jQuery - Einstieg

1. Einstieg – Theorie
2. jQuery einbinden
3. jQuery Funktion = $
4. Übung Hallo Welt
5. Übung 2: Event-timestamp
6. Theorie: Selektion von Elementen, um zugreifen zu können
7. Übung: Alle angewählten „checked“-Buttons zählen und live ausgeben
8. Eigenschaften auslesen
9. Übung: Radio-Buttons zählen mit „length“

1)Einstieg – Theorie

jQuery ist eine JavaScript-Datei, die man in seine Webseiten aufnimmt. JQuery macht nichts, was man nicht auch mit reinem JavaScript erreichen könnte.

jQuery-Dokumentation unter: http://api.jquery.com

JavaScript-Frameworks stellen für jeden Entwickler eine enorme Arbeitserleichterung dar. Die *Frameworks* erledigen im Hintergrund die Arbeit, während man "vorne" oftmals schon mit wenigen Zeilen Code ansprechende Ergebnisse erzielen kann.
Was aber leistet jQuery eigentlich?

- DOM-Manipulation
- Ajax-Funktionalitäten
- Zahlreiche Effekte stehen zur Auswahl
- Erweitertes Event-System

Diese Aufzählung zeigt, wie leistungsfähig das *Framework* tatsächlich ist. Und die genannten Funktionalitäten kann man ganz einfach in der Webseite nutzen.

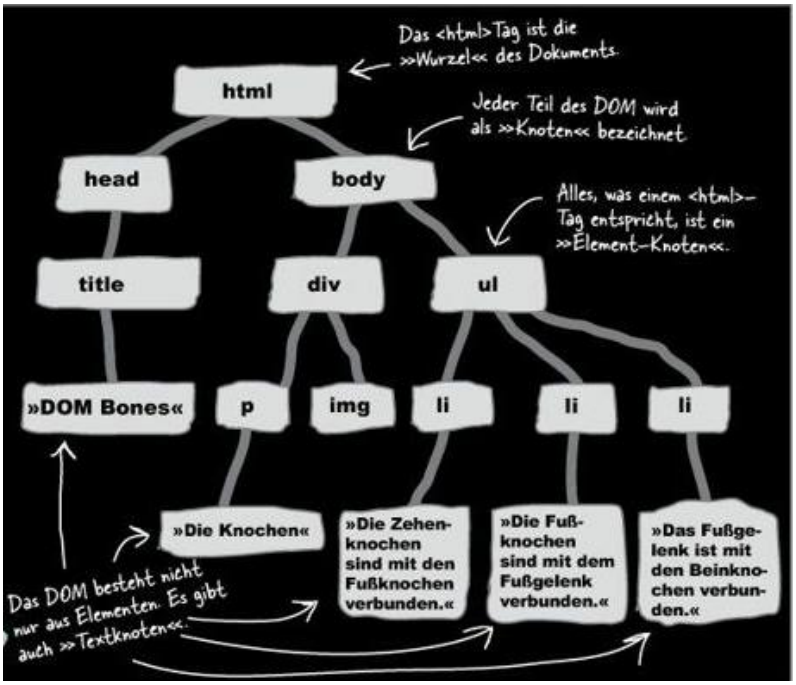
jQuery ist eine spezielle **JavaScript-Bibliothek** zum Ändern von Webseiten nach Bedarf (»on the fly«)

Intuitive Benutzeroberflächen mit Drag & Drop, sich ausklappende Bereiche, Tabs, die unterschiedliche Inhalte anzeigen – für all solche Dinge ist als clientseitige Skriptsprache im Browser meistens **JavaScript** zuständig. Größere Dinge von Hand per JavaScript zu erstellen, ist jedoch relativ mühsam. Dafür springen JavaScript-Frameworks wie jQuery in die Bresche. **jQuery ist im Kern eine JavaScript-Datei**, die man in der Seite einbinden kann. Danach steht einem neben den klassischen Java-Script-Befehlen diejenigen von jQuery zur Verfügung. Wenn man richtig gut JavaScript programmieren könnte, könnte man sämtliche jQuery-Funktionen im Grunde selbst nachbauen. Nur hätte das einen gewissen Aufwand. Das Team von jQuery hat entsprechend viele Jahre Arbeit bereits in die Entwicklung dieser Bibliothek investiert und diese Arbeit kann man nun nutzen.

