JavaScript und JavaScript Frameworks

Manipulation des DOM

Übersicht und Zielstellung

- In diesem Abschnitt erfahren Sie:
  - wie das DOM aufgebaut ist,
  - was ein Knoten innerhalb des DOM ist,
  - wie man Elemente des DOM ermittelt
  - wie Elemente hinzugefügt werden,
  - wie Elemente verändert werden,
  - wie Elemente entfernt werden,

Start in das Thema

Das Document Object Model einer Webseite

Das **Document Object Model (DOM**) ist eine Programmierschnittstelle für Webdokumente, die es ermöglicht, auf den Inhalt und die Struktur einer Webseite zuzugreifen und sie zu manipulieren. Das DOM repräsentiert das HTML-Dokument als **eine Baumstruktur**, in der jedes Element als ein **Knoten (Node) dargestellt** wird.

Jeder Knoten im DOM hat **Eigenschaften**, die **Informationen** über das Element enthalten, wie z.B. den Namen, den Inhalt und die Attribute. Über das DOM können Entwickler auf diese Eigenschaften zugreifen und sie ändern, um die Darstellung der Webseite zu beeinflussen.

Zum Beispiel kann man mit dem DOM durch Skripting-Methoden wie JavaScript bzw. jQuery Elemente **hinzufügen**, **entfernen** oder **ändern**, ohne dass die zugrunde liegende HTML-Datei direkt bearbeitet werden muss.

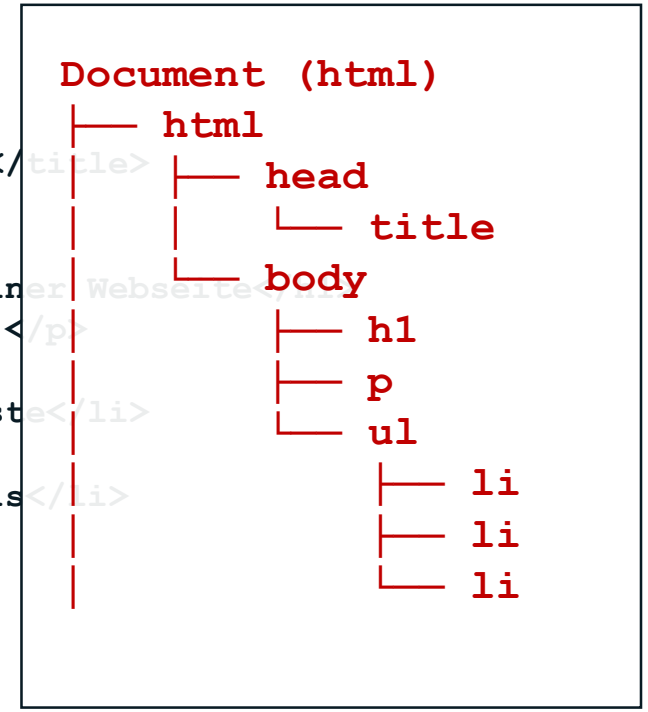
Das DOM ist somit ein wichtiges Konzept in der Webentwicklung, da es Entwicklern ermöglicht, auf dynamische Weise auf den Inhalt und die Struktur einer Webseite zuzugreifen und sie zu verändern.

Das Document Object Model einer Webseite

Der HTML Code

```
<!DOCTYPE html>
<html>
  <head>
    <title>Meine Webseite</title>
  </head>
  <body>
    <h1>Willkommen auf meiner Webseite</h1>
    <p>Ich bin ein Absatz.</p>
    <ul>
      <li>Ich bin eine Liste</li>
      <li>Ich auch</li>
      <li>Und ich ebenfalls</li>
    </ul>
  </body>
</html>
```

Das DOM für diese Webseite



Beschreibung

Das oberste Element des DOM-Baums ist das **Document-Element**, das die gesamte Webseite repräsentiert. Es enthält ein einzelnes **Kind-Element**, nämlich das **html-Element**.

Das **html-Element** hat **zwei Kind-Elemente**, das **head-Element** und das **body-Element**.

