Scripting rules of thumb

JavaScript is case sensitive. Identifiers as traditional, except can contain numbers and can begin with a $.

$ -- valid JavaScript identifier?
http://www.authenticsociety.com/blog/JavaScript_DollarSign

Camel casing is traditional.

salesTaxRate, idCode

Comments:

// for single line
/* */ for multi-line
Scripting rules of thumb

Statements should be terminated with a semicolon, even though parser will automatically put in if omitted. Why?

Code blocks use { }. Conventional in JavaScript to use even around single-line blocks (if statements) and to use Java-style formatting.

```javascript
if (condition) {
  do this;
}
```

Opening (or closing) brace goes on line with the statement that controls access to the code block.

Variables and typing

Variables are not explicitly typed, and type may be changed implicitly through assignment.

'var' placed in front of identifier to denote a variable declaration.

```javascript
var name;
var age = 17;
var gender, grade;
var shoeSize = 7, hairColor = "red";
```
Variable scoping rules.

Variable created with `var`--local scope (same code block) only.

Variables *can* be created without `var` (appear in assignment). Those variables have *global scope*.

- Hard to maintain.
- May cause name collision
- Don't do this!

Identifier types

6 JavaScript identifier types:

- `typeof` operator allows us to determine type of an in-scope identifier.

http://einstein.etsu.edu/~pittares/CSCI3110/examples/2-2.htm
Type conversion functions

—evaluates val and returns true or false.
  true: boolean true, nonempty string, nonzero number, any object
  false: boolean false, empty string, 0, NaN, null, undefined.

—performs numeric conversion on val
  boolean true returns 1, false returns 0.
  if val is numeric, returns val.
  if val is undefined, returns NaN.
  null string returns 0. If string is numeric equivalent returns that equivalent. If other returns NaN.

Type conversion functions

—returns val converted to an "integer"

If first non-whitespace character in val is not numeric or string is null, returns NaN.
Continues parsing val until reaches end or a non-numeric value. Returns that outcome.
If numeric substring begins 0x, interpreted as hexadecimal. If begins with 0, interpreted as octal.
Favored over Number since returns NaN on a null string val instead of 0.
An optional second parameter can specify the base of val. Returned value is always base 10.
http://einstein.etsu.edu/~pittares/CSCI3110/examples/2-3.htm
Type conversion functions

—returns val converted to a "float"

Similar to parseInt, except
  . is a valid part of a number, but can only occur once.
Zeros at the beginning are ignored.
If val in hexadecimal form ("0x...") 0 is returned.
If val in scientific notation form, float equivalent is returned. (2.7845e10)

  returns val as a string.
If val is null or undefined, "null" or "undefined" returned.
If val is number, optional parameter can specify base of output.
  var x = 10;
  var y = x.toString(2);