties** of all HTML elements, and the **methods**(interface) to access them (via JavaScript).
- Summary:
  - **The HTML DOM is a standard for how to represent/store, get, change, add, or delete HTML elements.**
  - **See https://www.w3schools.com/js/js_htmldom.asp**

HTML DOM Nodes

- In the HTML DOM (Document Object Model), everything is a **node**:
- The document itself is a document node
- All HTML elements are element nodes
- All HTML attributes are attribute nodes
- Text inside HTML elements are text nodes
- Comments are comment nodes

Element Objects

- In the HTML DOM, the **Element object** represents an HTML element.
- Element objects can have **child nodes** of type element nodes, text nodes, or comment nodes.
- A **NodeList object** represents a list of nodes, like an HTML element's collection of child nodes.
- Elements can also have attributes. Attributes are attribute nodes.
- What Examples of Attributes (of HTML Elements) Can You Think Of???

Attributes vs Properties

- **When writing HTML source code, you can define attributes on your HTML elements**.
- **Then, once the browser parses your code, a corresponding DOM node will be created. This node is an object, and therefore it has properties.**
- For instance, this HTML element:

  <input type="text" value="Name:">

  has 2 attributes.
- Once the browser parses this code, an ***HTMLInputElement* object** will be created in the DOM, and this object will contain dozens of properties like:
  - accept, accessKey, align, alt, attributes, autofocus, baseURI, checked, childElementCount, childNodes, children, classList, className, clientHeight, etc.
- For a given DOM node object, properties are the properties of that object, and HTML attributes are the elements of the attributes property of that object

dicsussion courtesy of : http://stackoverflow.com/questions/6003819/properties-and-attributes-in-html

IN Summary

- Attributes – are what's written in HTML Elements.
  - https://www.w3schools.com/html/html_attributes.asp
- Properties (derived from HTML attributes) – are what's in DOM objects (found in the attribute property).
  - https://www.w3schools.com/jsref/dom_obj_attributes.asp

DOM Hierarchy

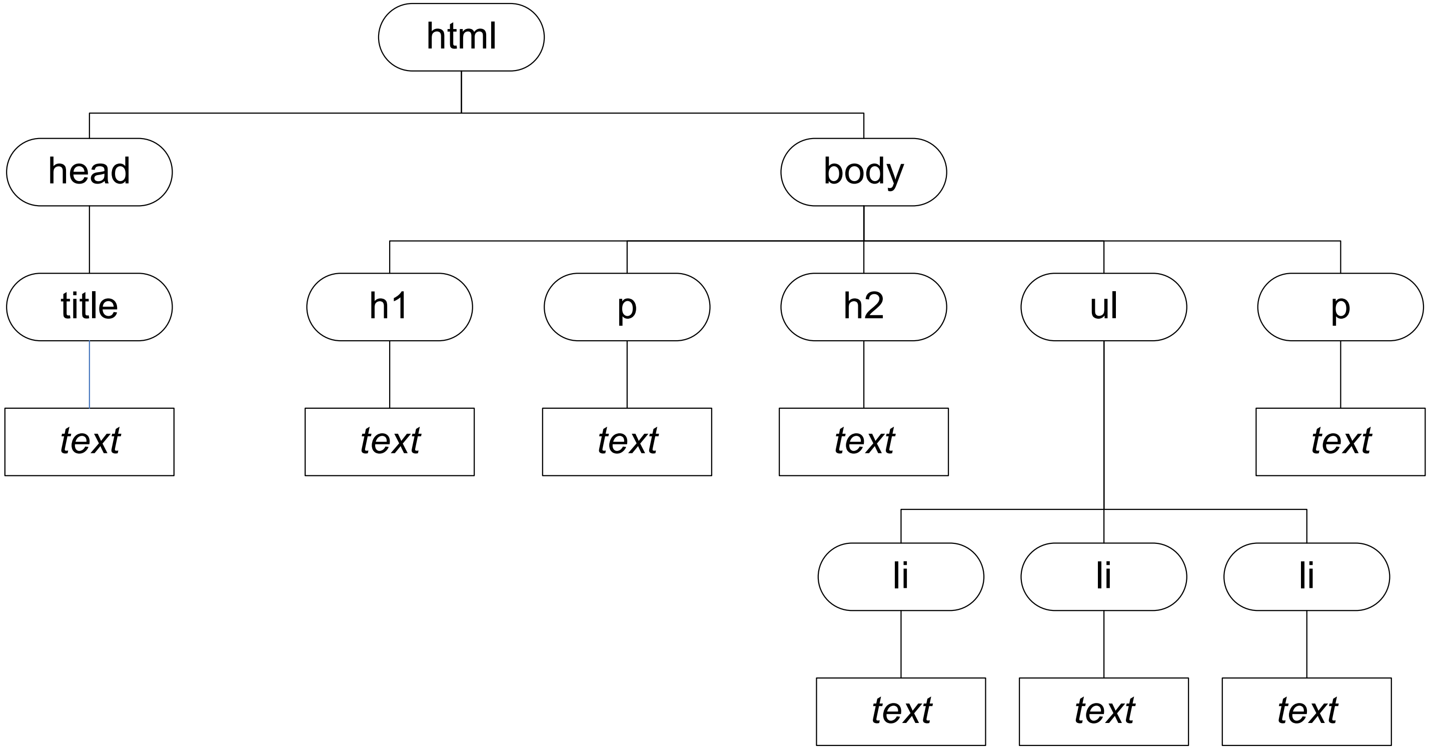
- https://www.w3schools.com/js/js_htmldom.asp

