# CS 146: Intro to Web Programming and Project Development

*Instructor: Iraklis Tsekourakis*

*Lieb 213*

Email: itsekour@stevens.edu


# JavaScript

# JavaScript Statements

- In HTML, JavaScript statements are "instructions" to be "executed" by the web browser
- The following statement tells the browser to write ”I am Hercules” inside an HTML element with id=”herc":

```
<script>
document.getElementById("herc").innerHTML = "I am Hercules";
</script>
```

# JavaScript Statements Example

- Most JavaScript programs contain many JavaScript statements
- The statements are executed, one by one, in the same order as they are written
- Semicolons separate JavaScript statements

```
var x = 5;
var y = 6;
var z = x + y;
document.getElementById("demo").innerHTML = z;
```

- *JavaScript ignores multiple spaces; you can add white space to your script to make it more readable*

# JavaScript Keywords

| Keyword | Description |
|---|---|
| break | Terminates a switch or a loop |
| continue | Jumps out of a loop and starts at the top |
| debugger | Stops the execution of JavaScript, and calls (if available) the debugging function |
| do ... while | Executes a block of statements, and repeats the block, while a condition is true |
| for | Marks a block of statements to be executed, as long as a condition is true |
| function | Declares a function |
| if ... else | Marks a block of statements to be executed, depending on a condition |
| return | Exits a function |
| switch | Marks a block of statements to be executed, depending on different cases |
| try ... catch | Implements error handling to a block of statements |
| var | Declares a variable |

# JavaScript Comments

- Single line comments: //one-liner
- Multi-line comments: /* multi-liner ….. */
  - It is most common to use single line comments
  - Block comments are often used for formal documentation
- Use comments to prevent execution!

# JavaScript Variables

- While JS has variables that can hold a given type, the type of a variable can change throughout its lifespan
  - A var holding an *int* could become a *string,* an *object,* an *array*, etc…
- Because of their changeable nature, you do not need to declare the type of a variable, however, you should use the keyword *var* to declare a variable!
  - var exampleVar;
- You can also use the pre/post increment/decrement like in other languages
  - var++ or --var

# JavaScript Variables

- It's a good programming practice to declare all variables at the beginning of a script
- One statement, many variables

```
var person = "John Doe", carName = "Volvo", price = 200;
```

- Re-declaring JS variables: will it lose its value?

```
var carName = "Volvo";
var carName;
```

- What happens when you put a number in quotes? JS happens!

```
var x = "5" + 2 + 3;
```

# JavaScript Arithmetic Operators

| Operator | Description |
|---|---|
| + | Addition |
| - | Subtraction |
| * | Multiplication |
| / | Division |
| % | Modulus |
| ++ | Increment |
| -- | Decrement |

# JavaScript Assignment Operators

| Operator | Example | Same As |
|---|---|---|
| = | x = y | x = y |
| += | x += y | x = x + y |
| -= | x -= y | x = x - y |
| *= | x *= y | x = x * y |
| /= | x /= y | x = x / y |
| %= | x %= y | x = x % y |

# JavaScript String Operators

- When used on strings, the + operator is called the concatenation operator

```
txt1 = "John";
txt2 = "Doe";
txt3 = txt1 + " " + txt2;
```

```
txt1 = "What a very ";
txt1 += "nice day";
```

```
x = 5 + 5;
y = "5" + 5;
z = "Hello" + 5;
```

# JavaScript Comparison Operators

| Operator | Description |
|---|---|
| == | equal to |
| === | equal value and equal type |
| != | not equal |
| !== | not equal value or not equal type |
| > | greater than |
| < | less than |
| >= | greater than or equal to |
| <= | less than or equal to |
| ? | ternary operator |

**JavaScript** provides **three** different value-comparison operations: strict equality (or "**triple equals**" or "identity") using ===, loose equality ("double **equals**") using ==, and Object.is

# JavaScript Logical Operators

| Operator | Description |
|---|---|
| && | logical and |
| \|\| | logical or |
| ! | logical not |

# JavaScript Program Flow

- JavaScript is interpreted and executed at the same point as the rest of your HTML, which means that if you embed JS in your HTML and your code contains a blocking method (like alert or prompt), the rest of your HTML will not be able to load until you pass that block
- It is typically better to write methods and call them when certain events happen (more on events later on)
- If you have many methods, it is cleaner to put them all inside a .js file and include it

# JS: Getting Input

- Besides Forms, it is possible to prompt the user for input
- This can be extremely annoying to the user, so only use it for testing purposes!
- var value = prompt(”Informative Message”, ”Default Value”);
- e.g.: var val(“How old are you?”, 18);
- Once you accept, the value you entered will be stored in *val*
- Note that by default, things are returned as a string but it’s possible to parse to numeric values

# JS: Parsing Strings into Numbers

- If a string begins with a number, it is possible to extract that number and ignore the rest
  - Note that this only works if the number is at the beginning
- You can use *parseInt(value)* or *parseFloat(value)*
  - *parseInt* will read digits until the first non-digit character and return an *int*
  - *parseFloat* will read digits or a single period sign (.) until the first non-digit or second period sign and return a floating point number
- If no number can be parsed, both methods return NaN
  - You can check if something is not a number using isNaN(var)
  - How do we check if it is a number?

# Exercise: Area of Triangle

A method for calculating the area of a triangle when you know the lengths of all three sides.

Let a,b,c be the lengths of the sides of a triangle. The area is given by:

$$Area = \sqrt{p(p-a)(p-b)(p-c)}$$

where p is half the perimeter, or $\frac{a+b+c}{2}$