onload Event - Cross Browser kompatibler DOMContentLoaded

== Hinweis ==
Dieser Artikel ist noch nicht fertiggestellt

== Einführung ==
Will man JavaScript Code ausführen, nachdem die Seite fertig geladen wurde benutzt man im Normalfall das onload Event im Body.

Quellcode

```
1. <body onload="alert('hello world')">
```

Leider führen unterschiedliche Browser diese Funktion zu unterschiedlichen Zeitpunkten aus.
Hat man nun umfangreiche, externe Quellen eingebunden (z.B. Kartenmaterial von Google Maps) dann wird die Funktion erst sehr spät ausgeführt.

== Beispiel ==
Das folgende Beispiel ist so bearbeitet, dass ein Bild für 5 Sekunden lädt. Das eigentliche Script wird also (mit den meisten Browsern) erst nach 5 Sekunden ausgeführt.
demo.easy-coding.de/javascript…cross-browser/regular.php

== Firefox Lösung ==
Der Firefox kennt das Event "DOMContentLoaded" (Mozilla Dokumentation), das ausgeführt wird wenn der HTML Code fertig interpretiert wurde. Damit hängt man sich vor nachgelagerte Scripte oder eingebundene Bilder.

Quellcode

```
1. document.addEventListener("DOMContentLoaded", function(){
2. alert('firefox sagt: fertig');
3. }, false);
```

== Internet Explorer Workaround ==
Wofür der Firefox Lösungen bietet, hat der Internet Explorer leider nur Workarounds.
Diese reichen über das mittels "defer" verzögerte Einhängen von externen <script> Elementen bis hin zu einem "Linksscrollen" das erst funktioniert, wenn die Seite fertig geladen wurde. Da das einhängen externer Scripte mehr Ressourcen verbraucht hat sich das JavaScript Framework jquery (in Version 1.3.2) für letzten Workaround entschieden.

Daher wird er auch hier behandelt.

Quellcode

```
1. (function () {
2. try {
3. // throws errors until after ondocumentready
4. d.documentElement.doScroll('left');
5. } catch (e) {
6. setTimeout(arguments.callee, 50);
7. return;
8. }
9. // no errors, fire
10. alert('internet explorer sagt: fertig');
11. })();
```

Alles anzeigen

Microsoft dokumentiert diesen Weg für alle bisher verfügbaren IE Versionen: msdn2.microsoft.com/en-us/library/ms531426.aspx.
Durch die Closure um den eigentlichen Event und die Verwendung von arguments.callee, das die Closure referenziert sind wir übrigens in einer Endlosschleife, dir wir im 50 Millisekunden Takt aufrufen.

== Event Syntax ==
Wenn das Leben nicht schon schwierig genug wäre, so bieten die verschiedenen Browser noch unterschiedliche Methoden um Events hinzuzufügen.
Diese Problematik wird im Script behandelt, auch wenn in diesem Wiki Artikel nicht darauf eingegangen wird.

== Event Queue ==
Die Anforderung an unser Script ist nun einerseits, dass man beliebig viele Events hinzufügen kann - andererseits sollen die Events auch noch gefeuert werden, wenn der Event versucht wird hinzuzufügen nachdem die Seite schon fertig ist.

Dafür gibt es eine Abfrage beim hinzufügen der Events:

Quellcode

```
1. // If the DOM is already ready
2. if ( this.isReady )
3. // Execute the function immediately
4. fn.call( window, this );
5. // Otherwise, remember the function for later
6. else
7. // Add the function to the wait list
8. this.readyList.push( fn );
```

== Lösung ==
Das Script ist als ready.js unter MIT/GPL Lizenzierung frei zur Verfügung. Für die produktive Verwendung empfiehlt sich die Verwendung der minimierten Version: ready.min.js (0,4 KB gzipped)
Für Tests könnt ihr außerdem den vollen Quelltext herunterladen: ready.js (3,2 KB)

Nach Einbindung fügt man Funktionen über folgende Syntax hinzu:

Quellcode

```
1. <script type="text/javascript" src="ready.min.js"></script>
2. <script type="text/javascript">
3. ready.push(function() {
4. alert('ready event geladen');
```

```
5. });
6. </script>
```

Eine Live Demo findet ihr unter demo.easy-coding.de/javascript/onload-event-cross-browser/.

