- What is JavaScript?
- Language developed by Netscape
- Primary purpose is for "client-end" processing of HTML documents
    - JavaScript code is embedded within the html of a document
    - An interpreter in the browser interprets the JavaScript code when appropriate
    - Code typically allows for "preprocessing" of forms and can add "dynamic content" to a Web page

- How to include JavaScript in html?
- JavaScript programs require the <SCRIPT> tag in .html files

<script type = "text/javascript">

ACTUAL JavaScript code here

</script>

- These can appear in either the <HEAD> or <BODY> section of an html document
  - Functions and code that may execute multiple times is typically placed in the <HEAD>
    - These are only interpreted when the relevant function or event-handler are called

- Code that needs to be executed only once, when the document is first loaded is placed in the <BODY> close to, if not at the bottom

- Some browsers may not support scripting (i.e. have it turned off)
  - To be safe, you could put your scripts in html comments as specified in Sebesta text - I wouldn't suggest this though
    - <noscript></noscript>
    - This way browsers that do not recognize JavaScript will look at the programs as comments
- Note that, unlike PHP scripts, JavaScript is visible in the client browser

- However, if we want to prevent the script itself from being (easily) seen, we can upload our JavaScript from a file
  - This will show only the upload tag in our final document, not the JavaScript from the file
  - Use the src option in the tag

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

    - However, the source is likely still not really hidden
    - Minify or uglify
    - Thus you should not "hardcode" into Javascript anything that you don't want the client to see

```
<html>
  <head>
   <title>First JavaScript Example</TITLE>
  </head>
  <body>
    <h2>This line is straight up HTML</h2>
    <div>
      <script type="text/javascript">
        document.write("These lines are produced by<br/>");
        document.write("the JavaScript program<br/>");
        alert("Hey, JavaScript is fun!");
      </script>
    </div>
    <h2>More straight up HTML</h2>
    <script type="text/javascript" src="asdf.js"></script>
  </body>
</html>
```

- JavaScript variables have no types
  - Type is determined dynamically, based on the value stored
    - This is becoming familiar!
    - The **typeof** operator can be used to check type of a variable
- Declarations are made using the **var** keyword
  - Can be implicitly declared, but not advisable
  - Declarations outside of any function are global
  - Declarations within a function are local to that function
  - Variables declared but not initialized have the value **undefined**

- Variable identifiers are similar to those in other languages (ex: Java)
    - Cannot use a keyword
    - Must begin with a letter, $, or _
        - Followed by any sequence of letters, $, _ or digits
    - Case sensitive

- Numeric operators in JavaScript are similar to those in most languages
  - +, –, *, /, %, ++, --
  - Precedence and associativity are also fairly standard
- Strings
  - Have the + operator for concatenation
  - Have a number of methods to do typical string operations
    - charAt, indexOf, toLowerCase, substring
  - Looks kind of like Java – intentionally

- Similar to PHP, with mixed number/string type expressions, JavaScript will coerce if it can
    - If operator is + and an operand is string, it will always coerce other to string
    - If operator is arithmetic, and string value can be coerced to a number it will do so
        - If string is non-numeric, result is NaN (NotaNumber)
    - We can also explicitly convert the string to a number using parseInt and parseFloat
        - Again looks like Java

- Relational operators:

  ==, !=, <, >, <=, >=

  - The above allow for type coercion. To prevent coercion there is also

  ===, !==

    - Similar to PHP

- Boolean operators

  &&, ||, !

  - &&, || are short-circuited (as in Java and PHP)
    - Discuss

- Control statements similar to Java
- if, while, do, for, switch
    - Variables declared in for loop header are global to the rest of the script
- Functions
- Similar to Java functions, but
    - Header is somewhat different

```
function name(param_list)
```

        - Return type not specified (like PHP, since JS has dynamic typing)
        - Param types also not specified

- Functions execute when they are called, just as in any language
- To allow this, function code should be in the <HEAD> section of the .html file
- Variables declared in a function are local to the function
- Parameters are all value
- No parameter type-checking
- Numbers of formal and actual parameters do not have to correspond
    - Extra actual parameters are ignored
    - Extra formal parameters are undefined
- All actual parameters can be accessed regardless of formal parameters by using the **`arguments`** array
- Hoisting -- UGHHHHHHHHHH!!!!!!!!!!!!