Das **head-Element** enthält **ein Kind-Element**, das **title-Element**, das den Titel der Webseite enthält.

Das **body-Element** hat **drei Kind-Elemente**: ein **h1-Element** mit der Überschrift, ein **p-Element** mit einem Absatztext und ein **ul-Element** mit einer Liste.

Das **ul-Element** hat wiederum **drei Kind-Elemente**, die **li-Elemente**, die die einzelnen Listenelemente repräsentieren.

Knoten hinzufügen

Die Methode .after()
Inhalt soll **nach** einem bestimmten Element eingesetzt werden.

Der Inhalt kann auch weitere HTML-Elementen enthalten.

Der betroffene HTML-Code

```
<h1 class="ziel">Das ist eine Überschrift</h1>
```

Der Query-Code

```
$(document).ready(function() {
   $('.ziel').after('<p>Das ist ein Absatz durch jQuery eingebunden.</p>');
});
```

An jedes selektierte Element mit dem Selektor **`.ziel`** wird ein Absatz angehängt, also hinter das schließende Tag **`</h1>`.**

Das Ergebnis im Seiten-Quelltext

```
<h1 class="ziel">Das ist eine Überschrift</h1><p> Das ist ein Absatz durch jQuery eingebunden.</p>
```

Knoten hinzufügen

Bereits existierende Elemente, können auch an ein anderes angehängt oder umgehängt werden.

Der betroffene HTML-Code

```
<h1>Das ist eine Überschrift</h1>
<p class="ziel">Das ist ein Absatz</p>
```

Der jQueryCode:

```
$(document).ready(function() {
    $('.ziel').after($('h1'));
});
```

Das Ergebnis:

```
<p class="ziel">Das ist ein Absatz</p>
<h1>Das ist eine Überschrift</h1>
```

Die Überschrift wurde nun an das Zielobjekt angehängt. Allerdings wird hier in der Methode .after() ein anderer Selektor $('h1‘) angehängt. Die Elemente werden im DOM neu sortiert. Das <h1>-Element wird nicht kopiert. Es wird ausgeschnitten und an der neuen Position angehängt.

Beispiel auf die Schnelle

Nach erfolgreichem Laden eines Dokumentes soll nach der bereits vorhandenen Überschrift folgenden Text in einem -Element hinter der Überschrift platziert werden.

" Dieser Absatz ist per jQuery als Knoten hinter der Überschrift eingefügt worden. "

1. Schritt: Legen Sie ein neues Beispiel Dokument an. jQueryDemo05.html

2. Schritt: Binden Sie jQuery in das Dokument ein.

3. Schritt: Erstellen Sie im <body> folgenden Inhalt.

```
<h1>Manipulation des DOM</h1>
```

4. Schritt: Ergänzen Sie nun im <head> den Code für das jQuery.

```
// Warten, bis das Dokument geladen ist
  $(document).ready(function() {
// Fügen Sie ein DOM-Element ein
      $("h1").after("<p>Dieser Absatz . . .!");
  });
```

Knoten hinzufügen

Methode .insertAfter()

Eine andere Methode ist .insertAfter(), die das gleiche Ergebnis liefert:

```
$(document).ready(function() {
    $('<p>Absatz</p>').insertAfter('.ziel');
});
```

Worin besteht nun der Unterschied zwischen .after() und .insertAfter()?

Die Methode .after() kann eine anonyme Funktion aufrufen und die Methode .insertAfter() kann das nicht?

```
$(document).ready(function() {
    var count = 0;
    $('.ziel').after(function() {
        count++;
        return '<p>Absatz mit dem Index: '+ count + 'wurde eingefügt</p>';
    });
});
```

Dazu wird der HTML-Code erweitert

```
<h1 class="ziel">Überschrift 1</h1>
<h2 class="ziel">Überschrift 2</h2>
```

Hier wird die Anzahl der hinzugefügten Absätze gezählt und mit angezeigt.
Die Variable **count** wird bei jeder Iteration über das Element mit der CSS-Klasse **ziel** um eins erhöht.
Die Variable **count** lassen Sie sich anschließend als Inhalt des <p>-Elements ausgeben.