Ein weiterer großer Vorteil von jQuery ist, dass man sich nicht mehr mit den Macken der verschiedenen Browser herumärgern muss.

Den Kern von jQuery bildet die Selektor-Engine »Sizzle«. jQuery empfängt Auswahlkriterien auf der Grundlage von CSS-Selektoren und liefert eine Reihe von Elementen aus dem Dokument zurück, die diese Kriterien erfüllen. Auf die so ausgewählten Elemente kann man dann eine Vielzahl von Funktionen anwenden. So lassen sich verschiedene Operationen durchführen oder Event-Listener hinzufügen.

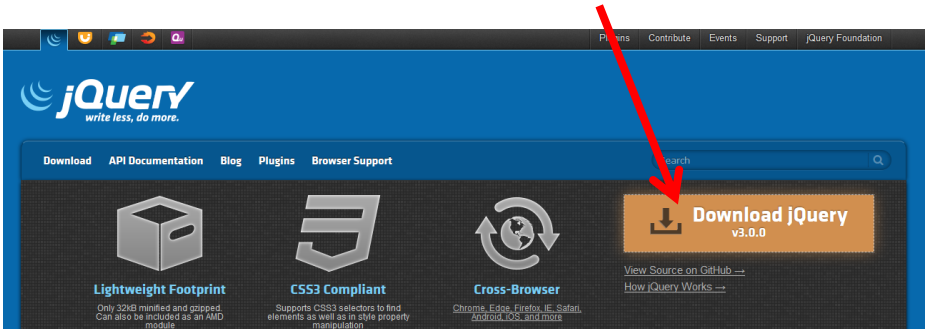
DOM:. Der Browser verwendet das Document Object Model (DOM), um aus einfachem HTML-Markup und CSS-Code eine anklickbare Seite aufzubauen – inklusive Text, Bildern, Videos und der anderen Dinge, die wir uns gerne ansehen. Was passiert tatsächlich, wenn der Browser eine Webeite darstellt?
Man kann sich den DOM auch als Röntgenbild vom Aufbau der Seite vorstellen. Im Gegensatz zu einem Röntgenbild können JavaScript und jQuery das DOM jedoch verwenden, um die *Struktur der Seite zu verändern*.



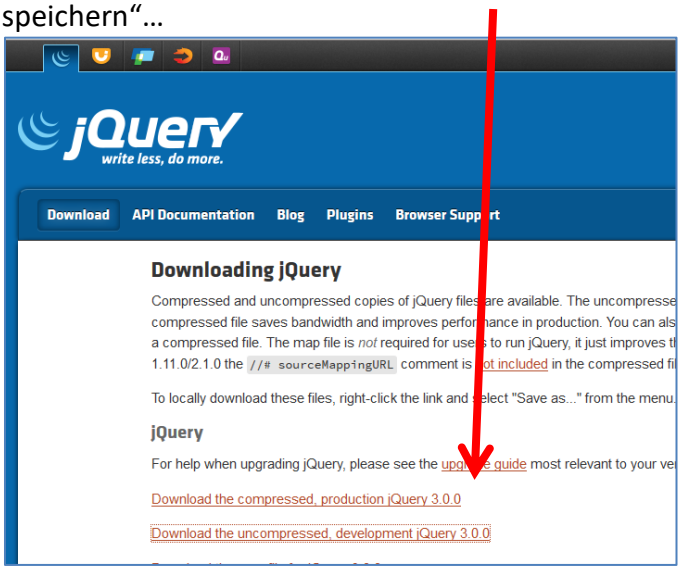
Der JavaScript-Interpreter verändert die ursprünglichen HTML- und CSS-Dateien nicht, sondern er modifiziert die DOM-Darstellung der Seite im Speicher des Browsers.

2)jQuery einbinden

Die offizielle Projektwebseite findet man unter http://jquery.com/. Von dort kann man das *Framework* herunterladen. Der Download besteht aus einer JavaScript-Datei.

