JavaScript

Chapter 8

- Once a page is sent from the web server to the client, the page is static.
- We want to create programs that run on the client's machine.
- We'll implement dynamic client-side interfaces using **JavaScript**

- JavaScript allows us to:
  - insert dynamic text into HTML.
  - react to mouse clicks and keyboard events.
  - get information about the client's machine.
  - perform calculations on a client's machine.
- JavaScript has similar syntax to Java but that is where the similarities end.
- JavaScript is much more relaxed.
- JavaScript is interpreted.

- Many of you are familiar with procedural programming.
- **Event-Driven Programming** - a program that waits for a user interaction before executing code.
- **Events** - mouse clicks, key presses, timer, etc.
- **Event Handler** - function that is called when an event is triggered.
- Events are handled by:
    - Deciding which event we want to respond to.
    - Writing a JavaScript function with code we want to run.
    - Attach that function to the event.

Step 1

- We want to create a control on our page.
- We will create a *button* that is unrelated to any form.

Example

```
<button>Button Text</button>
```

Step 2

- After our control has been placed, we need to write a write a JavaScript function to handle the event.
- First need to create a JavaScript file. Let's call it *example.js*.
- Inside that file, we'll create our function.

```
function buttonExample(){
    alert("Thanks for clicking me!");
}
```

Step 3

- Our last step is to attach our function to the button click.
- We must first get our JavaScript file loaded up.
- We put the following code in the header portion of our HTML page.
- For some reason, the script tag is not self-closing.

Example

```
<script src="example.js" type="text/javascript"></script>
```

Step 3

- Must attach our function to the button.
- There are several ways to attach a function to an event so we'll start off with the easiest way.
- **Event Attributes** - represent actions you can perform on a given HTML element. All events start with *on*, e.g. onclick, onchange, onsubmit…
- The original and modified HTML code is shown below.

Example

```
<button>Button Text</button>
<button onclick="buttonExample();">Button Text</button>
```

- Some browsers have JavaScript disabled.
- Some users have JavaScript blockers installed.
- **noscript** - an HTML tag that is shown if the client is not able to run JavaScript.

Example

<noscript><p>This page requires JavaScript to run correctly!</p></noscript>

- Each tag on a page is represented as an object.
- These objects are collectively referred to as the **Document Object Model**.
- Interacting with these objects let's us manipulate the web page's appearance dynamically.
- Objects can be accessed by id's.

Example

<img src="somePic.jpg" id="myPic" />
If we want to change the picture, we can use the following JavaScript in some function:

```
    var icon = document.getElementById("myPic");
    icon.src = "otherPic.jpg";
```

Example

In the HTML code:

<button onclick="buttonFunction();">Button Example</button>

<p id="answer">Old text here.</p>

In the .js file:

```javascript
function buttonFunction(){
    var para = document.getElementById("answer");
    para.innerHTML = "New text here!";
}
```

- The changed text will only affect the current page.
- A page reload or someone else visiting the site will see the original text.

- There are 8 fundamental types in JavaScript: Number, Boolean, String, Array, Object, function, null, undefined.
- JavaScript is loosely typed. You never need to declare a type.
- Does implicit typing transparently.

- There are Number constants: Number.MAX_VALUE, Number.MIN_VALUE, Number.NaN, Number.NEGATIVE_INFINITY, Number.POSITIVE_INFINITY.
- If operands are not Numbers, JavaScript will attempt to convert them into a Number.
- Different from PHP, if JavaScript can't convert a value to a number, it becomes Number.NaN.

```
6 * 7 == 42
1 + 3.5 == 4.5
7 / 2 == 3.5
1 / 0 = Infinity
10 / "myString" == NaN
```

- JavaScript has variables and they must start with a letter or underscore. Variables can contain letters, numbers and underscores.
- Variables are declared and initialized by using: **var name = value;**
- Semi-colons are optional to end statements however they are strongly recommended. Page 26-28 on http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf explains automatic semicolon insertion.
- **Implicit Variable Declaration** - declaring a variable without using var (not recommended).
- Your favorite operators exist: ++, --, +=, *=, . . .

```
var someVar = 70;
someVar = "hello"; //does this work?
/* this is a multi-line
comment as expected */
```

