### Code Structure

JavaScript is a weakly typed language – i.e. a simple boolean:

- `true`
- `false`

Variables declared with the `var` keyword result in a new global variable of that name being created. Variables declared with the same name the last declaration obliterates all previous ones. Using a function are declared with the same name the last name length. Names are case-sensitive.

- `3*2` is 6
- `3 + 2` is 5
- `3 - 2` is 1

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.

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**

- `undefined` is the value 2
- `null` is 2
- `true` is 1
- `false` is 0

Special Values

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

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

### Math Object

- `Math` is the number 2
- `Date` is the number 2
- `Number` is the number 2

**Number Object**

- `Number.MAX_VALUE` is the number 2
- `Number.MIN_VALUE` is the number 2
- `Number.NEGATIVE_INFINITY` is the number 2
- `Number.POSITIVE_INFINITY` is the number 2

**String Object**

- `String.length` is the number 2
- `String.toLowerCase` is the number 2
- `String.toUpperCase` is the number 2

### Operators

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

JavaScript Quick Reference Card

---

**Code Structure**

```javascript
//Implicit global variable creation
{name} = "ExplainThat!";
//Implicit global variable creation

```

**Nomenclature Rules**

FUNCTION AND 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.

**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**

- `undefined` is an object.
- `null` is a property.
- `true` is a boolean.

**Special Values**

- `false`, `infinite`, `NaN`, `null`, `true`, and `undefined` are recognized.
- `null` is an object.

**Operators**

- `+` is addition.
- `-` is subtraction.
- `*` is multiplication.
- `/` is division.
- `%` is modulus.
- `&&` is logical AND.
- `||` is logical OR.
- `!` is logical NOT.

**Internal Functions**

- `toFixed()` returns a number rounded to the closest integer.
- `toExponential()` returns a number in scientific notation with `m` decimal places.
- `toPrecision()` returns a number rounded to `n` figures.

**String Object**

- `length` returns the number of characters in the string.
- `toLowerCase` returns a string in lower case characters.

**Number Object**

- `MAX_VALUE` is a constant.
- `MIN_VALUE` is a constant.
- `NEGATIVE_INFINITY` is a constant.
- `POSITIVE_INFINITY` is a constant.

**Date Object**

- `toDateString()` returns the date in the locale language.
- `toDateString()` returns the date in the locale language.
- `toTimeString()` returns the time in the locale language.

**Math Object**

- `Math.PI` is the constant.
- `Math.E` is the constant.
- `Math.LN10` is the constant.
- `Math.LOG2E` is the constant.
- `Math.SQRT2` is the constant.

**Operators**

- `+` is addition.
- `-` is subtraction.
- `*` is multiplication.
- `/` is division.
- `%` is modulus.
- `&&` is logical AND.
- `||` is logical OR.
- `!` is logical NOT.

**Internal Functions**

- `toFixed()` returns a number rounded to the closest integer.
- `toExponential()` returns a number in scientific notation with `m` decimal places.
- `toPrecision()` returns a number rounded to `n` figures.

**String Object**

- `length` returns the number of characters in the string.
- `toLowerCase` returns a string in lower case characters.

**Number Object**

- `MAX_VALUE` is a constant.
- `MIN_VALUE` is a constant.
- `NEGATIVE_INFINITY` is a constant.
- `POSITIVE_INFINITY` is a constant.

- `Math.PI` is the constant.
- `Math.E` is the constant.
- `Math.LN10` is the constant.
- `Math.LOG2E` is the constant.
- `Math.SQRT2` is the constant.

- `toFixed()` returns a number rounded to the closest integer.
- `toExponential()` returns a number in scientific notation with `m` decimal places.
- `toPrecision()` returns a number rounded to `n` figures.

- `length` returns the number of characters in the string.
- `toLowerCase` returns a string in lower case characters.

**Operators**

- `+` is addition.
- `-` is subtraction.
- `*` is multiplication.
- `/` is division.
- `%` is modulus.
- `&&` is logical AND.
- `||` is logical OR.
- `!` is logical NOT.

**Internal Functions**

- `toFixed()` returns a number rounded to the closest integer.
- `toExponential()` returns a number in scientific notation with `m` decimal places.
- `toPrecision()` returns a number rounded to `n` figures.

**String Object**

- `length` returns the number of characters in the string.
- `toLowerCase` returns a string in lower case characters.
### External JavaScript

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

### Inline JavaScript

```javascript
//use parameters to provide meaningful messages
```

### Comments

/* Comments spanning multiple lines */
// Simple, single line, comment

### Conditional Execution

```javascript
if (Condition) CodellTrue; else CodellFalse
```

### Error Handling

```javascript
function whenError(msg,url,lineNo){
    //use parameters to provide meaningful messages
}
```

### Looping

```javascript
for (var i = 0; i < num; i++) { ...
```

```javascript
while (num > 0) {
    ...
```

```javascript
function doLoop(num){
    do{
      //user code here
```}

```javascript
function forLoop(num){
    var i;
    for (i=0;i<num;i++){
      alert(num);
    }
```

```javascript
break causes immediate termination of the loop.
loop statements after continue are skipped and the next execution of the loop is performed.
```

```javascript
function forInLoop(){
    var s,x;
    for (x in document) {
        s=x + ' = ' + document[x];
        alert(s);
    }
```

```javascript
This code is best tested in Opera which offers the option of stopping the script at each alert. In place of document, any JavaScript object or array can be used to loop through its properties/elements.
```