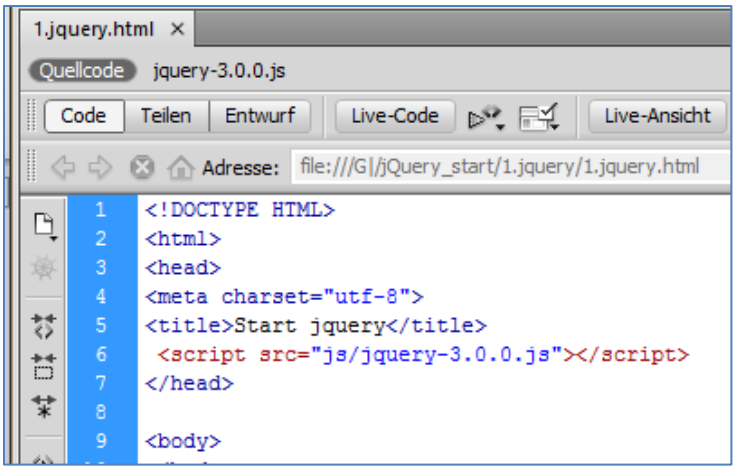


Wähle deine Version, z.B. „uncompressed…“ und dann : rechter Mausklick und „Ziel speichern“…



Um das *Framework* nutzen zu können, ist nun zunächst einmal nichts anderes nötig, als diese Datei in die betreffende Seite einzubinden. Das geschieht genauso, wie man das von herkömmlichen JavaScript-Dateien her gewohnt ist.

```
<head>
 <script src="js/jquery-3.0.0.js"></script>
</head>
```



In diesem Beispiel wird davon ausgegangen, dass die die Datei „jquery-3.0.0.js“ im Verzeichnis „js“ liegt.

Das ist allerdings nicht die einzige Möglichkeit. Denn man kann jQuery tatsächlich auch nutzen, ohne dass das *Framework* lokal gespeichert werden muss. Bereitgestellt wird das Ganze von der Google AJAX Libraries API. Dabei handelt es sich um ein von Google initiiertes Projekt, über das sich zahlreiche Open-Source-JavaScript-Bibliotheken einbinden lassen. Der Aufruf mittels script-Element sieht in diesem Fall folgendermaßen aus:

```
<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/3.0.0/jquery.min.js">
</script>
```

Man muss natürlich explizit die jQuery-Version angeben, die eingebunden werden soll. Welche Versionen momentan zur Verfügung stehen, kann man auf der Seite http://docs.jquery.com/Downloading jQuery nachlesen.

Die **Versionsnummer** ist im Dateinamen angegeben, z.B. `jquery-3.0.0.js`

Häufig wird auf Websites auch eine Version mit der **Dateierweiterung .min.js** verwendet. Dabei werden die überflüssigen Leerzeichen und Wagenrückläufe entfernt.
Dieser Vorgang wird **Minifizierung** genannt. Die Datei wird dadurch viel kleiner, sodass sie sich schneller herunterladen lässt. Allerdings sind diese Dateien viel schwieriger zu lesen.

3)jQuery Funktion = $

Das Dollarzeichen mit den runden Klammern ist der Kurzname für die jQuery-Funktion. Dadurch müssen wir nicht bei jedem Funktionsaufruf »jQuery()« schreiben.

Die jQuery-Funktion wird oft auch **„jQuery-Wrapper“ genannt: $**

Man kann in jQuery Elemente genauso ansprechen wie in CSS. Also einen Selektor selbst, wie z.B. „a“, eine Klasse z.B. „.my_class“ oder einen ID-Selektor, der mit einer Raute beginnt z.B. „#my_id“. Natürlich kann man auch direkt eine CSS-Eigenschaft ansprechen, wie z.B. „text-align: left“. Im nachfolgenden Beispiel wird der Selektor „a“ angesprochen.

4)Beispiel: Hallo, Welt!

Als Ergebnis wird ein Meldungsfenster ausgegeben, in dem die Zeichenfolge *Hallo, Welt!* angezeigt wird.

