Events und Event-Handling

1)In JavaScript:

Ereignisse in einer HTML-Seite treten auf, wenn etwas mit beliebigen Elementen der Seite geschieht. Dies kann sein

- Eine Nutzeraktion wie z.B. Anklicken, Selektieren oder Ziehen von Objekten
- Laden oder Entladen des Dokuments

Damit ein Event sinnvoll behandelt werden kann, sind drei Dinge von Bedeutung:

1. Das Ereignis muss überhaupt bemerkt werden
2. Es muss bekannt sein, um was für ein Ereignis es sich handelt, an welchem Element und bei welchen Koordinaten es auftritt
3. Eine Funktion muss mit dem Auftritt des Events verknüpft sein.

Zu 1)Für die Erfüllung von Punkt eins sorgt der Event-Listener. Aufgabe des Listeners ist es , den sogenannten Event-Handler zu starten, ein Funktionsobjekt, das die JavaScript-Anweisungen enthält, die beim Eintritt des Ereignisses, auf das der Handler wartet, ausgeführt werden.

```
<p id="p1" onclick="alert("Ein Ereignis tritt auf")">Bitte hier klicken</p>
```

Zu 2)Die Eigenschaften des Event-Objekts: Hier im Beispiel handelt es sich um ein Ereignis vom Typ „Klick", das an einem <p>-Container auftritt, den man deshalb als „target" bezeichnet. Praktischerweise fasst JavaScript diese Information zusammen und stellt sie als „Event-Objekt" zur Verfügung. Ein solches Event-Objekt wird bei jedem auftretenden Ereignis gebildet.

Zu 3)Der Event-Handler ist die an das Auftreten des Ereignisses gebundene Funktion, die das Ereignis „behandelt". Einfach zu zeigen ist dies bei der Scriptbindung – die Funktion ist fett markiert:

```
meinP.addEvent-Listener("click", function() {
        alert("Ein Ereignis trat auf");
        }, false);
```

Überblick:

Der „Listener" wird an ein Objekt gebunden, um an diesem auf das Auftreten von Ereignissen (Events) zu „horchen". Es wird quasi zu einer Eigenschaft des beobachteten Objekts. Seine Aufgabe ist es. Im Event-Fall den „Handler" zu starten – eine Funktion, die das Ereignis „behandelt". Diese Funktion wird auch als „Callback" bezeichnet. Im Grunde sind Callback und Handler also dasselbe.

2)In jQuery

jQuery bietet zum Unterschied zu JavaScript seine eigene Interpretation des Event-Objekts.

z.B.

- .click(function()) bindet einen Handler (function) an das Click-Event
- .click(function(e)) wie oben, nur dass die Funktion das Event-Objekt „e“ übergeben bekommt
- .dbclick(function()) bindet einen Handler an das Doppel-Click-Event
- .mousever(function()) bindet einen Handler an das Mousover-Event

Es gibt alle Methoden die Übergabe, wenn in der Klammer von „function“ etwas steht, z.B. „e“.

typische Syntax:

```
$(“inhalt p“).click(function() {
        alert(“Der Absatz wurde angeklickt.“);
});
```

Direktes Binden von Event-Listenern mit „.on()“

Bindet an die Elemente der aktuellen Collection (hier „p“) einen Handler „function“ an das Event „click“.

```
$(“p“).on(“click“, function() {
        alert(“Du hast mich angeklickt.“);
        });     //Hier beispielsweise erhalten alle <p>-Container einen Click-Listener.
```

Man muss nicht alle <p> auswählen, sondern man kann aber auch nur bestimmte Elemente mit dem Event auswählen, z.B. die, die folgende Klasse besitzen: „.klickbar“

```
$(“.klickbar“).on(“click“, function() {
        alert(“Element wurde angeklickt.“);
        });
```

Lösen direkt gebundener Listener:

Ein Ablösen ist nur nötig, wenn der Event-Listener nicht bis zum Entladen der Seite an einem Element gültig bleiben soll.