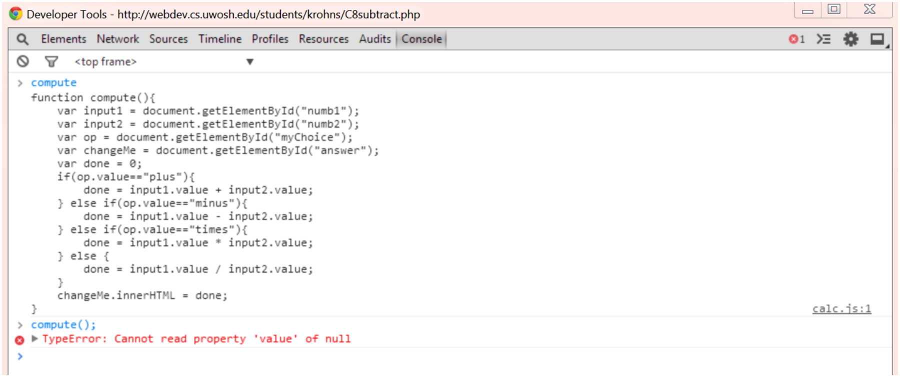
- As seen before, DOM objects can be accessed to alter the page content dynamically.
- Some properties are shared by all objects:
  - **className** - the class attribute, "" if none.
  - **id** - the id attribute, undefined if none.
  - **innerHTML** - HTML markup between opening and closing tags, "" if none.
  - **style** - object representing the style attributes.
  - **tagName** - the element's tag, e.g. div, p.
- Other attributes are also represented. For example, an img has a src attribute. Therefore, it has a src property.
- Many DOM property names are identical to the attribute name.

Example

Create a simple page to subtract one number from another.

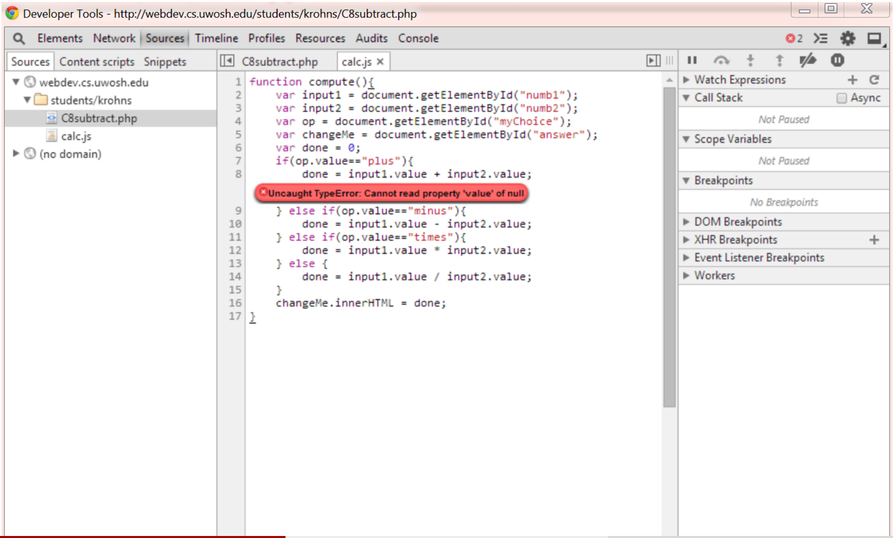
- You press a button and nothing happens. . . now what?
- Since JavaScript is not compiled, there is no compiler to give you error messages to fix your syntax.
- In many browsers, there are logs to view by hitting **Ctrl + Shift + J**. This will bring up the JavaScript/Error Console.
- Ensure the function is spelled correctly and the js file is being loaded correctly.
- You can use alert statements just like print statements but there are better ways.

- You can use the console to call functions without using the web page.
- Is a good first check to see if anything is working.
- One can type in short snippets of code to see what the results are. **This is very useful for debugging**.


Developer Tools - http://webdev.cs.uwosh.edu/students/krohns/C8subtract.php
Elements Network Sources Timeline Profiles Resources Audits Console
1
<top frame>

```
> compute
  function compute(){
      var input1 = document.getElementById("numb1");
      var input2 = document.getElementById("numb2");
      var op = document.getElementById("myChoice");
      var changeMe = document.getElementById("answer");
      var done = 0;
      if(op.value=="plus"){
          done = input1.value + input2.value;
      } else if(op.value=="minus"){
          done = input1.value - input2.value;
      } else if(op.value=="times"){
          done = input1.value * input2.value;
      } else {
          done = input1.value / input2.value;
      }
      changeMe.innerHTML = done;
  }                                                                    calc.js:1
> compute();
✖ ▶ TypeError: Cannot read property 'value' of null
>
```

- A very useful tool is the debugger with Chrome. Ctrl + Shift + J to open up the console. Then click on the *Sources* tab.
- If there was an error with your code, you will see the error here.



```
function compute(){
    var input1 = document.getElementById("numb1");
    var input2 = document.getElementById("numb2");
    var op = document.getElementById("myChoice");
    var changeMe = document.getElementById("answer");
    var done = 0;
    if(op.value=="plus"){
        done = input1.value + input2.value;
Uncaught TypeError: Cannot read property 'value' of null
    } else if(op.value=="minus"){
        done = input1.value - input2.value;
    } else if(op.value=="times"){
        done = input1.value * input2.value;
    } else {
        done = input1.value / input2.value;
    }
    changeMe.innerHTML = done;
}
```

Common Errors

- js file is not included.
- function is not being called correctly.
- Using the wrong id's with DOM elements.
- Make sure that the object you are trying to use isn't null. For example, use the console to check if the object exists.
- Ensure you are checking the right property. Do you need to use value or innerHTML?
- All of your JavaScript programs must have this at the top of them: **"use strict";**
  - Forces you to declare variables before using them.
  - Overwriting function definitions becomes illegal.
  - Disables some discouraged JavaScript keywords.