Knoten hinzufügen

Methode .clone()
Wenn das Element, das Sie herauslösen und an anderer Stelle wieder einfügen wollen, nicht ausgeschnitten bzw. verschoben werden soll, sondern dupliziert und dann eingefügt werden soll, verwenden Sie die Methode **.clone().**

Der jQuery Code:

```
$(document).ready(function() {
    $('.ziel').clone().after($('h1'));
});
```

Hinweis: Der bereits benutzte HTML-Code kann auch hier verwendet werden.

Das Ergebnis im Quellcode:

```
<h1>Das ist eine Überschrift</h1>
<p class="ziel">Das ist ein Absatz</p>
<h1> Das ist eine Überschrift </h1>
```

Hinweis: Alle über Methode .data() hinzugefügten Daten werden automatisch mit dupliziert. Dazu später mehr!

Beispiel auf die Schnelle

Ein bereits vorhandener Absatz soll nach einer Überschrift als Kopie hinzugefügt werden.

1. Schritt: Legen Sie ein neues Beispiel Dokument an. jQueryDemo06.html

2. Schritt: Binden Sie jQuery in das Dokument ein.

3. Schritt: Erstellen Sie im <body> folgenden Inhalt.

```
<p>Das ist ein vorhandener Absatz</p>
<h1>Manipulation des DOM</h1>
```

4. Schritt: Ergänzen Sie nun im <head> den Code für das jQuery.

```
// Warten, bis das Dokument geladen ist
  $(document).ready(function() {
// Fügen Sie ein DOM-Element ein
      $("h1").clone().after($('p'));
  });
```

Knoten hinzufügen

Methoden .before() und .insertBefore()
Für .after() und .insertAfter() gibt es die selbstverständlich auch Varianten für die anderen Richtungen.

`.before() und .insertBefore()`

Der jQuery Code:

```
$(document).ready(function() {
    $('h1').before('<p>Dieeser Absatz wurde eingefügt</p>');
});
```

Hinweis: Der bereits benutzte HTML-Code kann auch hier verwendet werden.

Das Ergebnis:

```
<p class="ziel">Das ist ein Absatz</p>
<h1>Das ist eine Überschrift</h1>
```

Die Methode .before() kann eine anonyme Funktion aufrufen und die Methode .insertBefore() kann das nicht?

Kleine Übung

Erstellen Sie ein kleines Beispiel nach folgenden Vorgaben:

1. Aufgabe: Legen Sie ein neues Beispiel Dokument an. jQueryUebung05.html

2. Aufgabe: Legen Sie eine Überschrift an!
Erstellen Sie nach der Überschrift einen Absatz.

3. Aufgabe: Fügen Sie nun Elemente im DOM hinzu!
- hinter dem Absatz einen Div-Container mit dem Inhalt “Ich bin ein Container!“
- erstellen Sie eine Kopie des Absatzes und fügen Sie diese vor der Überschrift ein

Knoten hinzufügen

Methoden .append() und .appendTo()
Die Methoden **.append()** und **.appendTo()** erlauben es, weitere Elemente oder auch Inhalte **in ein** selektiertes Element einzufügen.

Folgender HTML-Code soll für einen Vergleich zur Methode .after() genutzt werden.

```
<div class="ziel">
   <p>Absatz</p>
</div>
```

Der jQuery Code:

```
$(document).ready(function() {
   $('.ziel').append('<p>Ein weiterer Absatz</p>');
});
```

Das Ergebnis bei der Verwendung von **.append()**:

```
<div class="ziel">
   <p>Absatz</p>
   <p>Ein weiterer Absatz</p>
</div>
```

Das Ergebnis bei der Verwendung von **.after()** wäre:

```
<div class="ziel">
   <p>Absatz</p>
</div>
<p>Ein weiterer Absatz</p>
```

Knoten hinzufügen

Anonyme Funktion in der Methode .append()
Wie bei .after() und .insertAfter() besteht auch bei .append() und .appendTo() der Unterschied darin, dass bei der Methode .append() auch wieder eine anonyme Funktion aufgerufen werden könnte.