- More relaxed version of Java arrays
    - Size can be changed and data can be mixed
    - Cannot use arbitrary keys as with PHP arrays
- Creating arrays
    - Using the new operator and a constructor with multiple arguments

      ```
      var A = new Array("hello", 2, "you");
      ```
    - Using the new operator and a constructor with a single numeric argument

      ```
      var B = new Array(50);
      ```
    - Using square brackets to make a literal

      ```
      var C = ["we", "can", 50, "mix", 3.5, "types"];
      ```

- Array Length
  - Like in Java, length is an attribute of all array objects
  - However, in Javascript it does not necessarily represent the number of items or even mem. locations in the array
  - Unlike Java, length can be changed by the programmer
  - Actual memory allocation is dynamic and occurs when necessary
    - An array with length = 1234 may in fact have memory allocated for only a few elements
    - When accessed, empty elements are undefined

- Array Methods
- There are a number of predefined operations that you can do with arrays
  - concat two arrays into one
  - join array items into a single string (commas between)
  - push, pop, shift, unshift
    - » Push and pop are a "right stack" (add to end, remove last)
    - » Shift and unshift are a "left stack" (remove first, add to front)
  - sort
    - » Sort by default compares using alphabetical order
    - » To sort using numbers we pass in a comparison function defining how the numbers will be compared
  - reverse

- These operations are invoked via method calls, in an object-based way
  - Also many, such as sort and reverse are mutators, affecting the array itself
- JavaScript also has 2-dimensional arrays
  - Created as arrays of arrays, but references are not needed
  - var list = [[1, 2, 3], [4, 5, 6], [7, 8, 9]];

- JavaScript is an object-based language
- It is NOT object-oriented (technically…)
- It has and uses objects, but does not support some features necessary for object-oriented languages

HOWEVER

- New JS spec for 2016 will include classes, which means JS is moving to a more OO approach:
    - Polymorphism - because it is a dynamic language
    - Encapsulation - due to object creation patterns
    - Inheritance - due to code reuse pattern

- JavaScript objects are actually represented as property-value pairs
  - Actually similar to keyed arrays in PHP
  - The object is analogous to the array, and the properties are analogous to the keys
    - However, the property values can be data or functions (methods)
  - Ex:

```
var my_tv = new Object();
    my_tv.brand = ”Samsung";
    my_tv.size = 46;
    my_tv.jacks = new Object();
    my_tv.jacks.input = 5;
    my_tv.jacks.output = 2;
```

- Note that the objects can be created and their properties can be changed dynamically
- Also, objects all have the same data type – object
- We can write constructor functions for objects if we'd like, but these do not create new data types – just easy ways of uniformly initializing objects

```
function TV(brand, size, injacks, outjacks) {
 this.brand = brand;
 this.size = size;
 this.jacks = new Object();
 this.jacks.input = injacks;
 this.jacks.output = outjacks;
}
…
var my_tv = new TV("Samsung", 46, 5, 2);
```

- Once an object is constructed, I can change its properties if I want to
  - Let's say I want to add a method to my TV called "show_properties"

```
function show_properties() {
  document.write("<ul>");
  document.write("<li>Here is your TV:</li>");
  document.write("<li>Brand: " + this.brand + "</li>");
  document.write("<li>Size: " + this.size + "</li>");
  document.write("<li>Input Jacks:</li>");
  document.write("<li>" + this.jacks.input + "</li>");
  document.write("<li>Output Jacks:</li>");
  document.write("<li>" + this.jacks.output + "</li>");
    document.write("</ul>");
}
…
```

- We can do a lot with Javascript objects
- Even though Javascript is not truly object-oriented, we can program in an object-based way
    - Encapsulating data and methods within objects
    - Utilizing methods for operations on the objects
- We will be using Javascript objects a lot with client-side programming

- JavaScript regular expression handling is also based on that in Perl
- The patterns and matching procedures are the same as in Perl, Java and PHP (PCRE)
- However, now the functions are methods within a string object (similar to Java)

```
var s = "a man, a plan, a canal: panama";
var loc = s.search(/plan/);
var matches1 = s.match(/an/g);
var matches2 = s.match(/\w+/g);
var matches3 = s.split(/\W+/);
s = s.replace(/\W/g, "-");
```

- Note that match is similar to the PHP match function
  - » Returns the matched pieces as opposed to the non-matched pieces (that split returns)