Kostenloses eBook

LERNEN

ajax

Free unaffiliated eBook created from
Stack Overflow contributors.

#ajax

Inhaltsverzeichnis

Über ... 1

Kapitel 1: Erste Schritte mit Ajax ... 2

Bemerkungen ... 2

Examples ... 2

Installation oder Setup ... 2

Was ist AJAX? ... 2

Was ist jQuery? ... 2

So fügen Sie Ihrer Website jQuery hinzu ... 3

Einfaches jQuery-Beispiel für die Kommunikation mit dem Server ... 3

Sync - Async-Ajax-Anforderungen ... 4

Einfache Ajax-Anforderung, die mit dem XMLHttpRequest-Objekt gesendet wurde ... 5

Verwendung von Ajax in Vanilla Javascript mit einem einfachen Rückruf ... 6

Durchführen eines asynchronen AJAX-Aufrufs mit TypeScript ... 7

Produkt einem Warenkorb hinzufügen ... 7

Kapitel 2: Rückrufe ... 10

Examples ... 10

Fehler werden mit "error" Callback interpretiert ... 10

Credits ... 12

Über

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: ajax

It is an unofficial and free ajax ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official ajax.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Kapitel 1: Erste Schritte mit Ajax

Bemerkungen

AJAX **(ein** synchroner **J** avascript nd **X** ML) können Sie externe Daten verlangen , **ohne** dass die Ausführung von Code zu blockieren. In vielen Fällen wird dies implementiert, indem Teile einer Seite oder Informationen von einem Server (über XMLhttpRequests) angefordert und anschließend mit Javascript verarbeitet und angezeigt werden.

Der nicht blockierende Charakter von AJAX macht es zu einem so weit verbreiteten Softwaremuster. Da Javascript im Browser blockiert, führt ein synchroner externer Anruf dazu, dass der Browser für die Dauer des Anrufs nicht reagiert, bis er Daten zurückgibt oder das Zeitlimit überschritten hat. Tatsächlich wird Ihre Anwendung vollständig von der externen Architektur und deren Leistungsfähigkeit abhängig.

AJAX-Aufrufe werden normalerweise abstrahiert, um zusätzliche Funktionalität oder Lesbarkeit bereitzustellen. Die Implementierungen basieren jedoch (normalerweise) auf der XMLHttpRequest- Spezifikation .

Examples

Installation oder Setup

Was ist AJAX?

> AJAX steht für Asynchronous JavaScript und XML. Kurz gesagt, es ist die Verwendung des XMLHttpRequest-Objekts für die Kommunikation mit serverseitigen Skripts. Es kann Informationen in verschiedenen Formaten senden und empfangen, darunter JSON, XML, HTML und sogar Textdateien. -Mozilla Developer Network 2016

Die einfachste Möglichkeit, AJAX zu implementieren, insbesondere wenn Sie die Kommunikation mit Servern planen, ist die Verwendung von jQuery.

Was ist jQuery?

> jQuery ist eine schnelle, kleine und funktionsreiche JavaScript-Bibliothek. Es vereinfacht das Durchqueren und Manipulieren von HTML-Dokumenten, die Ereignisbehandlung, Animation und Ajax mit einer benutzerfreundlichen API, die für eine Vielzahl von Browsern geeignet ist. -jquery.com

Für diejenigen, die nicht viel jQuery verwendet haben, stellen Sie sich diese Funktionen als Funktionen vor, mit denen wir unser Leben einfacher machen können. Dies ist perfekt für die Verwendung mit AJAX, da die Menge an Code reduziert wird, den wir schreiben müssen, um

dasselbe zu erreichen!

