

JavaScript-Objekte

AJAX

JSON

Interaktive Medien II
FS 20

Einführung

Die Themen JavaScript-Objekte, AJAX und JSON gehören zusammen.

Ziel

Daten vom Server laden und auf der Webseite anzeigen

Ziel

Daten vom Server laden und auf der Webseite anzeigen

JSON AJAX JS-Objekte

Ziel

Daten vom Server laden und auf der Webseite anzeigen

JSON	AJAX	JS-Objekte
------	------	------------

JavaScript-Objekte ?

Bündelung zusammengehörender Daten

Daten werden nach einfachen Regeln in JavaScript strukturiert zusammengefasst.

Mit JavaScript lässt sich der Inhalt von JavaScript-Objekten sehr leicht auf der Webseite anzeigen.

Beispiel

```
let person1 = {  
    alter: 42,  
    anrede: 'Herr',  
    name: {  
        vorname: 'Urs',  
        nachname: 'Thöny'  
    },  
    interessen: ['Film', 'Fahrrad fahren']  
};
```

Lernziele: JavaScript-Objekte

Regeln für JavaScript-Objekte kennen

Daten im korrekten JavaScript-Objekt-Syntax speichern können

Inhalte von JavaScript-Objekten in der Webseite darstellen können

Ziel

Daten vom Server laden und auf der Webseite anzeigen

JSON	AJAX	JS-Objekte
------	------	------------

Ziel

Daten vom Server laden und auf der Webseite anzeigen

JSON AJAX JS-Objekte

AJAX ?



AJAX ?

[...]

Die Technologie ermöglicht es, einzelne Teile einer Webseite bei Bedarf asynchron zu laden, so dass sie dynamisch wird. Der angezeigte Inhalt lässt sich gezielt so manipulieren, ohne die komplette Seite neu zu laden.

<https://www.dev-insider.de/was-ist-ajax-a-782233/>

AJAX ?

Mit JavaScript Daten vom Server laden.

Lernziele: AJAX

Mit AJAX Daten vom Server laden können
Kontrollieren, ob Ladenvorgang erfolgreich war

geladene Daten in der Webseite darstellen

Ziel

Daten vom Server laden und auf der Webseite anzeigen

JSON AJAX JS-Objekte

Ziel

Daten vom Server laden und auf der Webseite anzeigen

JSON AJAX JS-Objekte

JSON ?

Die Daten werden nach einfachen Regeln in einer Textdatei strukturiert zusammengefasst.

JSON ist ein textbasiertes Daten-Austauschformat.

Daten werden in einer bestimmten Text-Struktur von einem Sender (Server) zu einem Empfänger (Webseite) übertragen.

Beispiel

```
{  
  "anrede": "Herr",  
  "name": {  
    "vorname": "Urs",  
    "nachname": "Thöny"  
  },  
  "interessen": ["Film", "Fahrrad fahren"],  
  "alter": 42  
}
```

```
let person1 = {  
    alter: 42,  
    anrede: 'Herr',  
    name: {  
        vorname: 'Urs',  
        nachname: 'Thöny'  
    },  
    interessen: ['Film', 'Fahrrad fahren']  
};
```

JavaScript-Objekt

```
{  
    "anrede": "Herr",  
    "name": {  
        "vorname": "Urs",  
        "nachname": "Thöny"  
    },  
    "interessen": ["Film", "Fahrrad fahren"],  
    "alter": 42  
}
```

JSON

JSON = JavaScript Object Notation

Lernziele: JSON

JSON-Regeln kennen

Daten im korrekten JSON-Syntax speichern können

JSON mit AJAX vom Server laden können

geladenes JSON in JavaScript-Objekt umwandeln können

Inhalte von JavaScript-Objekten (aus JSON erstellt) in der Webseite darstellen können

JavaScript-Objekte

(JS-Objekte)

Eigenschaften von JS-Objekten

Ein Objekt funktioniert als eine Zuordnungsliste, die unter bestimmten Namen weitere Unterobjekte, auch Member genannt, speichert.

Diese Unterobjekte teilt man in Eigenschaften und Methoden.

Methoden sind ausführbare Funktionen, die dem Objekt zugeordnet sind.

Eigenschaften sind alle nicht ausführbaren Unterobjekte.

