CS1520 Recitation
Week 2

Javascript 1
Syntax, Variable, and Equality

http://cs.pitt.edu/~jlee/cs1520

Jeongmin Lee, (jlee@cs.pitt.edu)

Today

- Embed code
- Syntax
- Declare variable
- Numeric, String, Objects
- Control statements
- == vs. ===

NOTE:
Many examples are based on w3schools.com, javascript.info, and tutorialspoint.com

Review of syntax

Embed code of javascript

- 3 Ways to include
  - (In body section)
  - `<script>//directly write code here</script>`
  - `<script src="path/to/file.js"></script>`
  - `<script src="http://www.scripts.com/file.js"></script>`

Embed code of javascript

```
<!DOCTYPE HTML>
<html>

<body>

  <p>Before the script...</p>

  <script>
    alert( 'Hello, world!' );
  </script>

  <p>...After the script.</p>

</body>

</html>
```

Embed code of javascript

```
<!DOCTYPE HTML>
<html>

<body>

  <p>Before the script...</p>

  <script src="hello.js"> </script>

  <p>...After the script.</p>

</body>

</html>
```

hello.html

```
alert(“hello world!”);
```

hello.js

Syntax

- Identifier should begin with a letter or underscore (numbers cannot be the first character)
- Case sensitive
- Comments: // and /* .. */
- Semicolons are optional to use (but preferred)

Declare variable

- Variables can be implicitly or explicitly declared.
- Starts with "var" for variable
- (NOTE: in modern JS, you can use "let" instead of var)

    - var myNum = 0;
    - var myString = 'Hello world';

    - var myArray = []; //
      equiv. to var myArray = new Array();

    - var myObj = {}; //
      var myObj = new Object();

Declare variable

```
<!DOCTYPE HTML>
<html>

<body>

  <p>Before the script...</p>

  <script>
    var msg = ‘hello world!’;
    alert(msg);
  </script>

  <p>...After the script.</p>

</body>

</html>
```

Declare variable

- For constant typed variable, use “const”
    - const fixed_num = 0;
    - fixed_num = 999; // error!

Numeric operators

- Numeric operators - ++, --, +, -, *, /, %
  - % : remainder
  - ** : exponentiation
  - ++, -- : increment / decrement
  - Prefix (++a) returns new value
  - Postfix (a++) returns old value while value is increased (try!!)

- All operations are in double precision

- The Math Object provides floor, round, max, min, trig functions, etc. e.g., Math.cos(x)
- The Number Object
- Some useful properties:
  - MAX_VALUE, MIN_VALUE, NaN,
  - POSITIVE_INFINITY, NEGATIVE_INFINITY, PI
  - e.g., Number.MAX_VALUE

String

- A string in JavaScript must be surrounded by quotes:
    - `let str = "Hello";`
    - `let str2 = 'Single quotes are ok too';`
- You can embed another string with ${str};
    - `let phrase = \`can embed ${str}\`;`

String operators

- Concatenation e.g, var str1 = ‘hello’ + ‘ ‘ + ‘world’;
- length e.g., var len = str1.length; (a property)
- charAt(position) e.g., str.charAt(3)
- indexOf(string) e.g., str.indexOf('B')
- substring(from, to) e.g., str.substring(1, 3)
- toLowerCase() e.g., str.toLowerCase()

Boolean

- The boolean type has only two values: true and false.
    - `var has_dinner = false;`
    - `var is_boring = true;`

Null

- null value does not belong to any of the type:
    - `var cup = null;`

Control

Interaction

- alert(message)
  - Open a box on the browser with the message

- result = prompt(“question”, “default_value”);
  - Open a box on the browser to get user input
  - Default value will be save when no user input typed

- result = confirm(“question”);
  - Open a box on the browser with “Okay” and “Cancel” box, to get boolean input from user

```
var x = prompt("your favorite color?", "white");
alert(x);

var y = confirm("did you get dinner?")
alert(y);
```

Control statements

- If / Else if / else

```
var year = prompt('When Pitt Established?');

if (year < 1787) {
  alert( 'Too old...' );
} else if (year > 1787) {
  alert( 'Too fresh' );
} else {
  alert( 'Exactly!' );
}
```

Control statements

- **Control expressions – three kinds**
  - Primitive values (True or False)
  - If it is a string, it is true unless it is empty or "0"
  - If it is a number, it is true unless it is zero

Control statements

- Relational Expressions
  - The usual six: ==, !=, <, >, <=, >=
  - Operands are coerced if necessary
    - If one is a string and one is a number, it attempts to convert the string to a number
  - If one is Boolean and the other is not, the Boolean operand is coerced to a number (1 or 0)

- Compound Expressions
  - The usual operators: &&, ||, and !