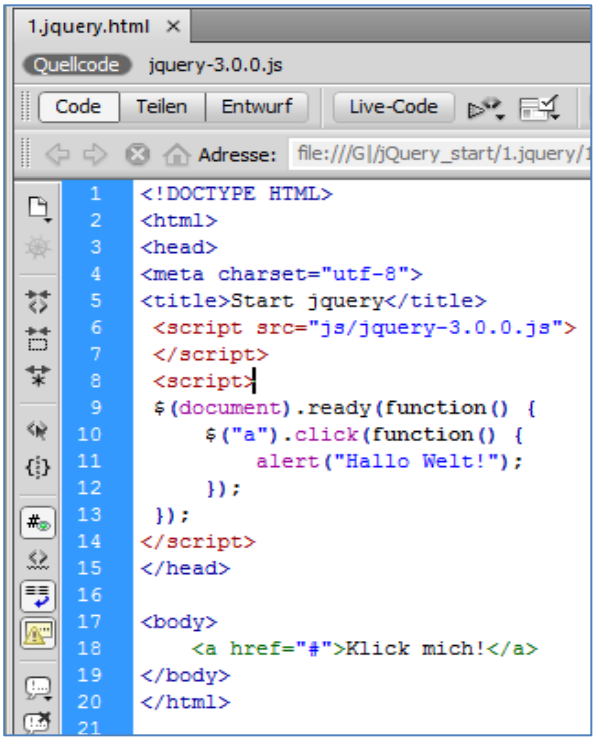
Übung: Erstelle die Datei „1.0.jquery.html“:

```
<head>
<script src="js/jquery-3.0.0.js"></script>
<script>
$(document).ready(function() {
 $("a").click(function() {
 alert("Hallo, Welt!");
 });
});
</script>
</head>
<body>  <a href="#">Klick mich!</a>
</body>
```

Beachte: Das Einbinden des Scripts und die Funktion stehen beide in einem separaten <script>-Block!

An dieser Stelle soll noch nicht explizit auf die Syntax eingegangen werden. Wichtig ist momentan lediglich, dass zunächst

- über $(document).ready(function()) ein sogenanntes ready-Element registriert wird. Dadurch wird sichergestellt, dass die entsprechenden DOM-Events zur Verfügung stehen.
- Anschließend wird der Selektor a angegeben, durch den alle a-Elemente des Dokuments



ausgewählt werden.

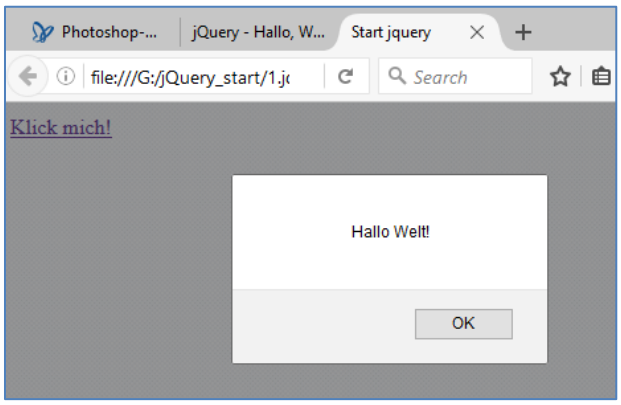
- Durch die click()-Funktion weist man im aktuellen Fall jedem Link das Klick-Event zu.
- Die Funktion besteht aus der bekannten JavaScript-Funktion alert().

Im Dokumentkörper <body> werden auf diese Weise sämtliche a-Elemente, also *Hyperlinks*, angesprochen.

```
<a href="#">Klick mich!</a>
```

Nach dem Anklicken des *Hyperlinks* wird ein einfaches Meldungsfenster angezeigt.

Ergebnis:



So einfach lässt sich also jQuery nutzen. Auch wenn man das *Framework* für solche einfachen Sachen natürlich normalerweise nicht verwendet. Die Funktionsweise ist aber auch bei komplexeren Anwendungen identisch.

Erklärung:

1) Die Platzierung des Codes steht in <head>. Es wäre aber auch durchaus sinnvolle, den Code ganz ans Ende, direkt vor das schließende </body> zu schreiben. Denn so kann man dem Browser die Möglichkeit geben, den gesamten HTML-Code im DOM aufzubauen. Wenn ein JavaScript nicht erreichbar sein sollte oder einen Fehler enthält, dann wäre der Großteil der Seite trotzdem noch nutzbar.

2)Damit der gesamte DOM aufgebaut ist, bevor man mit jQuery darauf zugreift, gibt es in jQuery das „ready()-Event“. Durch den Aufruf von

$(document).ready(…);

kann man sicherstellen, dass sämtliche Manipulationsbefehle des Frameworks erst nach Aufbau der gesamten DOM-Struktur ausgeführt werden.

3)Nachdem die Seite vollständig aufgebaut wurde, greift das Scrip mit Hilfe von „$(„a“) sämtliche Links im DOM ab und leitet bei allen gefundenen Elementen (bei uns hier jedoch nur eines) die Zuweisung des Eventhandlers mittels „.click(…)“ ein.

4)Und hier liegt die große Stärke von jQuery, denn man ist nicht gezwungen, das Ergebnis der Abfrage vor der Weiterverarbeitung in einer Variablen oder Ähnlichem zu speichern, sondern man kann die benötigten Funktionen oder Methoden direkt nacheinander ausführen lassen.