Sie können als mit dem Objekt verbundene Variablen erklärt werden.



Syntax

Ein Objekt wird immer durch geschweifte Klammern eingeschlossen.

Nicht alles, was in geschweiften Klammern steht ist ein Objekt.

```
let beispiel = {  
}
```

Syntax - Objekt

Jede Objekt-Eigenschaft beginnt mit einem Eigenschaftsname, gefolgt von einem Doppelpunkt und dem Eigenschaftswert.

Eigenschaftnamen werden **nicht** in Anführungszeichen geschrieben.

Bei Eigenschaftnamen wird Gross- und Kleinschreibung unterschieden.

Konvention: Eigenschaftnamen werden kleingeschrieben.

```
let beispiel = {  
    eineEigenschaft : 'Der Wert einer Eigenschaft'  
}
```



Syntax

Mehrere Eigenschaften werden durch Kommata getrennt.

Nach dem letzten Eigenschaftswert darf kein Komma stehen.

```
let beispiel = {  
    ganzzahl : 42,  
    fliesskommazahl : 13.37,  
    eigenschaftsname : 'Eigenschaftswert'  
}
```

The code example is annotated with three blue speech bubbles:

- A bubble above the first comma after "ganzzahl" contains the word "Komma".
- A bubble above the second comma after "fliesskommazahl" contains the word "Komma".
- A bubble next to the string "Eigenschaftswert" contains the text "kein Komma".

Syntax – Zusammenfassung

Ein Objekt wird immer durch geschweifte Klammern eingeschlossen.

Nicht alles, was in geschweiften Klammern steht ist ein Objekt.

Jede Objekt-Eigenschaft beginnt mit einem Eigenschaftsname, gefolgt von einem Doppelpunkt und dem Eigenschaftswert.

Die Eigenschaftnamen werden **nicht** in Anführungszeichen geschrieben.

Mehrere Eigenschaften werden durch Kommata getrennt.

Nach dem letzten Eigenschaftswert darf kein Komma stehen.

Syntax - Objekteigenschaft

Eine Eigenschaft ist wie eine Variable innerhalb eines Objekts. Ebenso wie Variablen kann sie jede Eigenschaft einen unterschiedlichen Datentypen enthalten:

- Zeichenkette in Anführungszeichen (einfach oder doppelt)
- Zahl ohne Anführungszeichen, Punkt als Dezimaltrenner
- Objekt in geschweiften Klammern
- Array in eckigen Klammern
- **undefined** kleingeschrieben, ohne Anführungszeichen
- **null** kleingeschrieben, ohne Anführungszeichen
- **bool** kleingeschrieben, ohne Anführungszeichen

gutes Beispiel

Geschweifte Klammern öffnen ...

```
let gutesBeispiel = {  
    ganzzahl : 42,  
    fliesskommazahl : 13.37,  
    zeichenkette : 'Hallo Welt',  
    unterobjekt : {  
        vorname : 'Urs',  
        nachname : 'Thöny'  
    },  
    einArray : ['foo', 'bar'],  
    undefinierbar : undefined,  
    nix : null,  
    entwederOder : true  
}
```

Zeichenketten können in einfachen oder doppelten Anführungszeichen stehen.

undefined ist korrekt, wird jedoch selten als Wert vergeben.

null ist korrekt, wird jedoch selten als Wert vergeben.

undefined, null und true/false müssen in Kleinbuchstaben geschrieben werden.

... und schliessen das JS-Objekt

SCHLECHTES Beispiel

Eigenschaftnamen werden in JS-Objekten (auch in Unterobjekten) immer ohne Anführungszeichen geschrieben!

```
let schlechtesBeispiel = {  
    ganzzahl : "42",  
    fliesskommazahl : 13,37,  
    zeichenkette : 'Hallo Welt'  
    unterobjekt : ({  
        "vorname" : "'Urs'",  
        "nachname" : Thöny  
    }),  
    einArray : {'foo', 'bar'},  
    undefinierbar : "undefined",  
    nix : NULL,  
    entwederOder : FALSE  
}
```

Das ist keine Zahl, sondern eine Zeichenkette.

Der Dezimaltrenner in Fliesskommawerten muss ein Punkt sein.

