and undefined

The special values variable type.

JavaScript is a weakly typed language – i.e. a simple boolean:
number:
string:
functions.
to that function. Local variables are visible to all nested
in a new global variable of that name being created.

keyword.
keyword as a variable or function name obliterates that
name length. Names are case-sensitive.
character  cannot  be  numeric.  Many  extended  ASCII

//implicit global variable creation
}

//Local variable declarations – visible in nested functions

var

function

innerFuncA(){

var

function

var

innerFuncB(){

var

//Variables local to innerFuncA

}/your code here
}

aName='ExplainThat!';

//Implicit global variable creation

//your code here

}]

Nomenclature Rules

Function and variable names can consist of any
alphanumeric character. $ and _ are allowed. The first
character cannot be numeric. Many extended ASCII
characters are allowed. There is no practical limit on
name length. Names are case-sensitive.

If two or more variables or functions or a variable & a
function are declared with the same name the last
declaration obliterates all previous ones. Using a
keyword as a variable or function name obliterates that
keyword.

Visibility & Scope

Assignments without the use of the var keyword result
in a new global variable of that name being created.

Variables declared with the var keyword outwith the
body of a function are global. Variables declared with the
var keyword inside the body of a function are local to
that function. Local variables are visible to all nested
functions.

Local entities hide globals bearing the same name.

Variable Types

string: var s = 'explainthat' or "explainthat"
number: var n = 3.14159, 100, 0,...
boolean: var flag = false or true
object: var d = new Date();
function: var Greet = function sayHello(){alert('Hello')}

JavaScript is a weakly typed language – i.e. a simple
assignment is sufficient to change the variable type. The
typeof keyboard can be used to check the current
variable type.

Special Values

The special values false, infinity, NaN, null, true &
undefined are recognized. null is an object. infinity
and NaN are numbers.

Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Example</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>3 + 2</td>
<td>5 explainthat</td>
</tr>
<tr>
<td>-</td>
<td>3 - 2</td>
<td>-1</td>
</tr>
<tr>
<td>*</td>
<td>3*2</td>
<td>6</td>
</tr>
</tbody>
</table>

slice – returns array slice. 1st arg is start position. 2nd arg is
last position + 1
sort – alphanumeric sort if no argument. Pass sort
function as argument for more specificity.
slice – discard and replace elements
unshift – append elements to start & return new length
Date Object

get#

getUTC#

set#

setUTC#

where # is one of Date, Day, FullYear, Hours,
Milliseconds, Minutes, Month, Seconds, Time,
TimezoneOffset
toDateString – the date in English.
toGMTString – the date & time in English.
toLocaleDateString – the date in the locale language.
toLocaleString – date & time in the locale language.
toLocaleTimeString – time in the locale language.
toTimeString – time in English.
toUTCString – date & time in UTC, English

valueOf – milliseconds since midnight 01 January 1970, UTC

Math Object

E, LN10, LN2, LOG10E, LOG2E, PI,
SQR1_2, SQR2

abs – absolute value

#(n) – trigonometric functions

a#(n) – inverse trigonometric functions

where # is one of cos, sin or tan
ceil(n) – smallest whole number >= n
exp(n) – returns e
floor(n) – biggest whole number <= n
log(n) – logarithm of n to the base e
max(n,n2) – bigger of n and n2
min(n,n2) – smaller of n and n2
pow(a,b) – a
random – random number between 0 and 1
round(n) – n rounded down to closest integer
sqrt(n) – square root of n

Number Object

MAX_VALUE – ca 1.7977E+308
MIN_VALUE – ca 5E-324
NEGATIVE_INFINITY, POSITIVE_INFINITY

n.toExponential(m) – n in scientific notation with m
decimal places.
n.toFixed() – n rounded to the closest whole number.
n.toPrecision(m) – n rounded to m figures.
Hexadecimal numbers are designated with the prefix
0x or 0X, e.g. 0x2F is the number 47.

String Object

length – number of characters in the string

s.charAt(n) – returns s[n]. n starts at 0
s.charCodeAt(n) – Unicode value of s[n]
s.fromCharCode(n) – string built from Unicode

values n, n...
s1.indexOf(s2,n) – location of s2 in s1 starting at position n

s1.lastIndexOf(s2) – location of s2 in s1 starting from

the end

s.substring(n) – returns substring starting from m upto
character preceding n. No n = extract till end. n < 0 =
extract from end.
s.toLowerCase() – returns s in lower case characters
s.toUpperCase() – care to guess?