So fügen Sie Ihrer Website jQuery hinzu

Wenn Sie Ajax verwenden müssen, müssen Sie jQuery zu Ihrem Projekt hinzufügen. http://jquery.com/download/ In diesem Link können Sie viele Möglichkeiten zum Hinzufügen von Jquery sehen. Sie können eine heruntergeladene Version von jQuery oder ein CDN verwenden. http://jquery.com/download/#jquery-39-s-cdn-provided-by-maxcdn . Es besteht jedoch ein gewisses Sicherheitsrisiko, wenn Sie CDN verwenden. Da das Projekt JQuery verwendet, kann ein Hacker den Aufruf manipulieren. Also besser, wenn Sie die heruntergeladene Version verwenden könnten. Hier erfahren Sie, wie Sie JQuery zum HTML-Projekt hinzufügen. Es ist einfach. Das erste Beispiel ist die Verwendung einer heruntergeladenen Quelle. Verwenden Sie diesen Link, um http://jquery.com/download/#jquery herunterzuladen. Wenn Sie nur jquery verwenden möchten, empfehle ich den Download. **Laden Sie die komprimierte, jQuery 3.1.1-** Datei herunter. Wenn Sie es herunterladen, fügen Sie jquery-version.min.js an der entsprechenden Stelle hinzu (z. B. im JavaScript-Ordner Ihres Projekts). Fügen Sie einfach einen Tag hinzu mit src = jquery / location wie unten.

```
<head>

<script src="path/from/html/page/to/jquery.min.js"></script>

</head>
```

Mal sehen, wie man ein CDN verwendet. Über diesen Link http://jquery.com/download/#using-jquery-with-a-cdn können Sie verschiedene CDN (Content Delivery Network) sehen.

```
<head>

<script   src="https://code.jquery.com/jquery-3.1.1.min.js"   integrity="sha256-
hVVnYaiADRTO2PzUGmuLJr8BLUSjGIZsDYGmIJLv2b8="   crossorigin="anonymous"></script>

</head>
```

Wie Sie in sehen können, müssen Sie nur die Tags hinzufügen, die der CDN-Provider zur Verfügung stellt. Fügen Sie der HTML-Seite nun einige Skripts hinzu, um zu überprüfen, ob sie funktionieren.

```
<script>
$(document).ready(function(){
    alert("jQuery Works")
});
</script>
```

Wenn Sie den Alarm " **jQuery Works" sehen** , bedeutet dies, dass Sie ihn korrekt hinzugefügt haben.

Einfaches jQuery-Beispiel für die Kommunikation mit dem Server

Aus der `jQuery.ajax` API -Website entnommen:

```
$.ajax({
    method: "POST",
    url: "some.php",
    data: {
        name: "John",
        location: "Boston"
    },
    success: function(msg) {
        alert("Data Saved: " + msg);
    },
    error: function(e) {
        alert("Error: " + e);
    }
});
```

Dieser Codeabschnitt ist aufgrund von jQuery leicht zu lesen und zu verstehen, was los ist.

- `$.ajax` - Dieses Bit ruft die `ajax` Funktionalität von jQuery auf.
- `method: "POST"` - diese Zeile gibt an, dass wir zur Kommunikation mit dem Server eine POST-Methode verwenden werden. Informieren Sie sich über Arten von Anfragen!
- `url` - Diese Variable gibt an, wohin die Anforderung **gesendet werden** soll. Sie eine Anfrage **an** irgendwo zu senden. Das ist die Idee.
- `data` - ziemlich einfach. Dies sind die Daten, die Sie mit Ihrer Anfrage senden.
- `success` - Mit dieser Funktion schreiben Sie, um zu entscheiden, was Sie mit den Daten tun sollen, die Sie zurückerhalten. `msg` ! Wie das Beispiel zeigt, wird derzeit lediglich eine Warnung mit der zurückgegebenen `msg` .
- `error` - Diese Funktion schreibt hier, um Fehlermeldungen anzuzeigen oder Aktionen bereitzustellen, wenn die `ajax` Anforderung fehlerhaft war.
- Eine Alternative zu `.done` ist

  ```
  success: function(result) {
      // do something
  });
  ```

Sync - Async-Ajax-Anforderungen

Asynchroner Ajax-Aufruf

Bei dieser Art von Ajax-Anruf wartet der Code nicht auf den Abschluss des Anrufs.

```
$('form.ajaxSubmit').on('submit',function(){
    // initilization...
    var form   = $(this);
    var formUrl    = form.attr('action');
    var formType   = form.attr('method');
```

```
    var formData   = form.serialize();

    $.ajax({
        url: formUrl,
        type: formType,
        data: formData,
        async: true,
        success: function(data) {
            // .....
        }
    });
    //// code flows through without waiting for the call to complete
    return false;
});
```

Synchroner Ajax-Aufruf

Bei dieser Art von Ajax-Anruf wartet der Code auf den Abschluss des Anrufs.

```
$('form.ajaxSubmit').on('submit',function(){
    // initilization...
    var form   = $(this);
    var formUrl    = form.attr('action');
    var formType   = form.attr('method');
    var formData   = form.serialize();
    var data = $.ajax({
            url: formUrl,
            type: formType,
            data: formData,
            async: false
        }).responseText;
    //// waits for call to complete
    return false;
});
```