Consider the Following Web Page:

```
<!DOCTYPE html>
<html>
  <head>
                <meta charset = "utf-8">
                <title>An Example That Illustrates The Document Object Model</title>
  </head>
  <body>
                <h1> Activities Available at the University of Minnesota </h1>
                <p>  There are a multitude of things to do here at the University of
                        Minnesota.  You can attend sporting events, concerts, exercise, join a
                        club, get a work-study job, or just chill with your friends - and those are
                just a few possibilites</p>
                <h2> Entertaining, but under the radar activities</h2>
                <ul>
                        <li>Attend a  Gopher track meet</li>
                        <li>Go to a Gopher baseball game</li>
                        <li>Play ultimate frisbee on the quad</li>
                </ul>
                <p>That is my list - yours may be quite different.
                  But, the point is, you should get involved in an activity
                  of your choice, research shows that it will
                  help your grades and enhance your college
                  experience </p>
  </body>
</html>
```

Dom_ex.html

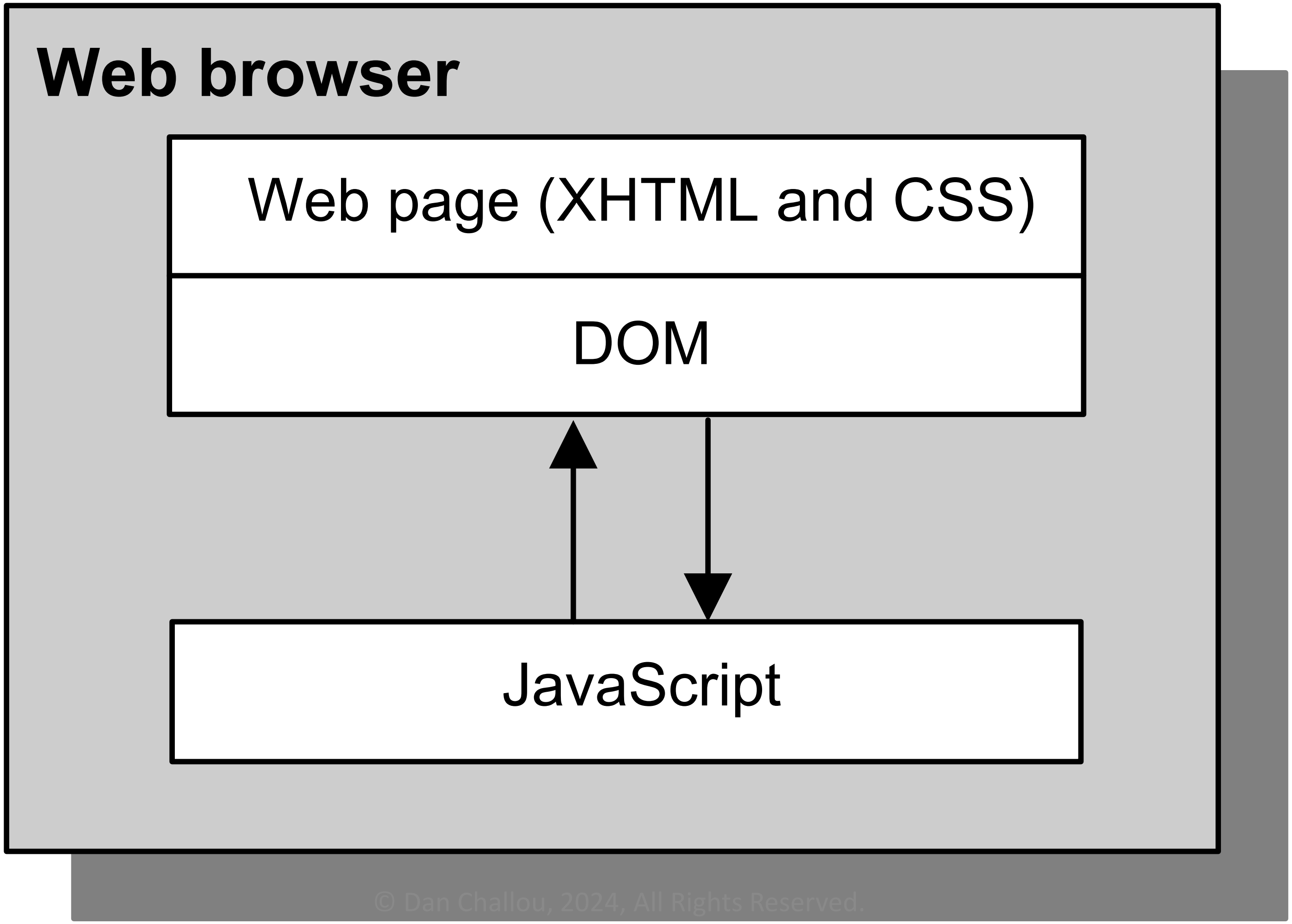
Think/Pair/Share: Is this the DOM for the Web Page specified on the previous slide?