Array Object

length – number of elements in the array

concat – concatenates argument, returns new array.
join – returns elements as a string separated by
argument (default is ,)
pop – suppress & return last element
push – adds new elements to end of array & returns
new length.
reverse – inverts new elements to end of array & returns
new length.
shift – suppress & return first element

Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Example</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>3 + 2</td>
<td>5 explainthat</td>
</tr>
<tr>
<td>-</td>
<td>3 - 2</td>
<td>-1</td>
</tr>
<tr>
<td>*</td>
<td>3*2</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operator</th>
<th>Example</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>3 - 2</td>
<td>-1</td>
</tr>
<tr>
<td>*</td>
<td>3*2</td>
<td>6</td>
</tr>
</tbody>
</table>
**JavaScript Quick Reference Card**

### Escape Sequences

- \n - new line
- \r - carriage return
- \t - tab character
- \' - apostrophe
- " - quote
- \uNNNN - Unicode character at NNNN

Example: `\u25BA` gives the character ▲

### JavaScript in HTML

#### External JavaScript

```html
<script type="text/javascript" defer="defer" src="/scripts/explainthat.js"></script>
```

#### Inline JavaScript

```html
<script type="text/javascript">"your code here"</script>
```

### Comments

- Single line comment: `// Simple, single line, comment`  
- Multiline comment: `/* Comments spanning multiple lines */`

#### Conditional Execution

```javascript
if (Condition) CodetTrue; else CodetFalse
```

#### Error Handling

Method 1: The onerror event

```javascript
function whenError(msg,url,lineNo){
    //use parameters to provide meaningful messages
}
window.onerror = whenError;
```

Place this code in a separate `<script>` pair to trap errors occurring in other scripts. This technique blocks errors without taking corrective action.

Method 2: The try..catch..finally statement

```javascript
function showLogValue(num){
    var s = "No Error";
    try{
        if (num < 0) throw "badnum";
        if (num == 0) throw "zero";
    } catch (err) {
        s = err;
    } finally{ alert(s,Math.log(num))}]
    }
}
```

The finally block is optional. The two techniques can be used in concert.

### Looping

```javascript
whileLoop(num){
    while (num > 0) {
        alert(num);
        num--;
    }
}
```

- `doLoop(num){
  do{
    alert(num);
    num--;
  }while (num > 0);
}
```

### Document Object

- `body` - the body of the document
- `cookie` - read/write the document cookies
- `domain` - where was the document served from?
- `forms[]` - array of all forms in the document
- `images[]` - array of all images in the document
- `referrer` - who pointed to this document?
- `URL` - the URL for the document
- `getElementsByTagName(t)` - getElementsByName(n) - array of elements named n
- `getElementById(id)` - array of tagged elements
- `write` - write plain or HTML text to the document
- `onload` - occurs when the document is loaded
- `onunload` - occurs when user browses away, tab is closed etc.

### Element Object

- By element we mean any HTML element retrieved using the document.getElementById methods.
- `attributes` - all element attributes in an array
- `className` - the CSS style assigned to the element
- `id` - the id assigned to the element
- `innerHTMLHTML` - HTML content of the element
- `innerText` - content of the element shorn of all HTML tags. Does not work in Firefox
- `offsetHeight` - element dimensions (# = Height/Width) or location(# = Left/Right) in pixels

### Notes

- `alert(msg)` - displays a dialog with `msg`
- `clearInterval(id)` - clears interval `id` set by setInterval
- `clearTimeout(id)` - clears timeout `id` set by setTimeout
- `confirm(msg)` - shows a confirmation dialog
- `print()` - prints the window contents
- `prompt(msg,[default])` - shows prompt dialog, optionally with default content. Returns content or null.
- `replace(url)` - rechanges the current document with the tag at url. Destroys document entry in browser history.

### Color Coding

- **red** - Option `object` - JavaScript DOM object
- **green** - only numeric values
- **blue** - JavaScript Keywords
- **magenta** - object methods

Tested with Internet Explorer 6+, Firefox 1.5+ & Opera 9.1+.

If you find this reference card useful please help us by creating links to our site [http://www.explainthat.at](http://www.explainthat.at) where you will find other quick reference cards and many other free programming resources.