Control statements

- Switch

```
switch (expression) {
  case value_1:
  // value_1 statements
  case value_2:
  // value_2 statements
  …
  [default:  // default statements]
}
```

- The statements can be either statement sequences or compound statements
- The control expression can be a number, a string, or a Boolean
- Different cases can have values of different types

Control statements

```
var a = 2 + 2;

switch (a) {
  case 3:
    alert( 'Too small' );
    break;
  case 4:
    alert( 'Exactly!' );
    break;
  case 5:
    alert( 'Too large' );
    break;
  default:
    alert( "I don't know such values" );
}
```

Control statements

- Loop statements
  - **while** (control_expression) statement or compound
  - **for** (init; control; increment) statement or compound
    init can have declarations, but the scope of such variables is the whole script
  - **do** {statement or compound}
    while (control_expression)

Control statements

```
while (condition) {
  // code
  // so-called "loop body"
}
```

```
let i = 0;
while (i < 3) { // shows 0, then 1, then 2
  alert( i );
  i++
}
```

Control statements

```
do {
  // loop body
} while (condition);
```

```
let i = 0;
do {
  alert( i );
  i++;
} while (i < 3);
```

Control statements

```
for (begin; condition; step) {
  // ... loop body ...
}
```

```
for (let i = 0; i < 3; i++) {
// shows 0, then 1, then 2
  alert(i);
}
```

Breaking the loop: break

```
let sum = 0;

while (true) {
   let value = +prompt("Enter a number", '');
  if (!value) break; // (*)
  sum += value;
}

alert( 'Sum: ' + sum );
```

Skip (continue) to the next iteration: continue

```
for (let i = 0; i < 10; i++) {
  // if true, skip the remaining part of the body
  if (i % 2 == 0) continue;
  alert(i); // 1, then 3, 5, 7, 9
}
```

== VS. ===

== vs. ===

- JavaScript has both **strict (===, !==)** and **type-converting (==, !=)** equality comparison.
- For strict equality the **objects** being compared must have the same type.
- Two **strings** are **strictly equal** when they have the same sequence of characters, same length, and same characters in corresponding positions.
- Two **numbers** are **strictly equal** when they are numerically equal (have the same number value).
- NaN is not equal to anything, including NaN. Positive and negative zeros are equal to one another.

Source: MDC (https://developer.mozilla.org/en/Core_JavaScript_1.5_Reference/Operators/Comparison_Operators)

== VS. ===

- Two **Boolean** operands are strictly equal if both are true or both are false.
- Two **objects** are strictly equal if they refer to the same Object.
- **Null** and **Undefined types** are == (but not ===). [I.e. (Null==Undefined) is true but (Null===Undefined) is false]

Source: MDC (https://developer.mozilla.org/en/Core_JavaScript_1.5_Reference/Operators/Comparison_Operators)

== VS. ===

- console.log(3 == "3"); // true; type conversion happened
- console.log(3 === "3"); // false.

Source:
https://appendto.com/2016/02/vs-javascript-abstract-vs-strict-equality/

== VS. ===

- console.log(true == '1'); // true
- console.log(true === '1'); // false

Source:
https://appendto.com/2016/02/vs-javascript-abstract-vs-strict-equality/

== VS. ===

- console.log(undefined == null); // true

- console.log(undefined === null); // false. Undefined and null are distinct types and are not interchangeable.

Source:
https://appendto.com/2016/02/vs-javascript-abstract-vs-strict-equality/

== VS. ===

- console.log(true == 'true'); // false. A string will not be converted to a boolean and vice versa.

- console.log(true === 'true'); // false

Source:
https://appendto.com/2016/02/vs-javascript-abstract-vs-strict-equality/

== VS. ===

- console.log("This is a string." == new String("This is a string.")); // true

- console.log("This is a string." === new String("This is a string.")); // false

- Strings **literals** are different from string **objects**
    - console.log(typeof "This is a string."); // string
    - console.log(typeof new String("This is a string.")); //object

Source: https://appendto.com/2016/02/vs-javascript-abstract-vs-strict-equality/

== VS. ===

- var a = [];
- var b = [];
- var c = a;

- console.log(a == b); // false

- console.log(a === b); // false

  => When comparing reference types both abstract and strict comparisons will **return false unless** both operands refer to the **exact same object**

- console.log(a == c); // true

- console.log(a === c); // true

Source:
https://appendto.com/2016/02/vs-javascript-abstract-vs-strict-equality/

Questions?