.off entfernt alle Event-Bindungen

Beispiel:

```
$(“p“).on(“click“, function() {
        alert(“Anklicken geht nur einmal.“);
        $(this).off(“click“);    //keine weiteren Klicks möglich
        });
```

Übung: Zeitstempel

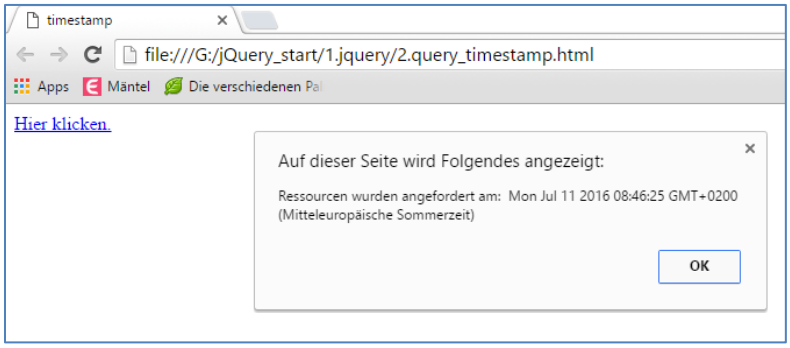
Eigenschaft des Ereignisses „event.timeStamp“: Zeitstempel des Ereignisses in Millisekunden seit 1. Jänner 1970.

Über die „timeStamp“-Eigenschaft kann man den Zeitpunkt eines Ereignisses erkennen. Die Eigenschaft gibt einen Wert zurück, den man als Argument eines Date-Objektes einsetzen kann:

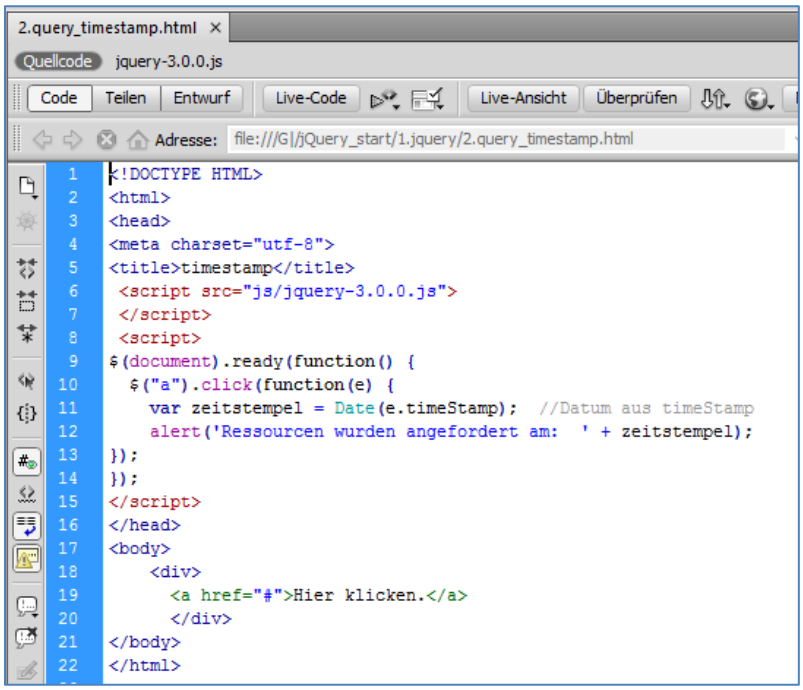
```
$(document).ready(function() {
  $("a").click(function(e) {
     var zeitstempel = Date(e.timeStamp);       //Datum aus timeStamp
     alert('Ressourcen wurden angefordert am:  ' + zeitstempel);
   });
});
```

Übung:
Erstelle eine neue „2.query_timestamp.html“. Verweise im <head> auf die jQuery-Datei im „js“-Ordner und füge darunter, ebenfalls im <head> die passende „timeStamp“ Methode an. Im <body> soll eine einzige Hier klicken das Klick-Event möglich machen.