Der Aufruf von

$(“a“).click(…)

selektiert nicht nur die Elemente, sondern leitet auch direkt eine Folgeaktion ein. Man kann hierbei auch mehrere Funktionsaufrufe nacheinander starten. Getrennt werden die einzelnen Funktionen dabei jeweils durch einen Punkt.

Es ist egal, ob man die gewünschte Funktion vorab deklariert und dem „click(…)“-Event lediglich deren Bezeichnung übergibt oder ob man, wie hier oben im Beispiel, die Funktion direkt innerhalb der Zuweisung deklariert und sofort übergibt. Bei größeren Funktionen ist es übersichtlicher die Funktion vorab zu deklarieren, damit der Code übersichtlicher bleibt.

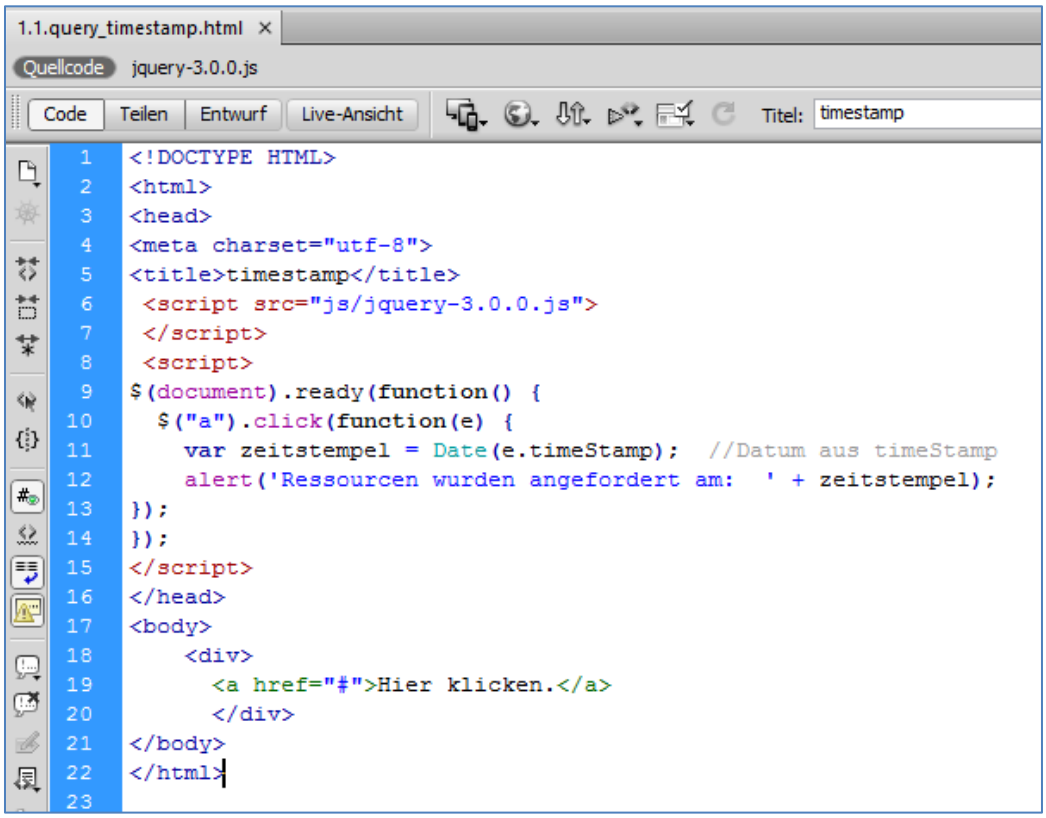
5)Übung 2: Event-timestamp

Ausgehend von der ersten Datei folgt hier eine kleine erste Erweiterung.
Speicher die Datei als „1.1.jquery_timestamp.html“

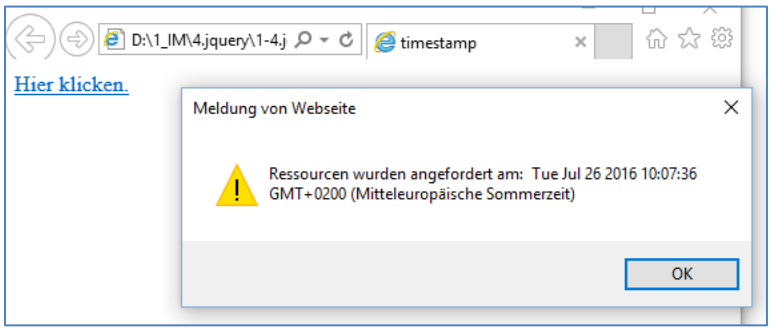
Mit dem Konstruktor kann ein Objekt erzeugt werden.

var e = jQuery.Event(„click“);

event.timeStamp
Der Zeitstempel in Millisekunden, wann das Ereignis erzeugt wurde.



