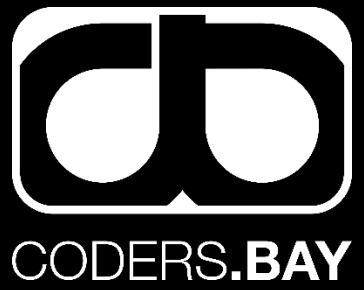


JAVASCRIPT – PHP KOMMUNIKATION

JSON

- Die **JavaScript Object Notation** (**JSON**) ist ein kompaktes Datenformat in einer einfach lesbaren Textform und dient dem Zweck des **Datenaustausches** zwischen Anwendungen.
- TEXT / STRING
- JSON ist von der Programmiersprache unabhängig.
- Parser und Generatoren existieren in allen verbreiteten Sprachen.

DATENTYPEN

JSON kenn folgende **Typen** von Elementen

- Nullwert (null)
- Boolscher Wert
- Zahl
- Zeichenkette
  - Eine Zeichenkette beginnt und endet mit doppelten Anführungszeichen (“)
  - Diesmal spielt das Anführungszeichen eine Rolle!
- Array
- Objekt

JSON - Beispiel

CODERS.BAY

```
{
  "Firma": "CodersBay",
  "Ort": "Linz",
  "Kursleiter": {
    "Name": "Mustermann",
    "Vorname": "Max",
    "maennlich": true,
    "Hobbys": ["Reiten", "Golfen", "Lesen"],
    "Alter": 42,
    "Kinder": [],
    "Partner": null
  }
}
```

JSON Parsen

CODERS.BAY

Javascript

- Von JSON zu einem Objekt

```
let json = '{"result":true, "count":42}';
let obj = JSON.parse(json);
```

- Von Objekt / Array zu einem JSON

```
let obj = {"result":true, "count":42};
let json = JSON.stringyfy(obj);
```

JSON Parsen

CODERS.BAY

PHP

- Von JSON zu einem Objekt

```
$json = '{"result":true, "count":42}';
$obj = json_decode($json); //object
$array = json_decode($json, true); //array
```

- Von Objekt / Array zu einem JSON

```
$obj = {"result":true, "count":42};
$json = json_encode($obj);
```

JSON Parsen

CODERS.BAY

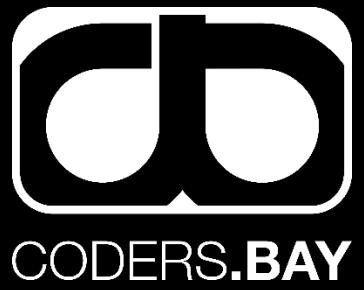
PHP

- Von JSON zu einem Objekt

```
$json = '{"result":true, "count":42}';
$obj = json_decode($json); //object
$array = json_decode($json, true); //array
```

- Von Objekt / Array zu einem JSON

```
$obj = {"result":true, "count":42};
$json = json_encode($obj);
```



AJAX

AJAX

CODERS.BAY

- Mit Ajax kann man Anfragen an den Server asynchron ausführen, das heißt im Hintergrund.
- Ajax steht für **Asynchronous JavaScript And XML**. XML, weil als Dateiformat für die Antwort oft mit XML gearbeitet wurde. Der Begriff Ajax wird heute weiterverwendet, weil er kurz ist - auch wenn man meist nicht mehr XML als Rückgabeformat einsetzt, sondern beispielsweise JSON (*JavaScript Object Notation*) oder direkt Text.
- Das Entscheidende an Ajax ist der erste Bestandteil der ausgeschriebenen Notation - der Begriff "asynchron". Bei der asynchronen Kommunikation erfolgt "das Senden und Empfangen von Daten zeitlich versetzt und ohne Blockieren des Prozesses durch bspw. Warten auf die Antwort des Empfängers".

EINSATZ

CODERS.BAY

Setzt man Ajax ein, so kann das folgendermaßen ablaufen:

Der Benutzer gibt etwas in das Formular ein oder wählt etwas aus einer Auswahlliste. Wenn er das getan hat, findet im Hintergrund eine Anfrage an den Server statt. Der Server liefert die Antwort. Währenddessen ist die Seite mit dem Formular im Browser aber nicht verschwunden (wie im klassischen Fall), sondern der Benutzer kann weiter an ihr arbeiten. Und wenn Antworten vom Server eintreffen, können diese direkt in die vorhandene Seite mit dem Formular eingefügt werden.