Ergebnis:



Code:

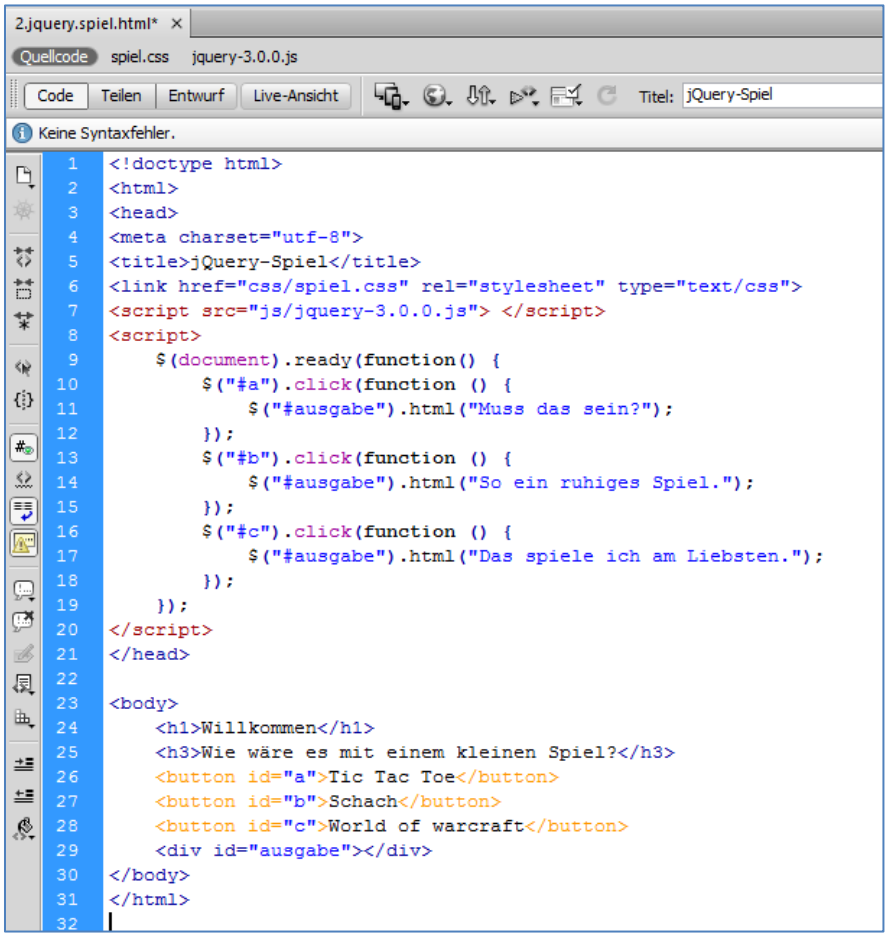


Beispiel click-Funktion inkl. „html-Methode“

Ralph Steyer in: jQuery, das universelle JavaScript-Framework für das interaktive Web und mobile Anwendungen; Hanser-Verlag, München, 2014, S.23-28

Erstelle die Site: „2.jquery.spiel.html“

Ergebnis:




```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>jQuery-Spiel</title>
<link href="css/spiel.css" rel="stylesheet" type="text/css">
<script src="js/jquery-3.0.0.js"> </script>
<script>
    $(document).ready(function() {
        $("#a").click(function () {
            $("#ausgabe").html("Muss das sein?");
        });
        $("#b").click(function () {
            $("#ausgabe").html("So ein ruhiges Spiel.");
        });
        $("#c").click(function () {
            $("#ausgabe").html("Das spiele ich am Liebsten.");
        });
    });
</script>
</head>

<body>
    <h1>Willkommen</h1>
    <h3>Wie wäre es mit einem kleinen Spiel?</h3>
    <button id="a">Tic Tac Toe</button>
    <button id="b">Schach</button>
    <button id="c">World of warcraft</button>
    <div id="ausgabe"></div>
</body>
</html>
```

Code:

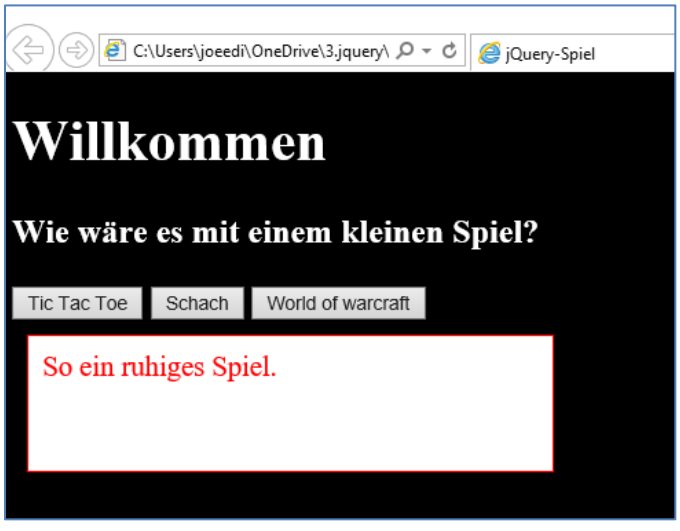
```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>jQuery-Spiel</title>
<link href="css/spiel.css" rel="stylesheet" type="text/css">
```