Ergebnis:



6)Theorie: Selektion von Elementen, um zugreifen zu können

Sehr häufig möchte man bestimmte Elemente aus dem DOM auswählen (selektieren) um sie zu verändern, ein- oder auszublenden oder sie sonst irgendwie zu bearbeiten.

Im obigen Beispiel haben wir das mit dem Element „a“ getan, welches wir mit dem Eventhandler „.click()“ versehen haben, um ein alert-Fenster zu öffnen.

Man unterscheidet prinzipiell drei Basisselektoren, mit denen man auf bestimmte Elemente im DOM zugreifen kann und will:

- Die Selektion erfolgte dabei über das jeweilige Element. Dabei werden alle Elemente mit dieser Bezeichnung selektiert. Beispiel: „a“ oder „p“ oder „div“
- .class selektiert alle Elemente mit der genannten Klasse
- #id selektiert alle Elemente mit dem übergebenen ID

a)Allgemeine Schreibweise:

Eine Collection wird erzeugt: jQuery(„selektor“)

kürzer schreibt man **$(„selektor“)**

Beachte: die Selektoren der $()-Funktion werden grundsätzlich als String, also in Anführungszeichen übergeben. Der Rückgabewert eines jQuery-Selektors ist jeweils eine „Collection“.

Beispiele:

- $(„p“) alle P-Container des Dokuments
- $(„h1“) alle Überschriften erster Ordnung
- $("input:checked") alle aktivierten Checkboxen
- $(„myID“) das Element mit dem ID „myID“
- $(„div“) alle div-Elemente

Details: http://api.jquery.com/attribute-equals-selector

b)Mehrfachselektoren, Attributselektoren

Mehrfachselektoren: Die Basisselektoren lassen sich verschärfen, indem zwei oder mehr Elemente bzw. Klassen gleichzeitig abgefragt werden:

$(„.detail.color“)

Attributselektoren: Man kann Elemente anhand ihrer Attribute auswählen. Dabei werden die Selektausdrücke in eckigen Klammern geschrieben.

Beispiel:
$(„img[alt]“) selektiert alle Bilder mit Alternativtextangaben wie
<img src=“header-img.png“ alt=“Headerbild“>
aber nicht <img src=“footer-img.png“>

c)Selektieren durch Filter

jQuery stellt eine Reihe von Filtern bereit, mit denen man eine Selektion einschränken kann. Der Filter muss innerhalb des Selektorstrings stehen mit folgender Syntax: :filter

Beispiele:
- :first selektiert das erste Element des übergebenen Typs
- :last selektiert das letzte Element
- :header selektiert alle Elemente, die Überschriften h1 bis h6 entsprechen. Eignet sich hervorragend, um aus längeren Texten Inhaltsverzeichnisse zu generieren.

Beispiel:

```
$(„div p:first“)
//folgendes Element wird ausgewählt:
      <div>
            <p>…</p>
            <p>…</p>
            <p>…</p>
      </div>
```

d)Formular-Filter

In Formularen ist es oft besonders wichtig, bestimmte Elementzustände abzufragen und zu verarbeiten. Insbesondere werden folgende Parameter oft verwendet:

:enabled
:disabled
:checked
:selected

7)Übung: Alle angewählten „checked“-Buttons zählen und live ausgeben

Erstelle die Datei „1.check.html“.

Ergebnis:

Erklärung:

Der Selektor ist hier „$("input:checked") der überprüft, ob Kontrollkästchen aktiviert wurden.

