CSCI311

JAVASCRIPT I

wow
such design

http://www.relativitycalculator.com/

And… xkcd colour names??!!:

https://www.w3schools.com/colors/colors_xkcd.asp

Today's plan

Quick look at group project

Learn basic JavaScript syntax

- variables, literals, constants
- functions
- scope

Learn to use simple alerts, prompts

Discover how errors are handled in browser

Group project: Coming up...

Monday in Lab:

- Time to start on client side code (HTML, CSS, Maybe JavaScript)
- Get informal feedback

Feb. 16:

- Functional Prototype due
- Midway Report due (who will do what, internal deadlines, stretch goals)

Feb. 23:

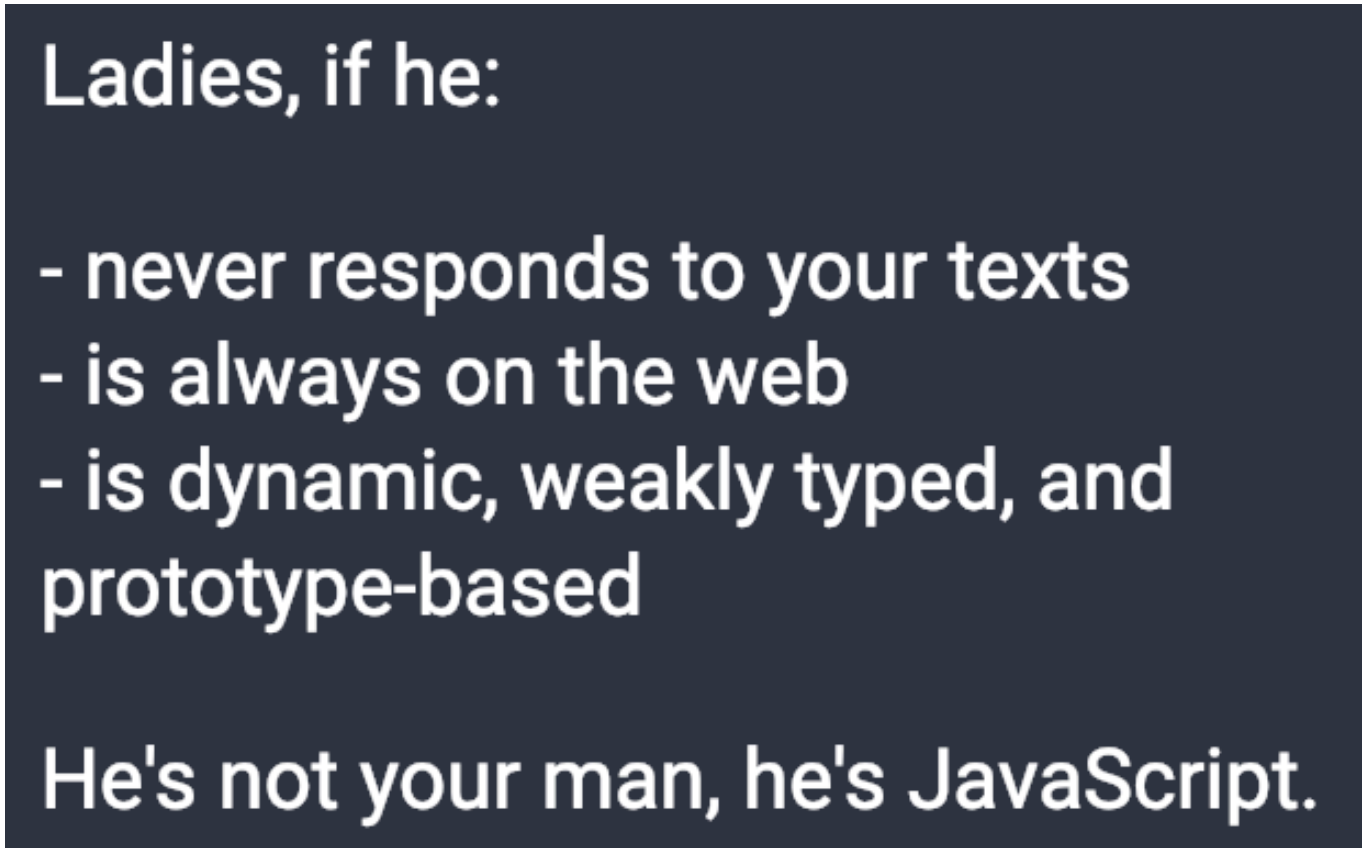
- Peer evaluations
- Team and self-assessments

Group Project: Requirements

Your project **must include**:

- Server side programming
- Client side programming
- Adherence to design principles
- Dynamic page generation (either server-side or client-side, or both)
- Database access and use
- HTML for structure
- CSS for style
- User login
- Adherence to security principles
- Adherence to accessibility principles

JavaScript!!!