Hier fehlt das Komma.

Auch Unterobjekte stehen in geschweiften Klammern.

Hier fehlen die Anführungszeichen.

Diese Zeichenkette beinhaltet die einfachen Anführungszeichen.
Korrekt, aber unschön!

Arrays stehen immer in eckigen Klammern.

Das ist nicht undefined, sondern eine Zeichenkette.

null muss in Kleinbuchstaben geschrieben werden,
undefined übrigens auch.

true und false müssen auch in Kleinbuchstaben geschrieben werden.

Dot.Syntax – Zugriff auf Eigenschaften

Auf Eigenschaften und ihre Werte zuzugreifen ist sehr einfach. Wir verwenden den **Dot.Syntax**.

Mit dem Objektnamen rufen wir das komplette **Objekt** ab.

Um auf den **Wert einer Eigenschaft** zuzugreifen schreiben wir den Objektnamen und hängen getrennt durch einen Punkt den Eigenschaftnamen an.

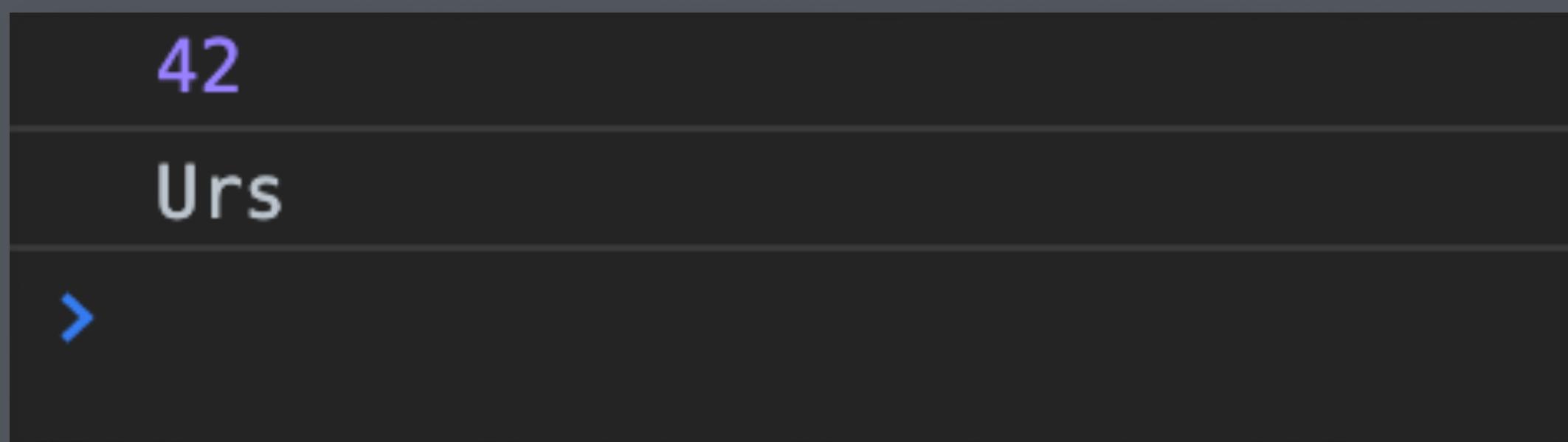
```
let wert1 = objekt.eigenschaft;
```

Um auf den **Wert einer Untereigenschaft** zuzugreifen schreiben wir den Objektnamen und hängen getrennt durch einen Punkt den Eigenschaftnamen an, woran wir wiederum durch einen Punkt getrennt den Untereigenschaftnamen anhängen.

```
let wert2 = objekt.eigenschaft.untereigenschaft;
```

Beispiel

```
let person1 = {  
    alter: 42,  
    anrede: 'Herr',  
    name: {  
        vorname: 'Urs',  
        nachname: 'Thöny'  
    },  
    interessen: ['Film', 'Fahrrad fahren'],  
};  
  
console.log(person1.alter);  
console.log(person1.name.vorname);
```



```
42  
Urs  
>
```

A dark terminal window showing the output of the JavaScript code. The first line '42' is in blue, indicating it's a number. The second line 'Urs' is in green, indicating it's a string. A blue '>' symbol at the bottom left indicates the prompt.