Der HTML-Code:

```
<div class="ziel">
   <p>Absatz</p>
</div>
<div class="ziel">
   <p>Absatz</p>
</div>
```

Der jQuery-Code

```
$(document).ready(function() {
   var count = 0;
   $('.ziel').append(function() {
      count++;
      return '<p>Absatz mit dem Index: '+ count +'</p>';
   });
});
```

Das Ergebnis im Quell-Code

```
<div class="ziel">
   <p>Absatz</p>
   <p>Absatz mit dem Index 1</p>
</div>
<div class="ziel">
   <p>Absatz</p>
   <p>Absatz mit dem Index 2</p>
</div>
```

Knoten hinzufügen

Methoden .prepend() und .prependTo()
Die Methoden **.prepend ()** und **.prependTo()** erlauben es, Elemente oder auch Inhalte **in ein** selektiertes Element als erstes Elemente einzufügen.

Folgender HTML-Code soll für einen Vergleich zur Methode .after() genutzt werden.

```
<div class="ziel">
   <p>Absatz</p>
</div>
```

Der jQuery Code:

```
$(document).ready(function() {
   $('.ziel').prepend('<p>Ein weiterer Absatz</p>');
});
```

Das Ergebnis bei der Verwendung von .**prepend()**:

```
<div class="ziel">
   <p>Ein weiterer Absatz</p>
   <p>Absatz</p>
</div>
```

Das Ergebnis bei der Verwendung von **.append()**:

```
<div class="ziel">
   <p>Absatz</p>
   <p>Ein weiterer Absatz</p>
</div>
```

Beispiel auf die Schnelle

In einer Liste sollen zwei Elemente hinzugefügt werden.

1. Schritt: Legen Sie ein neues Beispiel Dokument an. jQueryDemo07.html

2. Schritt: Binden Sie jQuery in das Dokument ein.

3. Schritt: Erstellen Sie im <body> folgenden Inhalt.

```
<ul>
  <li>Das ist das mittlere Element</li>
</ul>
```

4. Schritt: Ergänzen Sie nun im <head> den Code für das jQuery.

```
// Warten, bis das Dokument geladen ist
  $(document).ready(function() {
// Fügen Sie ein DOM-Element ein
      $("ul").append('<li>Das ist das letzte Element</li>');
      $("ul").prepend ('<li>Das ist das erste Element</li>');
  });
```

Kleine Übung

Erstellen Sie ein kleines Beispiel nach folgenden Vorgaben:

1. Aufgabe: Legen Sie ein neues Beispiel Dokument an. jQueryUebung06.html

2. Aufgabe: Legen Sie im <body>-Element nichts an!

3. Aufgabe: Fügen Sie nun Elemente im DOM hinzu!
   - im <body> eine Überschrift h1 mit beliebigem Text
   - nach der Überschrift soll ein Div angelegt werden
   - im Div sollen zwei Absätze mit beliebigem Inhalt eingefügt werden

Knoten entfernen

Das Entfernen von Knoten ist mittels der bereitgestellten JavaScript-Funktionen möglich, aber gelegentlich knifflig.
Hier bietet jQuery auch wieder eigene Methoden an, die das Entfernen von Knoten vereinfachen sollen.

Ab hier wird der Code parallel in einem Sie ein neues Beispiel Dokument angelegt. jQueryDemo08.html

Methoden .remove() und .detach()
Diese Methoden machen, auf den ersten Blick, allerdings dasselbe – sie entfernen Elemente der Collection aus dem DOM.

Methode .remove()
… entfernt erwartungsgemäß DOM-Elemente.

Der HTML-Code:

```
<div id="box">
   <p id="ziel">Wird mit .remove() entfernt.</p>
</div>
```

Der jQuery-Code:

```
$(document).ready(function() {
   // Das Element mit der ID ziel ziel selektieren und entfernen
   $('#ziel').remove();
});
```

Knoten entfernen

Die Methode .remove() kann auch mit Übergabe eines Parameters angepasst werden.