Ladies, if he:
- never responds to your texts
- is always on the web
- is dynamic, weakly typed, and prototype-based
He's not your man, he's JavaScript.

Javascript and html

How do we include JavaScript in our html?

- in <script> </script> element

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

- in separate file:

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

- In event in button or other element (onclick="getElementById('foo').innerHTML = 'Hello!';")

as of HTML5, the "type" is optional and will default to text/javascript if not provided

javascript- modify HTML

Javascript updating the HTML page content using the DOM object document

- document.writeln

javascript: syntax

basic unit:

- one-line statement or expression, followed by a semicolon

Case-sensitive

Free Format

Familiar function syntax

Familiar Comment style

javascript: methods or functions

JS is Object Oriented, so we use dot-notation to access Methods of Objects

document.write(“Hi there!”);

document.writeln(“boo!?”);

Defining Functions

Function declaration

- function *name*([*param*[, *param*[, ... *param*]]]) { *statements* }

Function expression

- function [*name*]([*param*[, *param*[, ... *param*]]]) { *statements* }

Anonymous functions:

- var myFunction = function() { statements }

Immediately Invokable Function Expressions (IIFE):

- (function() { statements })();

Calling Functions

Specify the object that the method belongs to

- String.charAt();

If no object specifies, it is assumed to be the window object

- alert(“danger will robinson!”); // window.alert(“danger will robinson!”);

Quick and dirty JS Functions

prompt()

alert()

console.log()

JavaScript variables

Create variables using one of two statements:

- var
  - var *varname1 [= value1] [, varname2 [= value2] ... [, varnameN [= valueN]]]*;
- Let
  - let *var1* [= *value1*] [, *var2* [= *value2*]] [, ..., *varN* [= *valueN*];

var Scope:

- Current execution context
  - Enclosing function
  - Global (if not in a function)

let Scope:

- Scope of current block (and enclosed blocks

JavaScript Variables

Declared

- Constrained to execution context they're declared in
- Created before code executed
- Non-configurable (cannot be deleted)

Undeclared

- Always global

Variable Hoisting

A consequence of variables being processed before any code executed

Variables can be used “before” being declared

Best practice:

- Declare variables at top

javascript: variables

Variable names are case sensitive

Must start with a letter, dollar sign or underscore

Subsequent characters can be digits 0-9

No reserved javascript keywords allowed (https://developer.Mozilla.org/en/JavaScript/Reverence/Reserved_Words)

Best practice: start variable names with a-z

javascript: constants

A read-only named constant

Created with the const keyword

- const *name1* = *value1* [, *name2* = *value2* [, ... [, *nameN* = *valueN*]]];

Same naming rules as for variables

Constants cannot change value or be re-declared

Cannot use the same name as an existing function or variable

```
const g = 10.5;
```

javascript: assignment

the single equals sign (=) is the assignment operator:

- e.g., variable = expression;

expression on the rhs is evaluated and the variable name on the lhs represents the value

```
var a = 0;  // declare variable a having value 0
a  = 100+1;   // variable a now has value 101
a = "cat";    // variable a now has value "cat"
var b = 0, c = true, d = "atom";  // 3 variables
a = b;          // variable a now has value zero
```

javascript: block

a block statement is used to group one or more statements within braces {}

commonly used with control flow

- loops, if/else...

```
{
    statement_1;
    statement_2;
     …
    statement_n;
}
```

javascript: block

Javascript does not have block scope

- variables declared within a block are scoped to the containing function or script
- any assignment of values continues beyond the block itself
- **Unless you use let to declare**

```
var a = 1;
{
    var a = 5;
}
 // variable a is 5
```

Javascript: variables

Multiple variables can be declared in one var statement, separated by a comma:

- var a=0, b, c=100, d="blue sky", e = a;

This practice is more efficient than declaring each variable with a separate var

BUT, it is harder to maintain code that is like this!