JS-Objekte – Übung 1

```
/* JS-Objekte | Übung 1 | 01_objekt_grundlagen
* 1. Erstellen Sie für den Kanton Bern ein JavaScript-Objekt mit dem Namen "kanton1"
* Das Objekt soll folgende Eigenschaften haben:
*   name,
*   amtssprache,
*   hauptort,
*   flaeche und für die Eigenschaft
*   bevoelkerung die Untereigenschaften
*   einwohner,
*   quote.
* Die Informationen entnehmen Sie der Seite:
*   https://de.wikipedia.org/wiki/Kanton\_Bern
* 2. Rufen Sie in der Konsole des Browsers die einzelnen Objektwerte auf.
*/
```

```
> kanton1
< ▶ {name: "Bern", amtssprache: Array(2), hauptort: "Bern", flaeche: 5959.24, bevoelkerung: {...}}
> kanton1.flaeche
< 5959.24
> kanton1.bevoelkerung.einwohner
< 1034977
>
```

Eigenschaftswerte in HTML anzeigen

Jetzt, wo wir auf Eigenschaftswerte eines Objektes zugreifen können, wollen wir sie auch in der Webseite anzeigen.

Dazu müssen wir die Werte in ein HTML-Element schreiben.

Vorgehen:

HTML-Element mit `document.querySelector()` identifizieren und ...

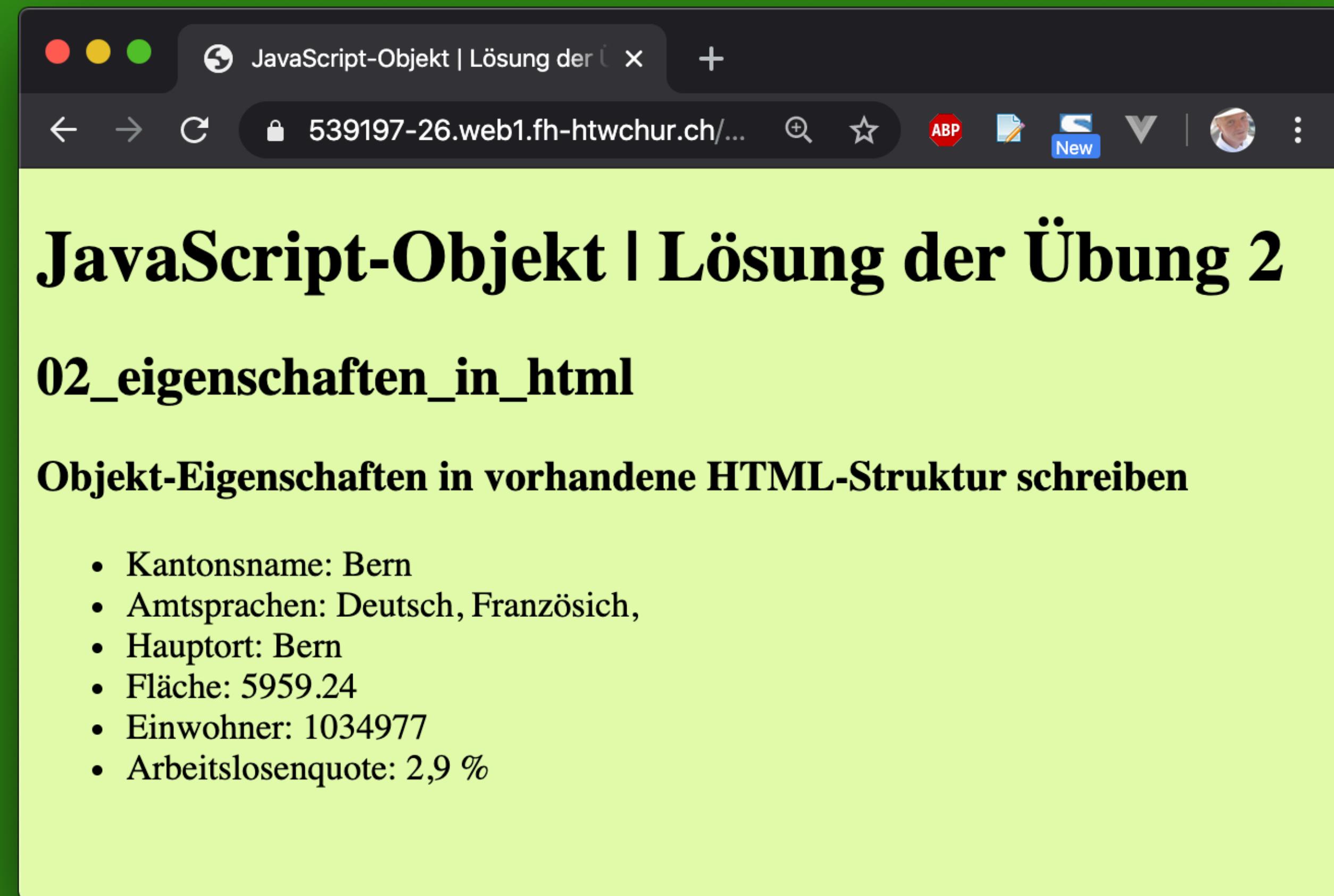
```
document.querySelector('#person_anrede')
```

