Doing Web Apps

JavaScript Coding Guidelines

Autor: Rüdiger Marwein
Letzte Änderung: 2012-05-04
Version: 1.0

Übernommen von den jQuery Coding Guidelines (gekürzt)

Quelle: http://docs.jquery.com/JQuery_Core_Style_Guidelines

Inhaltsverzeichnis

1.Abstände....3
2.Kommentare....3
3.Vergleiche....3
4.Blöcke....4
5.Funktionsaufrufe....5
6.Arrays und Objekte....5
7.Deklarationen und Wertzuweisung....6
8.Zeichenketten....6

1.Abstände

Verwenden Sie zur Einrückung immer Tabs.

Verwenden Sie Leerzeichen zur besseren Lesbarkeit (aber nicht zur Einrückung).

Öffnenden Klammern muss ein Leerzeichen folgen.

```
// Nein:
if(blah===="foo"){
  foo("bar","baz",{zoo:1});
}

// Ja:
if ( blah === "foo" ) {
  foo( "bar", "baz", {zoo: 1} );
}
```

Verwenden Sie keine Tabs innerhalb einer Zeile

```
var a = true,
  // Ja
  c = false,
  // Nein
  b      = false;
```

Leere Zeilen dürfen keine Leerzeichen enthalten.

Es dürfen keine Leerzeichen am Ende einer Zeile stehen.

2.Kommentare

Lange Kommentare verwenden

```
/*
...
*/
```

Einzeilige Kommentare erhalten je eine eigene Zeile und stehen über der Zeile, die sie referenzieren. Des weiteren muss sich über der Kommentarzeile eine Leerzeile befinden.

Beispiel:

```
var some = "stuff";

// We're going to loop here
for ( var i = 0; i < 10; i++ ) {}
```

3.Vergleiche

Strikte Prüfungen (===) sollen gegenüber == bevorzugt eingesetzt werden, soweit möglich.

4.Blöcke

if/else/for/while/try müssen immer geschweifte Klammern haben und müssen immer mehrzeilig sein.

Blöcke müssen immer geschweifte Klammer haben.

```
// Nein:
if ( true )
   blah();

// Ja:
if ( true ) {
   blah();
}
```

Ausdrücke dürfen nicht in der selben Zeile stehen wie Bedingungen.

```
// Nein:
if ( true ) return;
if ( true ) blah();

// Ja:
if ( true ) {
   return;
}

if ( true ) {
   blah();
}
```

else/else if/catch sitzen mit der öffnenden geschweiften Klammer in einer Zeile.

```
if ( blah ) {
   baz();
} else {
   baz2();
}
```

Ausnahme: `else if` ist zulässig, ein else { } ist nicht notwendig.

```
} else if ( test ) {
  blah();
}

// unnötig:
} else {
  if ( test ) {
    blah();
  }
}
```

Ternäre Operatoren sind nicht anstelle von if/else zu verwenden

```
// Nein:
var a = b == 0 ? "null" : "nicht null";

// Ja:
var a = "nicht null";
if ( b == 0 ) {
   a = "null";
}
```

5.Funktionsaufrufe

Verwenden sie immer Leerzeichen rund um Funktionsparameter.

```
foo( true );
foo( "blah" );
```

Ausnahme: `foo(true)` (keine Leerzeichen) ist erlaubt, wenn es sich innerhalb eines anderen Funktionsaufrufes befindet.

```
foo( bar(true) );
```

Ausnahme: Funktions-, Objekt- und Array-Literale werden direkt an die Klammern angelegt, SOFERN sie das einzige Argument sind.

```
foo(function() { });
foo([   ]);
foo({   });
```

Ausnahme: Mehrzeilige Funktions-, Objekt- und Array-Literale werden direkt an die schließende Klammer angelegt (und an die Öffnende NUR, wenn es das einzige Argument ist).

```
//Ja:
foo( true, { blah: "baz" });

//Nein:
foo( true, {
  blah: "baz"
});
```

Leere Funktionsaufrufe brauchen keine extra Leerzeichen.

```
foo();
```

Hinter Kommas und Semikolons muss immer ein Leerzeichen stehen, außer am Ende einer Zeile.

6.Arrays und Objekte

Leere Objekte und Arrays benötigen keine extra Leerzeichen.

```
[]
{}
```

7.Deklarationen und Wertzuweisung

Wertzuweisungen müssen immer mit einem Semikolon abgeschlossen werden.

Nach Semikolons muss immer ein Zeilenumbruch folgen.

Wertzuweisungen in einer Deklaration müssen immer in einer eigenen Zeile stehen. Deklarationen ohne Wertzuweisung müssen am Anfang der Deklaration stehen.

Beispiel:

```
var a, b, c,
  test = true,
  test2 = false;
```

8.Zeichenketten

Zeichenketten müssen immer mit Gänsefüßchen angegeben werden, nicht mit Hochkommas.