Das Ganze funktioniert über das **asynchrones JavaScript**.

- User Aktion
- Serververbindung aufbauen
- Serververbindung schließen
- User via Javascript neue Informationen geben

Arten

CODERS.BAY

jQuery Funktionen

- $.get
- $.post
- $.ajax

$.AJAX

CODERS.BAY

```
$.ajax({
   type: "POST",
   data: myData,                    //JSON
   url: 'delete_entry.php',
   success: function(phpData) {
      console.log(phpData)
   },
   error: function(error) {
      console.log(error);
   }
});
```

Asynchron Inhalte mit GET versenden

CODERS.BAY

Es gibt ein Formular mit einer Auswahlliste. Wenn der Benutzer etwas auswählt, werden die zu seiner Auswahl passenden Inhalte auf der Seite dargestellt. Das Entscheidende: Das Ganze funktioniert, ohne dass ein Absende-Button gedrückt wurde; auch wird die Seite nicht neu geladen. Die Datenübertragung zwischen Client und Server findet im Hintergrund statt.

Im div-Element unterhalb des Formulars sollen im Beispiel die Ergebnisse der Abfrage angezeigt werden.

```
<form>
    <p>
        <label for="art">Pflanzenart</label>:
        <selet id="art" name="art">
            <option value="">wähle</option>
            <option value="Nutzpflanze">Nutzpflanze</option>
            <option value="Zierpflanze">Zierpflanze</option>
        </selet>
    </p>
</form>
<div id="ausgabe"></div>
```

In diesem Beispiel wird der Eventhandler `change` verwendet. Damit reagieren wir darauf, dass eine Änderung beim Auswahlfeld mit `id="art"` ausgeführt wird.

Zuerst wird überprüft, ob der aktuelle Wert nicht gleich dem Leerstring ist. `val()` ist ein jQuery-Befehl, um den Wert (Inhalt) von Formularfeldern auszulesen.

Die Aktion wird ausgeführt, wenn es zur Überprüfung kommt mit `if($(this).val() != "")`
Nur wenn eine Option mit einem Wert für `value` ausgewählt wurde, wird der Code ausgeführt

```
$('#art').on('change', function() {
    if($(this).val != "") {
        $.get(
            "daten.php",
            { wahl: $(this).val() },
            function(daten) {
                $('#ausgabe').html(daten);
            }
        );
    }
});
```

Asynchron Inhalte mit GET versenden

CODERS.BAY

`$.get()` ist eine jQuery-Methode, um GET-Requests asynchron durzuführen. Sie erwartet mehrere Parameter:

- Zuerst den Namen des Skripts, das asynchron aufgerufen werden soll. Im Beispiel heißt es *daten.php*
- Als weiten Parameter können Daten übergeben werden, die an das Skript übergeben werden sollen. Im Beispiel wird ein Parameter mit dem Namen `wahl` übergeben
- Als dritten Parameter gibt es eine Funktion, die aufgerufen wird, wenn die Antwort des Servers eintrifft. Diese Funktion enthält als Parameter die Daten, die das Skript zurückliefert; was mit ihnen geschehen soll, steht im Funktionsrumpf

Das PHP-Skript muss die per GET übertragenen Daten entgegennehmen und darauf basierend Inhalte zurück liefern

Asynchron Inhalte mit GET versenden

Zuerst wird überprüft, ob der GET-Parameter gesetzt ist und den Wert Nutzpflanzen hat. Ist dies der Fall, wird ein Array mit Nutzpflanzen definiert.

Im Beispiel wird direkt ein Array definiert. An dieser Stelle könnte natürlich auch eine Datenbankabfrage stattfinden, die eine Ergebnismenge zurückliefert

Dann wird eine Funktion `ausgeben()` aufgerufen, die für eine ordentliche Ausgabe sorgt.

Wenn das Skript pur aufgerufen wird, also beispielsweise mit `daten.php?wahl=Nutzpflanzen`, so liefert es - ohne sonstigen HTML-Code drum herum - eine Liste mit den angegebenen Nutzpflanzen.

Im Skript jedoch wird es per JavaScript/jQuery aufgerufen. JavaScript/jQuery wartet auf die Rückgabe des PHP-Skripts und baut den zurückgegebenen Inhalt in das Element mit `id="ausgabe"` ein.