Dazu kann als Parameter ein passender Selektor genutzt werden.

Der HTML-Code:

```
<div id="box">
   <p id="ziel">Wird mit .remove() entfernt.</p>
</div>
```

Nun kann der Selektor auf p eingestellt werden, was allerdings alle Absätze entfernen würde.

```
$(document).ready(function() {
   // Das Element mit der ID ziel ziel selektieren und entfernen
   $('p').remove();
});
```

Nun kann die ID als Parameter mit übergeben werden:

```
$(document).ready(function() {
   // Das Element mit der ID ziel ziel selektieren und entfernen
   $('p').remove('#ziel');
});
```

Knoten entfernen

Methode .detach()
Die Schwester-Methode .detach() entfernt zwar ebenfalls das Element oder die Elemente, behält für diese aber Daten- oder Event-Bindungen. Im Vergleich zum vorherigen Beispiel würde kein Unterschied erkennbar sein.

Betrachten wir also ein komplexeres Beispiel, bei dem beide Methoden Seite an Seite verglichen werden.

Es werden diesmal zwei Absätze benötigt, von denen einer detached und der andere removed wird. Anschließend werden zwei Button benötigt, der eine zum Entfernen und der andere zum Wiederherstellen (Rückgängig) machen.

```
<div id="container">
   <p id="ziel1">Wird removed. Der Absatz enthält Daten und ist anklickbar.</p>
   <p id="ziel2">Wird detached. Der Absatz enthält Daten und ist anklickbar.</p>
</div>
```

Nun noch die beiden Button

```
<button id="entf">Detach/Remove</button>
<button id="app">Re-Append</button>
```

Knoten entfernen

Teil 1 des jQuery-Codes

- die Elemente werden ermittelt,
- die „Ziel“-Elementen werden mit Daten versorgt,
- die „Ziel“-Elemente werden anklickbar gemacht,

```
$(document).ready(function() {
    var container = $("#container"), ziel1 = $("p#ziel1"), ziel2 = $("p#ziel2");
    // Daten binden:
    ziel1.data('daten','ziel1: Ich habe Daten!');
    ziel2.data('daten','ziel2: Ich habe Daten!');
    // Beide Absätze klickbar machen:
    $("p").click(function() {
        // gibt die Daten des aktuellen P aus
        alert($(this).data('daten'));
    });
});
```

Die Seite kann nun getestet werden. Hinweis: Die Button funktionieren noch nicht!

Knoten entfernen

Nun werden die Codes für die Funktionalität der Button angelegt. Zunächst der Button zum Entfernen der Elemente – einmal .remove() und einmal .detach() eingesetzt wird:

```
$("button#entf").click(function(){
        // beide Absätze entfernen:
        ziel1.remove();
        ziel2.detach();
});
```

Ein erneutes Einfügen wird über **.append()** am Container vorgenommen. Durch die am Anfang gebildeten Referenzen **ziel1** und **ziel2** besteht auf weiterhin Zugriff auf Textabsätze, auch wenn die entfernt wurden.

```
$("button#app").click(function() {
   // beide Absätze wieder einfü gen:
   ziel1.appendTo(container);
   ziel2.appendTo(container);
});
```

Die Seite kann nun erneut getestet werden. Klicken Sie zu erst die beiden Textabsätze an, entfernen Sie beide und stellen Sie die nach dem Entfernen wieder her. Klicken Sie nun die Absätze erneut an. Beurteilen Sie das Ergebnis.

Kleine Übung

Erstellen Sie ein kleines Beispiel nach folgenden Vorgaben:

1. Aufgabe: Legen Sie ein neues Beispiel Dokument an. jQueryUebung07.html

2. Aufgabe: Legen Sie im <body>-Element drei beliebige Absätze an und füllen Sie diese mit Text. Legen Sie nun Button an, um jeden einzelnen Absatz entfernen und wiederherstellen zu können.