- Similar to many languages, JavaScript represents Strings as sequences of characters.
- There is no char type in JavaScript.
- Escape characters are similar to Java.
- The + operator is overloaded in JavaScript. It means concatenation when either operand is a String.

```
"3" + 7 == "37"
3 + "7" == "37"
3 + 5 + "7" == "87"
"3" + 5 + 7 == "357"
var one = "7";
var two = "8";
alert(one + two); //78
alert(one * two); //56
```

- **parseInt** - accepts a parameter and attempts to return a number, if it can't, returns NaN.

```
//what do these return?
parseInt("100");
parseInt("2 turtle doves");
parseInt("doves");
parseInt(2.67);
parseInt(0123); //some old browsers read this as octal
parseInt(0123, 10);
parseFloat(3);
parseFloat(3.45);
parseFloat("3.45");
```

- There are many String methods built into JavaScript. You can find a listing at http://www.w3schools.com/js/js_strings.asp
- Some common ones you should know: charAt, charCodeAt, indexOf, substring, toLowerCase, trim, length.
- A string can be treated as an array of characters. In other words, stringVar.charAt(0) == stringVar[0]. However, you should avoid the [] syntax as it's not part of the specification and may not work on all browsers.

Example

Write a program to input the user's name into three text boxes. Provide a button to clear all boxes and provide a button to output the user's initials into some span. Make sure the initials are all uppercase even if the user didn't type them in like that.

- The *Math* object is a built-in object with many useful functions. A list is at http://www.w3schools.com/jsref/jsref_obj_math.asp
- **null** - indicates a lack of a value.
- **undefined** - something that is not specified or defined.
- A null variable exists but has no meaningful/useful value.
- undefined is returned by a function with no return value.

Control statements are almost identical to Java. Be careful with equality checking.

```
var x = 10;
if(x == "10"){
 alert("this prints");
}
if(x === "10"){
 alert("this doesn't print");
}
if(x != "10"){
 alert("this doesn't print");
}
if(x !== "10"){
 alert("this prints");
}
if(x == "10 days"){
 alert("this doesn't print");
}
if(x == parseInt("10 days")){ //also true for ===
 alert("this prints");
}
```

- Control statements are almost identical to Java.
- Be careful with equality checking.
- You should always use === instead of == because of oddities. 0 == "", null == undefined, " \t " == 0.

```
var x = 10;
if(x = 20){
 alert("equals");
} else {
 alert("not equals");
}
```

- Control statements are almost identical to Java. Examples are shown on the next few slides.
- Be careful with equality checking.

```
for(initialization; bool; incrementStep){
 //code here
}
while(bool){
 //code here
}
do{
 //code here
} while(bool);
```

- In JavaScript, Boolean is either true or false.
- Any primitive value can be evaluated as a Boolean type.
- The following values are considered false: 0, 0.0, NaN, ‘’, ””, null, undefined.
- Your favorite logical operators exist: &&, || and !
  - Logical operators can act on non-Boolean operands.
  - && will return the left operand if it can be converted to false.
  - || will return the left operand if it can be converted to true. Useful as the **default operator**.
- JavaScript short circuits the same as Java.

```
if(someVar == null){ ... } //this is the Java way
if(someVar){ //this is the JavaScript way
    //code here
}
//default operator
var someVar = someOtherVar || "default value";
```

- A useful website for operator precedence is https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Operator_Precedence

- **Local variable** - a variable declared using the *var* keyword inside a function.
- **Global variable** - a variable declared outside of a function or a variable implicitly declared.

```
var a = 10; //global variable
//what are a, b, c at this point?
function scopeTest(){
    a = 4; //accessing the global variable
    b = 3; //implicitly declared global variable
    var c = 20; //local variable
}
//what are a, b, c at this point?
scopeTest();
//after scopeTest has been called once
//what are a, b, c at this point?
```

- Arrays are built into JavaScript.
- Array indices are 0-based.
- Array methods you should know: concat, indexOf, join, push, pop, reverse, shift, unshift, slice, splice and sort.
- For those of you who have had programming languages, the following functions are allowed: filter, map, reduce.

```
var arrName = []; //empty array
var arrName = [ele1, ele2, ..., eleN]; //array with
    values
```

- Functions are allowed in JavaScript and they can have parameters and return a value.
- No types are declared in the parameter list nor is the keyword *var*.
- The wrong number of parameters is not an error!
  - Too few arguments passed in and the remaining parameters are given the value of undefined.
  - Too many arguments passed in and the extra arguments are ignored.
- Can only return 1 value. If no value is returned, undefined is returned.

- **alert** is used to pop up a dialog box with some information.
- **prompt** is used to pop up a dialog box and request information from the user.
- **confirm** is used to confirm something. Returns true if ok is pressed, false otherwise.

```
var year = prompt("What year were you born?");
//stores value into year if the user hits ok,
//stores null into year if the user hits cancel
var ok = confirm("Really?");
```