Objects

1) Object Literals
2) Retrieval
3) Update
4) Reference
5) Prototype
6) Reflection
7) Enumeration
8) Delete
9) Global Abatement
10) Javascript predefined objects

Almost "everything" in JavaScript can be objects. Strings, Dates, Arrays, Functions....

You can also create your own objects.

```
<script>
var person=new Object();
person.firstname="John";
person.age = "22";
document.write(person.firstname + " is " + person.age + " years old.");
</script>
```

The simple types of JavaScript are numbers, strings, booleans (true and false), null, and undefined. All other values are objects. Numbers, strings, and booleans are object-like in that they have methods, but they are immutable. Objects in JavaScript are mutable keyed collections. In JavaScript, arrays are objects, functions are objects, regular expressions are objects, and, of course, objects are objects.

An object is a container of properties, where a property has a name and a value. A property name can be any string, including the empty string. A property value can be any JavaScript value except for undefined.

Objects in JavaScript are class-free. There is no constraint on the names of new properties or on the values of properties. Objects are useful for collecting and organizing data. Objects can contain other objects, so they can easily represent tree or graph structures.

JavaScript includes a prototype linkage feature that allows one object to inherit the properties of another. When used well, this can reduce object initialization time and memory consumption.

1. Object Literals

```
var empty_object = {};

var stooge = {
"first-name": "Jerome",
```

```
"last-name": "Howard"
};

var flight = {
airline: "Oceanic",
number: 815,
departure: {
IATA: "SYD",
time: "2004-09-22 14:55",
city: "Sydney"
},
arrival: {
IATA: "LAX",
time: "2004-09-23 10:42",
city: "Los Angeles"
}
};
```

2. Retrieval

```
stooge["first-name"] // "Joe"
flight.departure.IATA // "SYD"
```

The undefined value is produced if an attempt is made to retrieve a nonexistent member:

```
stooge["middle-name"] // undefined
flight.status // undefined
stooge["FIRST-NAME"] // undefined
```

3. Update

```
stooge['first-name'] = 'Jerome';
```

If the object does not already have that property name, the object is augmented:

```
stooge['middle-name'] = 'Lester';
stooge.nickname = 'Curly';
flight.equipment = {
model: 'Boeing 777'
};
flight.status = 'overdue';
```

4. Reference

Objects are passed around by reference. They are never copied:

```
var x = stooge;
x.nickname = 'Curly';
var nick = stooge.nickname;
// nick is 'Curly' because x and stooge
```

```
// are references to the same object
var a = {}, b = {}, c = {};
// a, b, and c each refer to a
// different empty object
a = b = c = {};
// a, b, and c all refer to
// the same empty object
```

5. Prototype

The prototype property allows you to add properties and methods to an object.

object.prototype.name=value

```
<p id="demo">Click the button to add a property to an object.</p>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction()
{

function employee(name,jobtitle,born)
{
this.name=name;
this.jobtitle=jobtitle;
this.born=born;
}

var fred=new employee("Fred Flintstone","Caveman",1970);
employee.prototype.salary=null;
fred.salary=20000;

document.write(fred.salary);
// object.prototype.name=value
}
</script>

or

function add(x,y){
 return x+y;
}
employee.prototype.salary=add(2,2);
```

6. Reflection

```
typeof flight.number // 'number'
typeof flight.status // 'string'
typeof flight.arrival // 'object'
```

```
typeof flight.manifest // 'undefined'

typeof flight.toString // 'function'
typeof flight.constructor // 'function'

flight.hasOwnProperty('number') // true
flight.hasOwnProperty('constructor') // false
```

7. Enumeration

```
<script>
var stooge = {
"first-name": "Jerome",
"last-name": "Howard"
};

for (name in stooge) {
    document.writeln(name);
}
</script>
```

8. Delete

The delete operator can be used to remove a property from an object. It will remove a property from the object if it has one.

```
delete another_stooge.nickname;
another_stooge.nickname // 'Curly'
```

9. Global Abatement

JavaScript makes it easy to define global variables that can hold all of the assets of your application. Unfortunately, global variables weaken the resiliency of programs and should be avoided.

One way to minimize the use of global variables is to create a single global variable for your application:

```
var MYAPP = {};
```

That variable then becomes the container for your application:

```
MYAPP.stooge = {
"first-name": "Joe",
"last-name": "Howard"
};
MYAPP.flight = {
airline: "Oceanic",
number: 815,
departure: {
IATA: "SYD",
```

```
time: "2004-09-22 14:55",
city: "Sydney"
},
arrival: {
IATA: "LAX",
time: "2004-09-23 10:42",
city: "Los Angeles"
}
};
```

10. Javascript predefined objects

1) The instanceof Operator
2) The Array Object
3) The String Object
4) The Math Object
5) The Number Object
6) The Boolean Object
7) The Date Object