Einfache Ajax-Anforderung, die mit dem XMLHttpRequest-Objekt gesendet wurde

```
var httpRequest = new XMLHttpRequest();

httpRequest.onreadystatechange = getData;

httpRequest.open('GET', 'https://url/to/some.file', true);
httpRequest.send();

function getData(){
    if (httpRequest.readyState === XMLHttpRequest.DONE) {
        alert(httpRequest.responseText);
    }
}
```

`new XMLHttpRequest()` erstellt ein neues *XMLHttpRequest*- Objekt. Dies ist, womit wir unsere Anfrage senden

Das `onreadystatechange` Bit teilt unserer Anfrage mit, bei jeder Änderung des Status `getData()` aufzurufen

`.open()` erstellt unsere Anfrage - dies `.open()` eine *Anforderungsmethode* (' **GET** ', ' **POST** ' usw.), eine URL der **abgefragten** Seite und optional, ob die Anfrage **asynchryn** sein soll oder nicht

`.send()` sendet unsere Anfrage - dies akzeptiert optional Daten, die an den Server gesendet werden sollen, wie `.send(data)`

Schließlich ist `getData()` die Funktion, von der wir sagten, dass sie jedes Mal aufgerufen werden sollte, wenn sich der **Status** unserer Anfrage **ändert** . Wenn der readyState gleich **DONE** ist, wird der `responseText` **gemeldet,** `responseText` dem es sich nur um die vom Server erhaltenen Daten handelt.

Weitere Informationen finden Sie im Leitfaden zum Einstieg in MDN.

Verwendung von Ajax in Vanilla Javascript mit einem einfachen Rückruf

Hier ist unsere Funktion zum Erstellen eines einfachen Ajax-Aufrufs, der in Vanilla-Javascript (nicht es2015) geschrieben ist:

```
function ajax(url, callback) {
    var xhr;

    if(typeof XMLHttpRequest !== 'undefined') xhr = new XMLHttpRequest();
    else {
        var versions = ["MSXML2.XmlHttp.5.0",
                        "MSXML2.XmlHttp.4.0",
                        "MSXML2.XmlHttp.3.0",
                        "MSXML2.XmlHttp.2.0",
                        "Microsoft.XmlHttp"]

         for(var i = 0, len = versions.length; i < len; i++) {
            try {
                xhr = new ActiveXObject(versions[i]);
                break;
            }
            catch(e){}
         } // end for
    }

    xhr.onreadystatechange = ensureReadiness;

    function ensureReadiness() {
        if(xhr.readyState < 4) {
            return;
        }

        if(xhr.status !== 200) {
            return;
        }

        // all is well
        if(xhr.readyState === 4) {
            callback(xhr);
        }
    }

    xhr.open('GET', url, true);
```

```
    xhr.send('');
}
```

und es könnte verwendet werden als:

```
ajax('myFile.html', function(response) {
    document.getElementById('container').innerHTML = response.responseText;
});
```

Wenn Sie ECMAScript 6 (auch bekannt als es2015) verwenden möchten Sie die Methode **holen** können, die ein Versprechen zurückgibt:

```
fetch('myFile.json').then(function(res){
    return res.json();
 });
```

Für weitere Informationen über es2015 Promises folgen Sie dem Link unten: Promises