3. Aufgabe: Erstellen Sie nun den jQuery-Code für die einzelnen Button, die genau das Entfernen und Wiederherstellen der Absätze übernehmen.

Knoten ersetzen

Neben dem Einfügen oder Entfernen von Knoten im DOM ist es oft auch sinnvoll, einen Knoten bzw. den Inhalt eines Knoten zu ersetzen:
Ab hier wird der Code parallel in einem Sie ein neues Beispiel Dokument angelegt. jQueryDemo09.html

Methode .replaceAll()
Mit der Methode .replaceAll() wird ein in einer Collection selektiertes Element ersetzt.

```
$(document).ready(function() {
    $("<h1>Neue Überschrift</h1>").replaceAll(".ziel");
});
```

Alle Elemente der Klasse **`ziel`** werden durch die aktuelle Collection ersetzt, die im übergebenen HTML-String erzeugt wird.

Aus dem ursprünglichen HTML-Code:

```
<div class="container">
    <p class="ziel">Ein Absatz</p>
    <p>Ein weiterer Absatz</p>
</div>
```

wird der HTML-Code:

```
<div class="container">
    <h1>Neue Überschrift</h1>
    <p>Ein weiterer Absatz</p>
</div>
```

Knoten ersetzen

Methode .replaceWith()
Die Methode .replaceWith() ist der Definition her anders als .replaceAll(), erzielt aber auf Wunsch dasselbe Ergebnis.

Die jQuery-Anweisung wird dazu modifiziert:

```
$(document).ready(function() {
    $(".ziel").replaceWith("<h1>Neue Überschrift</h1>");
});
```

Der Methode kann auch ein jQuery-Objekt als Parameter übergeben werden:

```
$(document).ready(function() {
    $(".ziel").replaceWith( $("div p:last") );
});
```

Im Ergebnis werden die beiden <p>-Elemente vertauscht. Der Quelltext sieht anschließend so aus:

```
<div class="container">
    <p>Ein weiterer Absatz</p>
    <p class="ziel">Ein Absatz</p>
</div>
```

Knoten ersetzen

Methode .replaceWith() und eine Funktion
Im Gegensatz zu .replaceAll() kann bei .replaceWith() eine Funktion übergeben werden:

Der jQuery-Code könnte so aussehen:

```
$(document).ready(function() {
    $(".ziel").replaceWith( function() {
        // Den String erzeugen ...
        var string = „"<h2>Das ist nur ein Beispiel</h2>";
        // und zurü ckgeben:
        return string;
    });
});
```

Kleine Übung

Erstellen Sie ein kleines Beispiel nach folgenden Vorgaben:

1. Aufgabe: Legen Sie ein neues Beispiel Dokument an. jQueryUebung08.html

2. Aufgabe: Legen Sie im <body>-Element eine Überschrift 1 und einen Absatz mit Inhalt

3. Aufgabe: Erstellen Sie nun den jQuery-Code für folgende Zielstellung:
   - die Überschrift 1 soll durch eine Überschrift 2 und anderem Text ausgetauscht werden
   - der Absatz soll durch eine Überschrift 3 mit entsprechendem Inhalt ausgetauscht werden

.

Wrapping

Wrapping-Methoden
Mit den folgenden Methoden können vorhandene, im Rahmen einer Collection selektierte Elemente jeweils außen, innen oder in ihrer Gesamtheit mit erstelltem Inhalt umschlossen (wrappen) werden.

Ab hier wird der Code parallel in einem Sie ein neues Beispiel Dokument angelegt. jQueryDemo10.html

Umschließen
Die Methoden .wrap(), .wrapInner() und .wrapAll() erlauben das Umschließen von selektierten Elementen.

Umschließende Elemente entfernen
Im Gegenzug zum Umschließen können umschließende Elemente auch entfern werden.
Dafür steht die Methode .unwrap() zur Verfügung.

Methode .wrap()
Die konzeptionell einfachste und naheliegendste Art, ein Element zu wrappen, besteht darin, es mit einer HTML-Struktur zu umgeben.

```
<div class="container">
   <p>Ein Absatz</p>
   <p>Ein weiterer Absatz</p>
</div>
```

Die beiden Absätze im Inneren des Containers sind mit einem Div-Elemente umschlossen.