The Document Object Model - Summarized

- The Document Object Model is a collection of nodes – **OBJECTS** - (in a tree) that store the data necessary render the current web page in a browser.
- The DOM for a web page is built as the page is loaded by the Web Browser
- Javascript can modify the web page in the browser by modifying the DOM
- When the DOM is changed, the web browser immediately displays the results of the change
- Many JavaScript Applications manipulate the DOM based on user actions

How Web Technologies Interact in a Web Browser



JavaScript

- Is not Java, its syntax is closer to C, but it isn't C either
- JavaScript is dynamically typed (C, C++, Java – statically typed)
- It is closer in nature to functional languages (Scheme, Lisp, Closure, OCaml, Smalltalk,…)
- It uses objects, and enables you to create your own objects
- It does have classes (a relatively recent addition):

  https://javascript.info/getting-started
- It does support inheritance (is-a, has-a), but it excels at the latter
- See: http://www.crockford.com/javascript/javascript.html

  for an overview

What can in-browser JavaScript do?

- In-browser JavaScript can do everything related to webpage manipulation, interaction with the user, and the webserver.
- For example, in-browser JavaScript can:
  - Add new HTML to the page, change the existing content, modify styles.
  - React to user actions, run on mouse clicks, pointer movements, key presses.
  - Send requests over the network to remote servers, download and upload files (so-called AJAX and COMET technologies).
  - Get and set cookies, ask questions to the visitor, show messages.
  - Remember the data on the client-side (“local storage”).

Examples of what in-browser JavaScript CAN’t do:

- JavaScript on a webpage may not read/write arbitrary files on the hard disk, copy them or execute programs. It has no direct access to OS system functions.
- Different tabs/windows generally do not know about each other. Sometimes they do, for example when one window uses JavaScript to open the other one. But even in this case, JavaScript from one page may not access the other if they come from different sites (from a different domain, protocol or port). (same origin policy)
- JavaScript can communicate over the net to the server where the current page came from. But its ability to receive data from other sites/domains is crippled.

The limits specified above do not exist if JavaScript is used outside of the browser, for example on a server. Modern browsers also allow plugin/extensions which may ask for extended permissions.

What Can’t in-browser JavaScript do?

- JavaScript’s abilities in the browser are limited for the sake of the user’s safety. The aim is to prevent an malicious actor from accessing private information or harming the user’s data.

Consider the following JavaScript (intent: read in 2 numbers, compute & print sum and product)

```
<script>
     var number1; // first string entered by user
     var number2; // second string entered by user
     var sum; // sum of number1 and number2
     var product; // product of number1 and number2

     number1 = window.prompt( "Enter first integer" ); // 6 entered by user
     number2 = window.prompt( "Enter second integer" ); //5 entered by user

     sum = number1 + number2; // add the numbers
     product = number1 * number2; //multiply the numbers

     document.writeln( "<h1>The sum is " + sum + "</h1>" );
     document.writeln("<h1>The product is " + product + "</h1>" );
</script>
```

What is printed when the JavaScript above is executed, and 6 and 5 are input?

add_n_mult.html

Exercise 2 – (Think/Pair/Share) - **submit your answer via the Lecture 5, Exercise 2 submission item in the week 3 module on Canvas**

- The JavaScript function **parseInt(*argument*)** takes a string as an ***argument*** and returns an integer
- Example: **parseInt(“23”)** returns the integer number ***23***
- Use the function **parseInt** to rewrite the JavaScript given on slide 23 so that when 6 and 5 are entered:
  - The program returns 11 as the sum, and
  - 30 as the product
- **You can download the web page add_n_mult.html from the week 3 module on Canvas and edit it!**

Please give a thumbs-up and close your computer when you are done!

Next time

- CSS, DOM, JavaScript Revisited