== Quelltext ==

Quellcode

```
1. /**
2. * ready.js
3. *
4. * Author: Torben Brodt <[email]t.brodt@gmail.com[/email]>
5. * Summary: Cross-browser wrapper for DOMContentLoaded
6. * Updated: 07/09/2009
7. * License: MIT / GPL
8. * Version: 1.1
9. *
10. * URL:
11. * [url]http://www.easy-coding.de[/url]
12. * [url]http://jquery.com/dev/svn/trunk/jquery/MIT-LICENSE.txt[/url]
13. * [url]http://jquery.com/dev/svn/trunk/jquery/GPL-LICENSE.txt[/url]
14. *
15. * Full Description:
16. * A page has loaded after all external resources like images have been loaded.
17. * Should all scripts wait for that? a better bevaviour is to wait for the dom content being ready.
18. *
19. * This script has workarounds for all the big browsers meaning the major versions of firefox, internet explorer,
    opera, safari and chrome.
20. * You can use it without risk, since the normal "onload" behavior is the fallback solution.
21. *
22. * Most of the source is lended from jquery
23. */
24. var ready = new (function () {
25. var readyBound = 0, d = document, w = window, t = this, x;
26. t.isReady = 0;
27. t.readyList = [];
28. function bindReady() {
29. if ( readyBound ) return;
30. readyBound = 1;
31. // Mozilla, Opera and webkit nightlies currently support this event
32. if ( d.addEventListener ) {
33. // Use the handy event callback
34. x = "DOMContentLoaded";
35. d.addEventListener( x, function(){
36. d.removeEventListener( x, arguments.callee, false );
37. ready.ready();
38. }, false );
39. // If IE event model is used
40. } else if ( d.attachEvent ) {
41. // ensure firing before onload,
42. // maybe late but safe also for iframes
43. x = "onreadystatechange";
44. d.attachEvent(x, function(){
45. if ( d.readyState === "complete" ) {
46. d.detachEvent( x, arguments.callee );
47. ready.ready();
48.
49.
50.
```

```
51. }
52. });
53. // If IE and not an iframe
54. // continually check to see if the document is ready
55. if ( d.documentElement.doScroll && w == w.top ) (function(){
56. if ( t.isReady ) return;
57. try {
58. // If IE is used, use the trick by Diego Perini
59. // [url]http://javascript.nwbox.com/IEContentLoaded/[/url]
60. d.documentElement.doScroll("left");
61. } catch( error ) {
62. setTimeout( arguments.callee, 0 );
63. return;
64. }
65. // and execute any waiting functions
66. ready.ready();
67. })();
68. }
69. // A fallback to window.onload, that will always work
70. w.onload = ready.ready; // TODO: compliant? t.event.add( window, "load", t.ready );
71. };
72. // Handle when the DOM is ready
73. t.ready = function() {
74. // Make sure that the DOM is not already loaded
75. if ( !t.isReady ) {
76. // Remember that the DOM is ready
77. t.isReady = 1;
78. // If there are functions bound, to execute
79. if ( t.readyList ) {
80. // Execute all of them
81. for(var i=0; i<t.readyList.length; i++) {
82. t.readyList[i].call( w, t );
83. };
84. // Reset the list of functions
85. t.readyList = null;
86. }
87. // Trigger any bound ready events
88. d.loaded = true; // TODO: compliant? this(document).triggerHandler("ready");
89. }
90. };
91. // adds funtion to readyList if not ready yet, otherwise call immediately
92. t.push = function(fn) {
93. // Attach the listeners
94. bindReady();
95. // If the DOM is already ready
96. if ( t.isReady )
97. // Execute the function immediately
98. fn.call( w, t );
99. // Otherwise, remember the function for later
100. else
101. // Add the function to the wait list
102. t.readyList.push( fn );
103. return t;
104. };
105. })();
```

Alles anzeigen