Scripting: Tips zur JavaScript Beschleunigung

JavaScript-Performance
Um JavaScript zu beschleunigen gibt es verschiedene Theorien und sind meistens auch Browser abhängig.

Aufgrund des offenen Source-Code's ist die Firefox-Engine "Gecko" bestens dafür geeignet Performance-Tests durchzuführen.
Das Ergebnis sind folgende 6 Tricks:

1. `new Function()` gegen `function()` tauschen

Vorher:

```
element.onclick = new Function("...")
```

Nachher

```
element.onclick = function("...")
```

2. `eval()`, `with()` und `try-catch` meiden

Die Funtion `eval` brauchen eh nur unsichere Programmierer und läßt sich mit entsprechenden Fehlerprüfungen umgehen.

Vorher:

```
return eval("document.forms[0]." + field);
```

Nachher:

```
return document.forms[0][field];
```

Das `with`-Paradigma dient for allem der Erstellung von unleserlichen Code und der Faulheit der Programmierer.
Dennoch wird bei jedem Aufruf der ganze DOM-Baum durchsucht.
Ein kurzes Zwischenspeichern des gesuchten Objektes ist wesentlich effektiver.

Vorher:

```
with(document.forms[0]) {
```


(c) 2025 huschi <huschi@huschi.net>

```
  alert(elements.length);
}
```

Nachher:

```
var form = document.forms[0];
alert(form.elements.length);
```

Auch das `try-catch`-Paradigma läßt sich mit vernüftiger Fehlerkontrolle umgehen.

Vorher:

```
try {
  ...
} catch() {
  ...
}
```

Nachher:

```
if () {
  ...
} else {
  ...
}
```

3. Nach Möglichkeit immer lokale Variablen nutzen

Vorher:

```
var a, b = 1;

function test() {
  var c = a + b;
}
```


(c) 2025 huschi <huschi@huschi.net>
URL: https://huschi.net/index.php?action=artikel&cat=22&id=283&artlang=de

Nachher:

```
function test() {
  var a, b = 1;
  var c = a + b;
}
```

4. `for-in` nur wenn Array-Key keine fortlaufende Nummer ist

Vorher:

```
for (var i in array) {
  alert(array[i]);
}
```

Nachher:

```
var length = array.length;
for (var i = 0; i < length; i ++) {
  alert(array[i]);
}
```

5. Strings als Parameter vermeiden bei `setTimeout()` und `setInterval()`

Vorher:

```
setInterval("func()", 10000);
setTimeout("func(" + value + ")", 10000);
```

Nachher:

```
setInterval(func, 10000);
 setTimeout (function() {func(value)}, 10000);
```

6. JavaScript komprimieren

Hinter einem JavaScript-Kompressor verbirgt sich ein simples Prinzip:

(c) 2025 huschi <huschi@huschi.net>
URL: https://huschi.net/index.php?action=artikel&cat=22&id=283&artlang=de

Scripting: Tips zur JavaScript Beschleunigung

JavaScript-Code wird im Code verkleinert. Dazu zählen in erster Linie Funktions- und Variablennamen die Verkürzt werden. Zusätzlich werden alle unnötigen Zeichen (Space, Tab, Return) entfernt.

Das Ergebnis ist ein fast unleserlicher Code, der aber aufgrund seiner kompakten Form deutlich schneller vom Browser geparst werden kann.

Eindeutige ID: #1276
huschi
2007-12-11 09:45

(c) 2025 huschi <huschi@huschi.net>
URL: https://huschi.net/index.php?action=artikel&cat=22&id=283&artlang=de