Beachte aber, dass dieses Skript wirklich nur aus dem Im Listing gezeigten Code besteht. Es beinhaltet kein HTML-Grundgerüst. Das ist typisch für Daten, die per Ajax angefordert werden.

```php
<?php
// daten.php
if((isset($_GET["wahl"])) && ($_GET["wahl"] == "Nutzpflanze")) {
    $pflanzen = [
        "moschus Erdbeere",
        "Chester Thornless",
        "Annelises Rudolph",
        "Roter Weinbergpfirsich",
        "Pfälzer Fruchtfeige"
    ];
    echo ausgeben($pflanzen);
} else if((isset($_GET["wahl"])) && ($_GET["wahl"] == "Zierpflanzen")) {
    $pflanzen = [
        "Gelber Frauenshuh",
        "Knabenkraut",
        "Duftschneeball",
        "Stundentenblume",
        "Storchschnabel"
    ];
    echo ausgeben($pflanzen);
}
function ausgeben($was) {
    $str = "<ul>";
    foreach($was as $w) {
        $str .= "<li>$w</li>";
    }
    $str .= "</ul>";
    return $str;
}
```

Ajax: Formulardaten per POST versenden

Zuerst speichern wir einen Zugriff auf das Formular in der Variablen $form. Mit der `on()`-Methode reagieren wir auf das Submit-Ereignis. Hier wird das Ergebnis `submit` übergeben und als zweiten Parameter eine anonyme Funktion mit dem Parameter e. Dieser repräsentiert das Ereignis selbst. Wir könnten darüber weitere Informationen über das Ereignis erhalten. Im Beispiel hier der benötigt, um die Standardaktion mit `preventDefault()` zu unterbinden. Die Standardaktion bei einem `submit` ist ein Absenden des Formulars. Da wir den Versand jedoch selbst übernehmen wollen, müssen wir die Standardaktion außer Kraft setzen. Danach lesen wir die URL aus dem Formular aus. Für den Zugriff auf Attribute bietet jQuery die Methode `attr()`, der man als Attribut übergibt, das was man auslösen möchte. Danach kommt `$.post()`, über das sich ein POST-Request per Ajax durchführen lässt. Zuerst übergeben wir den Skriptnamen (in der Variablen `url` gespeichert), das per Ajax aufgerufen werden soll. Dann folgen die Daten, die verssendet werden sollen. Dafür kann man `serialize()` verwenden, das alle Formulardaten in serialisierter Form weiterreicht.

Die darauffolgende anonyme Funktion wird aufgerufen, wenn der Datenversand erfolgreich war, und enthält die Daten aus dem externen Script. Diese erscheinen jetzt im `.container`-Element und ersetzen dort vorher stehende Formular

```html
<div class="container">
    <form id="meinformular" action="send.php" method="post">
        <div class="textfeld">
            <label for="Vorname">Vorname</label>
            <input type="text" id="Vorname"  name="Vorname">
        </div>
        <div class="textfeld">
            <label for="Nachname">Nachname</label>
            <input type="text" id="Nachname"  name="Nachname">
        </div>
        <div>
            <input type="submit" value="Absenden">
        </div>
    </form>
</div>
```

```javascript
var $form = $('#meinformular');
$form.on('submit', function (e) {
    e.preventDefault();
    var url = $form.attr('action');
    $.post(
        url,
        $form.serialize(),
        function(daten) {
            $('.container').html('Vielen Dank: ' + daten);
        }
    )
});
```

AJAX: FORMULARDATEN PER POST VERSENDEN

CODERS.BAY

Im PHP-Skript werden alle erhaltenen Daten ausgegeben - sie erscheinen dann wieder auf der ursprünglichen Seite. Hier würde bei echten Beispielen eine Aktion wie ein E-Mai-Versand oder das Eintragen der Daten in ein Formular stattfinden - und die Meldung "Vielen Dank für ihre Anfrage" o. Ä. würde als Anzeigen genügen.
Wie du siehst, macht jQuery die Arbeit mit Ajax sehr einfach, man muss sich nicht um Details kümmern, sondern nur die notwenigsten Angaben hinschreiben - alles andere geschieht im Hintergrund

Wenn du nicht jQuery nutzen willst, aber trotzdem Unterstützung bei Ajax suchen, so solltest du dir einmal axio ansehen.

FORMULARDATEN

- Formulardaten kann man ganz einfach für PHP aufbereiten

```
let formData = $('#myForm').serialize();
```

Ende