Wrapping

Methode .wrap()
ein einfacher HTML-Code für das Beispiel:

```
<div class="container">
   <p>Ein Absatz</p>
   <p>Ein weiterer Absatz</p>
</div>
```

Die beiden Textabsätze im Inneren des Containers sollen mit Div-Element der Klasse .wrap umschlossen (gewrappt) werden.
Das ist der jQuery Code:

```
$(document).ready(function() {
   // In der Collection sind zwei Textabsätze:
   $("p").wrap("<div class='wrap'></div>");
});
```

Diese Zeilen bescheren folgendes Ergebnis:

```
<div class="container">
   <div class="wrap">
      <p>Ein Absatz</p>
   </div>
   <div class="wrap">
      <p>Ein weiterer Absatz</p>
   </div>
</div>
```

Im Ergebnis wird jedes einzelne der selektierten Elemente für sich mit dem neuen Element umhüllt.

Wrapping

Methode .wrapInner()
Diese Methode arbeitet anders als die Methode .wrap(). Umschlossen wird hier der Inhalt des selektierten Elementes.

Der Wrap findet also in den Elementen der Collection statt.

Gehen wir vom gleichen HTML-Gerüst aus:

```
<div class="container">
   <p>Ein Absatz</p>
   <p>Ein weiterer Absatz</p>
</div>
```

Die jQuery-Code:

```
$(document).ready(function() {
   // In der Collection sind wieder zwei Textabsätze:
   $("p").wrapInner("<div class='wrap'></div>");
});
```

Das Ergebnis:

```
<div class="container">
   <p><div class="wrap">Ein Absatz</div></p>
   <p><div class="wrap">Ein weiterer Absatz</div></p>
</div>
```

Wrapping

Achtung bei der Anwendung von .warpInner()
Für den Einsatz von wrapInner() sollte man immer die benutzte Collection beachten:

Nutzt man die p-Elemente für die Selektierung ergibt das folgendes Ergebnis

```
<div class="container">
   <p><div class="wrap">Ein Absatz</div></p>
   <p><div class="wrap">Ein weiterer Absatz</div></p>
</div>
```

Nutzt man die Klasse container für die Selektierung ergibt das folgenden jQuery-Code:

```
$(document).ready(function() {
   $(".container").wrapInner('<div class="wrap"></div>');
});
```

Das Ergebnis:

```
<div class="container">
   <div class="wrap">
      <p>Ein Absatz</p>
      <p>Ein weiterer Absatz</p>
   </div>
</div>
```

Wrapping

Methode .wrapAll()
Wurde im vorherigen Beispiel das Umschließen aller p Elemente durch die Selektion der Klasse container umgesetzt, ist das leider nicht immer möglich, wenn zum Beispiel der Div mit der Klasse container nicht vorhanden ist.:

Im Gegensatz zu der Methode .wrap(), die sich einzeln auf jedes Element der Collection bezieht, wird die Methode .wrapAll() alle selektierten Elemente komplett umhüllen.

Der HTML-Code

```
<div class="container">
   <p>Ein Absatz</p>
   <p>Ein weiterer Absatz</p>
</div>
```

Der jQuery-Code

```
$(document).ready(function() {
   $("p").wrapAll('<div class="wrap"></div>');
});
```

Das Ergebnis

```
<div class="container">
   <div class="wrap">
      <p>Ein Absatz</p>
      <p>Ein weiterer Absatz</p>
   </div>
</div>
```

Der Wrapper kann auch verschachtelt sein

```
$(document).ready(function() {
   $("p").wrapAll(
     '<div class="container"><div class="wrap"></div></div>'
   );
});
```

Wrapping

Die Methode **.unwrap()** bedeutet:
»Alle Parent-Element jedes Elements der Collection werden entfernt. « Dies gilt auch, wenn sich alle oder mehrere Elemente der Collection ein Parent-Element teilen – wie in diesem Fall.

```
<div class="container">
   <div class="wrap">
      <p>Ein Absatz</p>
   </div>
   <div class="wrap">
      <p>Ein weiterer Absatz</p>
   </div>
</div>
```

