JavaScript Basics

Based on “jQuery Fundamentals” by Rebecca Murphey
http://github.com/rmurphey/jqfundamentals

Contents

1 JavaScript Basics

Home Smooth CoffeeScript

Formats Markdown PDF HTML

Overview

JavaScript is a rich and expressive language in its own right. This section covers the basic concepts of JavaScript, as well as some frequent pitfalls for people who have not used JavaScript before. While it will be of particular value to people with no programming experience, even people who have used other programming languages may benefit from learning about some of the peculiarities of JavaScript.

If you're interested in learning more about the JavaScript language, I highly recommend *JavaScript: The Good Parts* by Douglas Crockford.

Syntax Basics

Understanding statements, variable naming, whitespace, and other basic JavaScript syntax.

A simple variable declaration

```
var foo = 'hello world';
```

Whitespace has no meaning outside of quotation marks

```
var foo =         'hello world';
```

Parentheses indicate precedence

```
2 * 3 + 5;     // returns 11; multiplication happens first
2 * (3 + 5);   // returns 16; addition happens first
```

Tabs enhance readability, but have no special meaning

```
var foo = function() {
    console.log('hello');
};
```

Operators

Basic Operators

Basic operators allow you to manipulate values.

Concatenation

```
var foo = 'hello';

var bar = 'world';

console.log(foo + ' ' + bar); // 'hello world'
```

Multiplication and division

```
2 * 3;
2 / 3;
```

Incrementing and decrementing

```
var i = 1;

var j = ++i;  // pre-increment:  j equals 2; i equals 2
var k = i++;  // post-increment: k equals 2; i equals 3
```

Operations on Numbers & Strings

In JavaScript, numbers and strings will occasionally behave in ways you might not expect.

Addition vs. concatenation

```
var foo = 1;
var bar = '2';

console.log(foo + bar);  // 12. uh oh
```

Forcing a string to act as a number

```
var foo = 1;
var bar = '2';

// coerce the string to a number
console.log(foo + Number(bar));
```

The Number constructor, when called as a function (like above) will have the effect of casting its argument into a number. You could also use the unary plus operator, which does the same thing:

Forcing a string to act as a number (using the unary-plus operator)

```
console.log(foo + +bar);
```

Logical Operators

Logical operators allow you to evaluate a series of operands using AND and OR operations.

Logical AND and OR operators

```
var foo = 1;
var bar = 0;
var baz = 2;

foo || bar;   // returns 1, which is true
bar || foo;   // returns 1, which is true

foo && bar;   // returns 0, which is false
foo && baz;   // returns 2, which is true
baz && foo;   // returns 1, which is true
```

Though it may not be clear from the example, the `||` operator returns the value of the first truthy operand, or, in cases where neither operand is truthy, it'll return the last of both operands. The `&&` operator returns the value of the first false operand, or the value of the last operand if both operands are truthy.

Be sure to consult the section called "Truthy and Falsy Things" for more details on which values evaluate to `true` and which evaluate to `false`.

Note

You'll sometimes see developers use these logical operators for flow control instead of using `if` statements. For example:

```
// do something with foo if foo is truthy
foo && doSomething(foo);

// set bar to baz if baz is truthy;
// otherwise, set it to the return
// value of createBar()
var bar = baz || createBar();
```