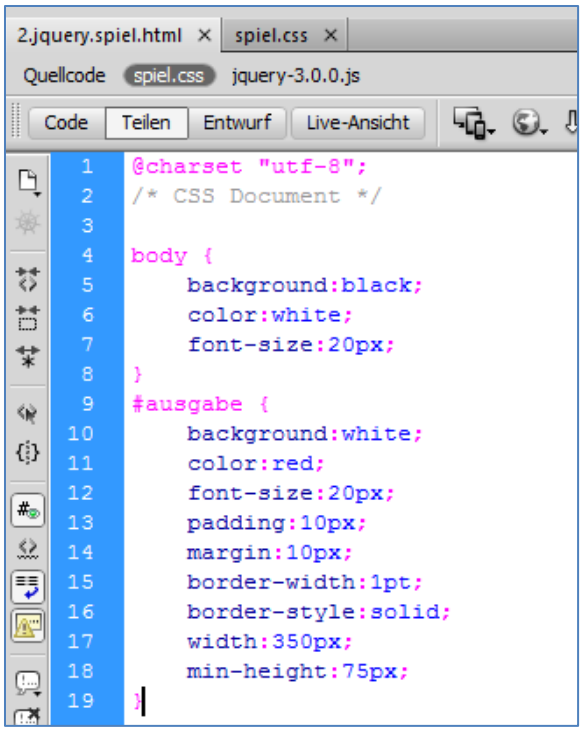
```
<script src="js/jquery-3.0.0.js"> </script>
<script>
        $(document).ready(function() {
    $("#a").click(function () {
                $("#ausgabe").html("Muss das sein?");
            });
            $("#b").click(function () {
                $("#ausgabe").html("So ein ruhiges Spiel.");
            });
            $("#c").click(function () {
                $("#ausgabe").html("Das spiele ich am Liebsten.");
            });
  });
</script>
</head>

<body>
        <h1>Willkommen</h1>
  <h3>Wie wäre es mit einem kleinen Spiel?</h3>
  <button id="a">Tic Tac Toe</button>
  <button id="b">Schach</button>
  <button id="c">World of warcraft</button>
  <div id="ausgabe"></div>
</body>
</html>
```

CSS – Verbesserung:

Damit die Optik verbessert wird, erstelle etwas CSS:

das wurde bereits im HTML-Code vorhergesehen und die CSS „spiel.css“ einbezogen,





Erklärung:

Im Inneren der „document.ready“-Methode werden drei Ereignisbehandlungsroutinen notiert, die jeweils die Reaktion bei einem Klick auf die angegebenen Elemente spezifizieren. Hier sind es drei Schaltflächen, die jeweils mit einer eindeutigen ID gekennzeichnet sind.

Die Zuordnung zur richtigen Funktion erfolgt über die ID und das Auflösen der Funktion innerhalb der Methode „click()“. Hier handelt es sich um eine „anonyme“ Funktion, da kein Bezeichner verwendet wird.

Klickt der Anwender auf eine Schaltfläche, wird in einem Bereich der Webseite eine spezifische Textausgabe angezeigt. Dazu verwendet man

…das Rautezeichen und die Methode
html () …für den Zugriff auf den Inhalt.

Html-Methode

Mit der Methode html () kann man den Inhalt eines Textknotens abfragen und setzen.

z.B. im Beispiel oben

```
$("#ausgabe").html("So ein ruhiges Spiel.");
```

Quellen:

Maximilian Vollendorf, Frank Bongers in: jQuery – Das umfassende Handbuch, Galileo Press, Bonn 2014, S. 137—159 (Events)

Sascha Schoppengerd, Amalia Schoppengerd in: jQuery Praxiseinstieg, mitp-Verlag, 2010

Ralph Steyer in: jQuery, das universelle JavaScript-Framework für das interaktive Web und mobile Anwendungen; Hanser-Verlag, München, 2014, S.23-28