Javascript, The Swiss Army Knife of Programming Languages

David Morcillo

23-11-2013

About me

- twitter.com/ultrayoshi
- github.com/ultrayoshi

Features

- Loosely typed language

Features

- Loosely typed language
- Object literal notation

Features

- Loosely typed language
- Object literal notation
- Prototypal inheritance

Features

- Loosely typed language
- Object literal notation
- Prototypal inheritance
- Global variables

Features

- Loosely typed language
- Object literal notation
- Prototypal inheritance
- Global variables
- Functions are first class objects

Features

ECMAScript

The standard that defines JavaScript is the third edition of *ECMAScript Programming Language*.

Hello World

index.html

```
<html>
  <head>
    <script>
      document.writeln(‘‘Hello, world!’’);
    </script>
  </head>
  <body>
  </body>
</html>
```

Syntax

Comments

Block comments formed with /* */ and line-ending comments starting with //. Example:

```
/*
  We are learning Javascript and comments are very important
*/
document.writeln(‘‘Hello World!’’); // Output: Hello World!
```

Syntax

Comments

Block comments formed with /* */ and line-ending comments starting with //. Example:

```
/*
  We are learning Javascript and comments are very important
*/
document.writeln(‘‘Hello World!’’); // Output: Hello World!
```

Names

Starts with a letter or underscore and optionally followed by on or more letters, digits or underscores. Beware of some reserved words.

```
bullet           // valid     _mana             // valid
3force           // invalid   lucky42           // valid
rocket-launcher  // invalid   grenade_launcher  // valid
```

Syntax

Numbers

Single number type represented internally as 64-bit floating point.

```
42
3.141516
10e5
1/0 // Output: Infinity
0/0 // Output: NaN
```

Syntax

Numbers

Single number type represented internally as 64-bit floating point.

```
42
3.141516
10e5
1/0 // Output: Infinity
0/0 // Output: NaN
```

Strings

Can be wrapped in single quotes or double quotes. It can contains 0 or more characters. All characters in Javascript are 16 bits wide.

```
‘‘Hello World’’
’Hello World’
‘‘This is\n a multiline string’’
’You can write ‘‘ on single quotes string’
```

Syntax

Functions

```
function helloWorld (name) {
    console.log('Hello ' + name + '!');
}

helloWorld('David'); // Output 'Hello David!'

var myFunction = function () {
    console.log('Hi there!');
};

myFunction(); // Output: 'Hi there!'
```

Syntax

Variables

Use the `var` keyword followed by a name to declare a variable. When used inside of a function, the `var` statement defines the function's private variables.

```
var player; // variable player declared on a global scope

function test() {
    var enemy; // Scoped to function test

    function test2() {
        var bullet;  // Scoped to function test2
    }
}
```

Syntax

if, else

```
var testOk = true;

if (testOk) {
    console.log(‘‘Captain obvious’’);
} else {
    console.log(‘‘I’m bored’’);
}
```

Here are the *falsy* values:

- `false`
- `null`
- `undefined`
- The empty string
- The number 0
- The number NaN

All other values are *truthy*.

Syntax

switch

```
var weapon = ‘‘rocketlauncher’’;

switch(weapon) {
    case ‘‘pistol’’:
        console.log(‘‘piu piu’’);
        break;
    case ‘‘shotgun’’:
        console.log(‘‘paaam!’’);
        break;
    case ‘‘rocketlauncher’’
        console.log(‘‘BOOOOM!’’);
        break;
    default:
        console.log(‘‘falcon punch!’’);
        break;
}
```

Syntax

while, do while

```
var counter = 0;
while (counter < 10) { // Ends when counter is equal to 10
    console.log(counter);
    counter += 1;
}

do {
    console.log(counter);
    i -= 1;
} while(counter > 0); // Ends when counter is equal to 0
```

Syntax

while, do while

```
var counter = 0;
while (counter < 10) { // Ends when counter is equal to 10
    console.log(counter);
    counter += 1;
}

do {
    console.log(counter);
    i -= 1;
} while(counter > 0); // Ends when counter is equal to 0
```

for

```
var i;

for (i = 0; i < 10; i += 1)
    console.log(i);
}
```

Objects

- Objects in Javascript are mutable keyed collections.

Objects

- Objects in Javascript are mutable keyed collections.
- Arrays, functions and regular expressions are objects.

Objects

- Objects in Javascript are mutable keyed collections.
- Arrays, functions and regular expressions are objects.
- A property name can be any string.

Objects

- Objects in Javascript are mutable keyed collections.
- Arrays, functions and regular expressions are objects.
- A property name can be any string.
- Objects can inherit properties of another through its prototype.

Objects

- Objects in Javascript are mutable keyed collections.
- Arrays, functions and regular expressions are objects.
- A property name can be any string.
- Objects can inherit properties of another through its prototype.

`Prototype`

All objects created from object literals are linked to `Object.prototype`.

Objects

- Objects in Javascript are mutable keyed collections.
- Arrays, functions and regular expressions are objects.
- A property name can be any string.
- Objects can inherit properties of another through its prototype.

`Prototype`

All objects created from object literals are linked to `Object.prototype`. If we try to retrieve a property value from an object, and if the object lacks the property name, then Javascript attempts to retrieve the property value from the prototype object.

Objects

Object.create

```
var soldier = {
    hp: 10,
    strength: 5,
    weapon: 'Pistol'
};

var knight = Object.create(soldier);
knight.weapon = 'Sword';
knight.shield = true;

console.log(knight.hp); // Output: 10
console.log(knight.weapon); // Output: 'Sword'
console.log(knight.shield); // Output: true
```

Visit http://www.objectplayground.com/ for a graphical explanation

Objects

hasOwnProperty

```
knight.hasOwnProperty('hp'); // Output: false
knight.hasOwnProperty('shield'); // Output: true
```

Objects

hasOwnProperty

```
knight.hasOwnProperty('hp'); // Output: false
knight.hasOwnProperty('shield'); // Output: true
```

for in

```
for (attr in knight) {
    if(knigth.hasOwnProperty(attr)) {
        console.log('Knight property ' + attr + ' with value ' +
            knight[attr]);
    }
}
// Output: Knight property shield with value true
```

Objects

hasOwnProperty

```
knight.hasOwnProperty('hp'); // Output: false
knight.hasOwnProperty('shield'); // Output: true
```

for in

```
for (attr in knight) {
    if(knigth.hasOwnProperty(attr)) {
        console.log('Knight property ' + attr + ' with value ' +
            knight[attr]);
    }
}
// Output: Knight property shield with value true
```

delete

```
console.log(knight.weapon); // Output: 'Sword'
delete knight.weapon;
console.log(knight.weapon); // Output: 'Pistol'
```

Functions