Code:

```
 8  <script>
 9 $(document).ready(function() {
10   function jogger() {
11     var n = $("input:checked").length;
12     $("div").text(n + (n <= 1 ? " is" : " Mal") + " insgesamt");
13   }
14   jogger();
15   $(":checkbox").click(jogger);
16 });
17 </script>
18 </head>
19 <body>
20 <h1>Wann gehst du laufen?</h1>
21     <span>
22       <form>
23         <p>Montag:<input type="checkbox" name="montag" value="mo"></p>
24         <p>Dienstag:<input type="checkbox" name="dienstag" value="di"></p>
25         <p>Mittwoch:<input type="checkbox" name="mittwoch" value="mi"></p>
26         <p>Donnerstag:<input type="checkbox" name="donnerstag" value="do"></p>
27         <p>Freitag:<input type="checkbox" name="freitag" value="fr"></p>
28         <p>Samstag:<input type="checkbox" name="samtag" value="sa"></p>
29         <p>Sonntag:<input type="checkbox" name="sonntag" value="so"></p>
30       </form>
31     </span>
32     <div></div>
33 </body>
```

```
<script>
$(document).ready(function() {
 function jogger() {
        var n = $("input:checked").length;
        $("div").text(n + (n <= 1 ? " is" : " Mal") + " insgesamt");
 }
 jogger();
 $(":checkbox").click(jogger);
});
</script>
</head>
<body>
<h1>Wann gehst du laufen?</h1>
  <span>
   <form>
        <p>Montag:<input type="checkbox" name="montag" value="mo"></p>
    <p>Dienstag:<input type="checkbox" name="dienstag" value="di"></p>
    <p>Mittwoch:<input type="checkbox" name="mittwoch" value="mi"></p>
    <p>Donnerstag:<input type="checkbox" name="donnerstag" value="do"></p>
    <p>Freitag:<input type="checkbox" name="freitag" value="fr"></p>
    <p>Samstag:<input type="checkbox" name="samtag" value="sa"></p>
    <p>Sonntag:<input type="checkbox" name="sonntag" value="so"></p>
   </form>
  </span>
  <div></div>
</body>
```

8)Eigenschaften auslesen

Eigenschaften und Methoden (object accessors) verändern nicht die Collection, sondern geben Informationen zurück.

- .each ist ein anonymes Funktionsobjekt, das beliebigen Inhalt besitzen kann und z.B. mit $(this) jeweils eine eigene Collection aus dem aktuellen verarbeiteten Element bildet.
- .length gibt eine Ganzzahl über die Anzahl der Items zurück

```
$(documten).ready(function () {
        var obj = $("#box p")
        obj.each(function) {
                $(this).text("obj hat" + obj.length + "Elemente");
        });   });
```

9)Übung: Radio-Buttons zählen mit „length“

Arbeiten mit einem Selektor: Erstelle folgende Datei namens „1.2.radio.html“.

Ergebnis:


```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>Radios zählen</title>
 <script src="js/jquery-3.0.0.js">
 </script>
 <script>
$(document).ready(function() {
  var input = $("form input:radio").wrap('<span></span>').parent().css({background:"blue"});
    $("div").text("Es wurden " + input.length + " Radioboxen gefunden").css("color", "blue");
    $("form").submit(function () { return false; });
});
</script>
</head>
<body>
<h1>Radiobuttons:</h1>
    <span>
      <form>
        <input type="radio" name="test">
        <input type="radio" name="test">
        <input type="radio" name="test">
      </form>
    </span>
    <div></div>
</body>
</html>
```

Code:

```
<head>
<meta charset="utf-8">
<title>Radios zählen</title>
 <script src="js/jquery-3.0.0.js">
 </script>
 <script>
$(document).ready(function() {
  var input = $("form input:radio").wrap('<span></span>').parent().css({background:"blue"});
          $("div").text("Es wurden " + input.length + " Radioboxen gefunden").css("color", "blue");
          $("form").submit(function () { return false; });
});
</script>
</head>
<body>
<h1>Radiobuttons:</h1>
   <span>
    <form>
          <input type="radio" name="test">
      <input type="radio" name="test">
      <input type="radio" name="test">
    </form>
   </span>
   <div></div>
</body>
```

Erklärung:

Der Selektor ist hier: $("form input:radio")
Es handelt sich dabei um ein Form-Element und darin gibt es „input“-Elemente vom Type „radio“.

```
<span>
  <form>
    <input type="radio" name="test">
    <input type="radio" name="test">
    <input type="radio" name="test">
  </form>
</span>
<div></div>
```

Nach den muss jedoch auch noch der -Bereich erwähnt werden, das der jQuery-Selektor in der Zeiler 11 dorthin auch Ausgabe legen will. Ohne dem würde nichts angezeigt werden.