Javascript: data types

Eight primitive data types:

- Boolean
- Null
- Undefined
- Number
- BigInt
- String
- Symbol

Javascript: numeric

An integer number is a sequence of digits

- Range: -2^53 to 2^53
- Base 10 integers don’t start with a zero
- Base 8 integers start with a 0 (deprecated)
- Base 16 integers start with 0x

```
var a = 0100; //a is 64

var b = 100; //b is 100

var c = 0x100; //c is 16


var d = 0x3a - 0200; //d is -70

var e = -073-0x0b; //e is -68
```

Javascript: numeric

Floating point literals

- Must have at least one digit and either a decimal point, or ‘e’
- Range is 5e-324 to 1.797e308
- Javascript keyword: Infinity or –Infinity
- Number.POSITIVE_INFINITY, Number.NEGATIVE_INFINITY, Number.MAX_VALUE,
- Number.MIN_VALUE

```
var a = 10.0101;

var b = -0.99;

var c = 1.45E10;

var d = 2e-2;

var bigNum = 2/0; //is infinity
```

Javascript: string

strings store a piece of text

JavaScript has 2 kinds of strings: primitives and objects

primitives: can use JavaScript String() or assignment:

- var txt = String(“Hello”);
- var txt = “Hello”;

objects: use new String()

- var txt = **new** String(“Hello”);

Use primitive form unless object form is needed

Javascript: string

JavaScript literal strings are *immutable*

- Cannot modify them after declared
- Characters within them cannot be changed
- There is no Javascript method or property that allows you to change the characters in a literal string

String Objects are mutable

Javascript: string

string length displayed using length method

- var txt_len = “hello”.length;

Empty string “” has a length of zero

Special characters such as “ ‘ \ and *backspace*, *newline, tab, carriage return*, can be defined within a string:

- “\b” “\”” ‘\’ ’ “\\” “\n” “\t” “\r”
- var t = “He said, \”Welcome\”.”;

Javascript: string

Concatenation operators are + and +=

- “Welcome to “ + “my house” makes: “Welcome to my house”
- welcome+= “ Thank-you.” adds the string “Thank-you.” to the end of the string variable named welcome
  - can only be used on String objects (mutable)

Also: string1.concat(string2) method

- returns a new string with string1 concatenated onto string2

```
var n = “abc”;

var t = n.concat(“xyz”); // t is “abcxyz”, n is “abc”
```

Javascript: string

To access individual characters within a string in two ways

- Using charAt method
  - “mouse”.charAt(1) is “o”
- As an array (first char is index zero)
  - “mouse”[1] is “o”
    - yes. You can do this!

```
var pet = “mouse”;
var c = pet.charAt(1); //c is “o”
c = pet[1]; //c is “o”
```

Javascript: string

substr method returns a portion of a string

- string.substr(start_index, length)
  - Length is option, but if not provided extract characters until end of string

```
var answer = “quick”;

var n1 = answer.substr(1,2); //ui

var n2 = answer.substr(2); //ick

var n3 = answer.substr(-1); //k
```

Javascript: string

Replace method substitutes one substring with another

- string.replace(search_string, new_string);

```
var t = “white car with white seat”;

var n = t.replace(“white”, “blue”);

var p = t.replace(/white/g, “red”);

//n is “blue car with white seat”

//p is “red car with red seat”

//t is “white car with white seat”
```

Javascript: string

toLowerCase and toUpperCase converts the string's case

- These two methods require no arguments

```
var city = "Victoria, BC";

var n1 = city.toLowerCase();

// victoria, bc

var n2 = city.toUpperCase();

//VICTORIA, BC

//city is still: Victoria, BC"
```

Javascript: string

string “null” is not the same as null

string “undefined” is not the same as undefined

string “” is not the same as null or undefined

Javascript: boolean

boolean values are either true or false

double-equals operator (==)

- tests if two operands represent the same value (but not the same type!)

triple-equals operator (===)