Durchführen eines asynchronen AJAX-Aufrufs mit TypeScript

Produkt einem Warenkorb hinzufügen

Das folgende Beispiel zeigt, wie Sie einer Datenbanktabelle mithilfe von AJAX und TypeScript asynchron ein Produkt (oder etwas anderes) hinzufügen.

```
declare var document;
declare var xhr: XMLHttpRequest;

window.onLoad = () =>
{
    Start();
};

function Start()
{
    // Setup XMLHttpRequest (xhr).
    if(XMLHttpRequest)
    {
        xhr = new XMLHttpRequest();
    }
    else
    {
        xhr = new ActiveXObject("Microsoft.XMLHTTP");
    }

    AttachEventListener(document.body, "click", HandleCheckBoxStateChange);
}

function AttachEventListener(element: any, e, f)
{
    // W3C Event Model.
    if(element.addEventListener)
    {
```

```
        element.addEventListener(e, f, false);
    }
    else if(element.attachEvent)
    {
        element.attachEvent("on" + e, (function(element, f)
        {
            return function()
            {
                f.call(element, window.event);
            };
        })
        (element, f));
    }

    element = null;
}

function HandleCheckBoxStateChange(e)
{
    var element = e.target || e.srcElement;

    if(element && element.type == "checkbox")
    {
        if(element.checked)
        {
            AddProductToCart(element);
        }
        else
        {
            // It is un-checked.
            // Remove item from cart.
        }
    }
    else
    {
        break;
    }
}

AddProductToCart(e)
{
    var element = <HTMLInputElement>document.getElementById(e.id);

    // Add the product to the Cart (Database table)
    xhr.onreadystatechange = function()
    {
        if(xhr.readyState == 4)
        {
            if(xhr.status == 200)
            {
                if(element != null)
                {
                    console.log("200: OK");
                }
                else
                {
                    console.log(":-(");
                }
            }
            else
            {
```

```
                // The server responded with a different response code; handle accordingly.
                // Probably not the most informative output.
                console.log(":-(");
            }
        }
    }

    var parameters = "ProductID=" + encodeURIComponent(e.id) + "&" + "Action=Add&Quantity=" +
element.value;

    xhr.open("POST", "../Cart.cshtml");
    xhr.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
    xhr.setRequestHeader("Content-length", parameters.length.toString());
    xhr.setRequestHeader("Connection", "close");

    xhr.send(parameters);

    return e.id;
}
```

Um dieses Beispiel zu vervollständigen, aktualisieren Sie Ihren serverseitigen Code, um diese Daten tatsächlich in die Datenbank einzufügen. Der obige Code setzt voraus, dass Sie C # verwenden und über eine `Cart.cshtml` Datei verfügen. Ersetzen `cshtml` jedoch einfach `cshtml` durch `php` und schreiben Sie Ihre eigene serverseitige Logik in der Sprache Ihrer Wahl.

Erste Schritte mit Ajax online lesen: https://riptutorial.com/de/ajax/topic/1082/erste-schritte-mit-ajax

Kapitel 2: Rückrufe

Examples

Fehler werden mit "error" Callback interpretiert

Fehler, die vom Server ordnungsgemäß verwaltet werden, werden mit einem bestimmten HTTP-Statuscode, der sich von 2xx unterscheidet, an Ihren Client zurückgesandt (siehe RFC 2616, Abschnitt 10).

Es wird empfohlen, Ihre Fehler global von `$.ajaxSetup()` wie im nachstehenden Beispiel gezeigt. Daher werden alle Fehler, die von Ihren Ajax-Aufrufen stammen, automatisch vom Ajax-Setup interpretiert.

```
$.ajaxSetup({
    error: function (jqXHR, exception, errorThrown) {
        var message;
        var statusErrorMap = {
            '400': "Server understood the request, but request content was invalid.",
            '401': "Unauthorized access.",
            '403': "Forbidden resource can't be accessed.",
            '500': "Internal server error.",
            '503': "Service unavailable."
        };
        if (jqXHR.status) {
            message = statusErrorMap[jqXHR.status];
            if (!message) {
                message = "Unknown Error.";
            }
        } else if (exception == 'parsererror') {
            message = "Error.\nParsing JSON Request failed.";
        } else if (exception == 'timeout') {
            message = "Request Time out.";
        } else if (exception == 'abort') {
            message = "Request was aborted by the server";
        } else {
            message = "Unknown Error.";
        }

        // How you will display your error message...
        console.log(message);
        console.log(errorThrown);
    }
});
```

Möglicherweise möchten Sie auch den `error` in einem bestimmten `$.ajax()` "überladen", wenn Sie auf eine bestimmte Fehlermeldung warten.

```
$.ajax({
    url: './api',
    data: { parametersObject },
    type:'post',
    dataType: 'json',
```

```
    success:function(output){
        // Interpret success
    },
    error: function(xhr,textStatus,ErrorThrown){
        // Specific error will not be interpreted by $.ajaxSetup
    }
});
```

Rückrufe online lesen: https://riptutorial.com/de/ajax/topic/7175/ruckrufe

Credits

S. No	Kapitel	Contributors
1	Erste Schritte mit Ajax	acupajoe, adielhercules, Alessio Cantarella, Community, delete me, JhWebDevGuy, jignesh prajapati, MAZux, Menuka Ishan, Nicholas Qiao, TimTheEnchanter, Tolga Evcimen
2	Rückrufe	maxime_039