

# JavaScript Gotchas

## Recommendations



JavaScript

Pierre Nallet

[www.javascriptgotchas.com](http://www.javascriptgotchas.com)



# A good return

Always terminate  
statements with ;

```
return ;
```

```
return x;
```

```
return {  
  x:2  
};
```

Start the returned  
expression on the same  
line

# switch and break

End your case statements  
with break ...

```
switch(dayOfWeek)
{
  case 0: ... break;
  case 1: case 2: return something;
  ...
  default: throw new Error("invalid arg");
}
```

... or return

Consider adding a default case

# Don't let blocks block you

```
if (condition) doSomething();  
doSomethingElse();
```


← Consider one line if statements

```
if (condition){  
    doSomething();  
}  
doSomethingElse();
```

← Consider adding braces


# The old parseInt

Always specify **radix** in **parseInt**



```
parseInt("010", 10) // 10
```

Consider using  
parseFloat or Number



```
parseFloat("010") // 10  
Number("010") // 10
```

# Number imprecision

Remember number precision problems



```
var noTruth = 0.3 === 0.1 + 0.2  
var truth = close(0.3, 0.1 + 0.2);
```

Consider using tolerance




Terminate loops with inequalities, not equalities



```
for(var i = 0; i < 10; i = i + 0.1)  
{  
  ...  
}
```

# Not a number is a number

Use `isFinite` to protect against non finite values




```
if(isFinite(n)){  
    // all is good  
}
```



Do **not** write `n === NaN`

Consider using `isNaN`

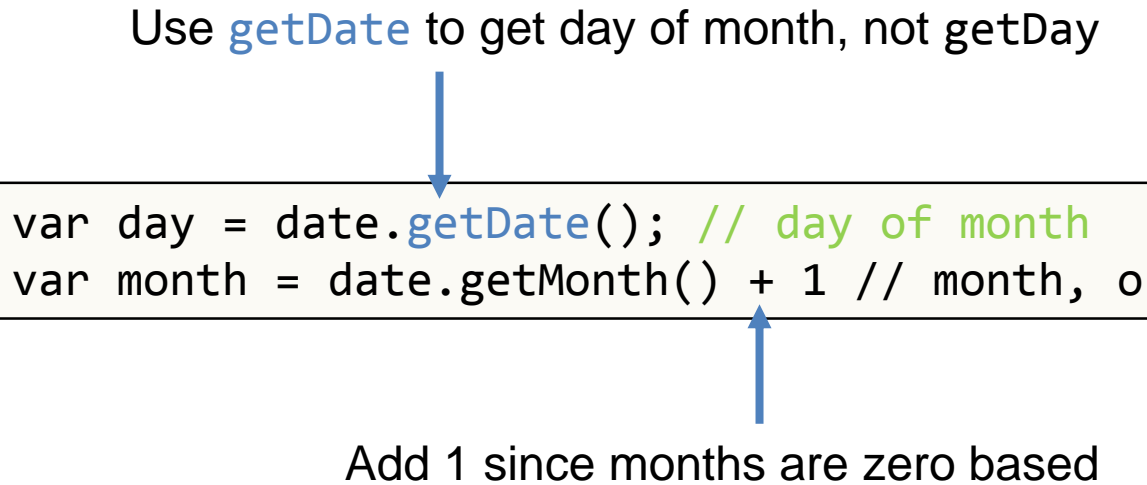


```
if(isNaN(n)){  
    // Could be infinite  
}
```

Remember: `isNaN` not the same as `Number.isNaN`

# A bad date

Use `getDate` to get day of month, not `getDay`



```
var day = date.getDate(); // day of month  
var month = date.getMonth() + 1 // month, o
```

The diagram consists of a central yellow box containing two lines of JavaScript code. A blue arrow points from the text 'Use getDate to get day of month, not getDay' above to the `getDate()` method in the first line. Another blue arrow points from the text 'Add 1 since months are zero based' below to the `+ 1` in the second line.

Add 1 since months are zero based



# + doesn't add up

Add items of  
the same type



```
2 + 3  
"hello " + name
```

Be careful  
when adding  
different types



```
"the number is " + num // OK  
1 + someString         // problem
```

Don't use + on objects

# forEach for everyone

use `forEach` for array iteration



```
array.forEach( function(item) {  
    // ...  
} );
```

Do **not** use for loops on arrays

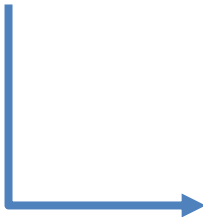
# push your array

Use **push** to add new items to an array.



```
myArray.push(post);
```


Verify array bounds before  
writing to arrays via an index



```
if (index < 0 || index >= posts.length) {  
    throw new Error('Invalid index');  
}  
posts[index] = post;
```

# splice, don't delete

Use `array.splice`, to remove items from array



```
var a = [1, 2, 3, 4, 5];  
a.splice(2, 1);  
// a contains 1,2,4,5
```

Do **not** delete on arrays

# Sorting out array.sort

```
var numbers = [1, 5, 10]; // [1, 5, 10]  
numbers.sort(ascending); // [1, 5, 10]
```

Use sort functions  
for sorting arrays

```
var ascending = function(x, y){  
  if (x > y)  
    return 1;  
  else if (x < y)  
    return -1;  
  return 0;  
}
```

You don't have to use a sort function if the array contains only strings

# Arrays vs. dictionaries

Use only numbers to access array items



```
var array = [1,2,3] ;  
array[0];
```


Use empty JavaScript objects for  
dictionaries, not arrays



```
var dict = {};  
dict.name = value1;  
dict['age'] = value2;
```

# All equals are not equal

Always be on guard against a missed = sign.  
Equality expression is == or ===, not =



```
if (userChoice == "Purchase") {  
    // should purchase  
}
```

# More equality is better

Prefer `===` and `!==`  
over `==` and `!=`



```
if (x === 0){  
    //do something  
}
```

Use `==` and `!=` only when you are sure you compare with the same type



# null is not undefined

Consider setting valid variables to null



```
var p = {name: 'Alice', age: null};
```

Checks for undefined



```
typeof x === "undefined"  
a.b === void 0
```

Checks for null or undefined



```
x == null
```

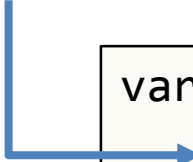
# Missing the argument

```
function addToCart(productName, quantity){  
    if(quantity === void 0) quantity = 1;  
    ...  
}
```

Consider default parameters


# var makes a difference

Always use var when  
declaring variables.



```
var setup = function (mode) {  
  var unload = mode === "Production";  
  // ...  
}
```

Consider using window.  
when accessing global  
variables



```
var setup = function (mode) {  
  window.unload =  
  // ...  
}
```

# namespaces


When (potentially) reinitializing a variable, use the `||` syntax.



```
var app = app || {}  
app.core = app.core || {}  
app.core.log = function() {}  
// ...
```

# this and that

Capture this before  
reaching callbacks



```
function computeSum(array){  
  this.total = 0;  
  var that = this;  
  array.forEach(function(item){  
    that.total += item;  
  });  
}
```

Stay up to date

[www.javascriptgotchas.com](http://www.javascriptgotchas.com)