Der jQuery-Code

```
$(document).ready(function() {
   $("p").unwrap();
});
```

```
<div class="container">
   <div class="wrap">
      <p>Ein Absatz</p>
      <p>Ein weiterer Absatz</p>
   </div>
</div>
```

Das Ergebnis für beide Fälle

```
<div class="container">
   <p>Ein Absatz</p>
   <p>Ein weiterer Absatz</p>
</div>
```

Wrapping

Die Methode **.unwrap()** einsetzen und eine Hierarchie-Ebene nach oben überspringen bedeutet:
Im Beispiel soll das Div Element mit der Klasse container entfernt werden – dafür wird eine Hierarchie-Ebene überprungen

```
<div class="container">
   <div class="wrap">
      <p>Ein Absatz</p>
   </div>
   <div class="wrap">
      <p>Ein weiterer Absatz</p>
   </div>
</div>
```

Der jQuery-Code

```
$(document).ready(function() {
   $("p").parent()unwrap();
});
```

```
<div class="container">
   <div class="wrap">
      <p>Ein Absatz</p>
      <p>Ein weiterer Absatz</p>
   </div>
</div>
```

Das Ergebnis für beide Fälle

```
<div class="wrap">
   <p>Ein Absatz</p>
   <p>Ein weiterer Absatz</p>
</div>
```

Hinweis: Das funktioniert unabhängig davon, ob die Textabsätze identisch oder individuell gewrappt sind.

Wrapping – kleine Übung

Die Methode **.unwrap()** einsetzen und eine Hierarchie-Ebene nach oben überspringen bedeutet:
Im Beispiel soll das Div Element mit der Klasse container entfernt werden – dafür wird eine Hierarchie-Ebene überprungen

```
<div class="container">
   <div class="wrap">
      <p>Ein Absatz</p>
   </div>
   <div class="wrap">
      <p>Ein weiterer Absatz</p>
   </div>
</div>
```

```
<div class="container">
   <div class="wrap">
      <p>Ein Absatz</p>
      <p>Ein weiterer Absatz</p>
   </div>
</div>
```

Der jQuery-Code

```
$(document).ready(function() {
   $("p").parent()unwrap();
});
```

Das Ergebnis für beide Fälle

```
<div class="wrap">
   <p>Ein Absatz</p>
   <p>Ein weiterer Absatz</p>
</div>
```

Hinweis: Das funktioniert unabhängig davon, ob die Textabsätze identisch oder individuell gewrappt sind.

Wrapping

Methode .unwrap()
Das unmittelbar umschließende Element der Collection kann mit dieser Methode entfernt werden:

Folgender HTML Code liegt vor:

```
<div class="container">
   <div class="wrap">
      <p>Ein Absatz</p>
   </div>
   <div class="wrap">
      <p>Ein weiterer Absatz</p>
   </div>
</div>
```

Der jQuery-Code (die Methode .unwrap() nimmt keinen Parameter entgegen):

```
$(document).ready(function() {
   $("p").unwrap();
});
```

Kleine Übung

Erstellen Sie ein kleines Beispiel nach folgenden Vorgaben:

1. Aufgabe: Legen Sie ein neues Beispiel Dokument an. jQueryUebung09.html

2. Aufgabe: Legen Sie im <body>-Element einen Div-Container an und geben Sie ihm die id="wrapper". Erstellen Sie in diesem Container eine Überschrift h1 und eine ungeordnete Liste mit 5 Elementen.

3. Aufgabe: Erstellen Sie nun den jQuery-Code für folgende Zielstellung:
   - Entfernen Sie die Div-Container
   - Entfernen Sie die ungeordnete Liste
   - umschließen Sie die Listenelemente mit einer geordneten Liste
   - umschließen Sie die Überschrift mit einem <header>
   - umschließen Sie die Liste mit einem <main>

VIELEN DANK!