- tests if two operands represent the same value **and** have the same type

non-zero numeric values equate to true

null, undefined, NaN, and “” evaluate to false

```
var a = true;

var b = false;

var c = (1==1); //c is true

var d = (a = 2); //d is true, a is 2)

var e = (1==“1”);// e is true

var f = (1 === “1”); // f is false
```

javascript: typing

JavaScript is a *dynamically typed* programming language

- variables are not defined by data type at declaration but by their values (or 'literals')

the type of a literal is defined based on context (run-time)

when combining literals of different types, the first type is used

Java and C are *statically typed* – the type of the variable is set at compile time permanently

Javascript: typeof

the `typeof` operator is unary

- use of () optional
- `typeof( "pumpkin" )`
- `typeof( 563 )`
- `typeof( true)`
- `typeof( null )`
- `typeof "squash"`
- returns **type** of the operand: `"number", "string", "boolean", "object", "function", undefined, "xml"`

Javascript: dynamic typing

```
var a = 99;

var b = "Ninety nine";

var c = 100 + 100;  // c is 200

var d = ( a < 100 );  // d is true

var e = d && (c > 100); // e is true

a = e;    // a is true

var f = “100” + 10; // f is 10010

var g = “100” – 10; // g is 90
```

Javascript: weak typing

JavaScript is also weakly typed

- no restrictions on use of operators (such as the plus sign) involving values of different data types

JavaScript rule:

- when you use + with a number and a string in any order you get a string result

```
var a = 100;

var b = "+100";

var sum = a + b;    // sum is "100+100"
  not 200

sum = parseInt(a) + parseInt(b);  //
  sum is 200
```

javascript: casting

JavaScript data type examples

- `"Count to " + 10 is  "Count to 10"`
- ` and  2.5 + "10" is "2.510"`

`parseInt()` and `parseFloat()`

- `parseInt( "12" )` returns the integer `12`
- `parseFloat( "33.23" )` returns `33.23`
- `parseInt( "23.66")` returns `23`
- `parseInt( undefined )` and `parseInt(null)` returns `NaN` (not a number)
- optional second argument is the radix (10 is default, 16, or 8 but that is deprecated) `parseInt("0xaa", 16)` is 170 decimal.

see http://jsfiddle.net/Stevelang/vpenh/

```
<script type="text/javascript">


      var answer = 99;
      answer = "Ninety nine ”;
      var question = "What is 9 times 11? " + answer;
      document.write(question + "<br />");
      question = answer + " is 9 times what number?”;
      document.write(question);

</script>
```

javascript: expressions

expressions in JavaScript come in four types

- assignment which assigns a value to a variable
- arithmetic evaluates to a number
- string evaluates to a string
- logical evaluates to a boolean value (true or false)

use the keyword `var/let` to declare a variable and optionally assign it an initial value

a variable declared using `var/let` with no initial value has the value `undefined`

Javascript: assignment

```
var x = 10;
var y = 5;

x += y;  // x is now 15 (10 + 5)

x *= y;  // x is now 75 (15 * 5)

x /= y;  // x is now 15 (75 / 5)

x %= y;  // x is now 0  (15 / 5 leaves 0
         //              remainder )
```

Javascript: assignment

```
var x = 10;
var y = 5;
var z;

x++;  // increment operator; x is now 11

y--;  // decrement operator; y is now 4

z = ++y;  // z is 5 and y is now 5
z = x--;  // z is 11 and x is now 10
```

JavaScript: Comparison Operators

Equality Operators:

- == (equality)
- === (identity)

Inequality Operators:

- != (inequality)
- !== (non-identity)

Comparison Operators:

- <, <=
- >, >=

Javascript: logical

And:
- &&

Or:
- ||

Not:
- !

Truthy and Falsy:
- What gets converted to true or false?

Short-circuiting

javascript: conditional

ternary operator as in C, C++

```
(expression) ? value1 : value2;
```

- if (*expression*) evaluates true, then value1 is returned; otherwise, value2 is returned

```
var a = ( 3 == 4 ) ? "y" : "n";    // a is "n"
```

Demo

prompt to ask: What is 4+5?

check their answer

- if wrong, show some amazing art
- if right tell them they’re awesome! (with a pic)

Errors

error.html example

More practice

do w3schools tutorials: Introduction, Where To, Output, Syntax

MDN Tutorial: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Introduction