... Eigenschaftswert mit `.textContent` in das Element schreiben.

```
document.querySelector('#person_anrede').textContent = person1.anrede;
```

JS-Objekte – Übung 2

```
/* JS-Objekte | Übung 2 | 02_eigenschaften_in_html
* Übertragen Sie die Werte aus dem JS-Objekt kanton1 in die li-Elemente der HTML-Liste.
*/
```



Liste aus Eigenschaftswerten erstellen

In Übung 1 haben wir die Eigenschaftswerte in eine vorbereitete HTML-Liste geschrieben.

Jetzt werden wir die Angaben zur Amtssprache in als eine eigene Liste innerhalb der vorhandenen Liste anzeigen.

Verschachtelte Listen sind nichts anderes, als eine eigene Liste innerhalb eines vorhandenen Listenpunktes.

siehe Wiki-selfHTML: [HTML/Textstrukturierung/ul](#)

- Kantonsname: Bern
- Amtssprachen:
 - Deutsch
 - Französisch
- Hauptort: Bern
- Fläche: 5959.24
- Einwohner: 1034977
- Arbeitslosenquote: 2,9 %

Liste aus Eigenschaftswerten erstellen

Vorgehen:

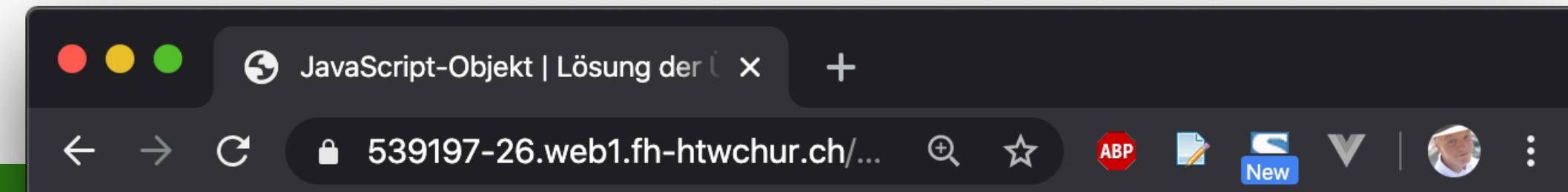
Mit `document.createElement('ul')` in JavaScript eine neue Liste erzeugen und in einer Variablen speichern.

Das Array mit den Amtssprachen durchlaufen und für jedes Element einen neuen Listenpunkt erstellen, den aktuellen Arraywert in den Listenpunkt schreiben und mit `.appendChild()` den Listenpunkt in die gerade erstellte Liste schreiben.

Zuletzt die neue Liste mit `.appendChild()` in den richtigen Listenpunkt schreiben.

JS-Objekte – Übung 3

```
/* JS-Objekte | Übung 3 | 03_liste_aus_eigenschaften_erstellen
* 1. Erstellen Sie für die Werte des Array kanton1.amtssprache mit Hilfe von JavaScript
eine ungeordnete Liste (ul-Element) und ...
* 2. fügen in diese Liste für jedes Array-Element ein HTML-Listen-Elemen (li-Element) mit
dem entsprechenden Wert ein.
* 3. Fügen Sie die vollständige HTML-Liste (ul-Element) in das HTML-Element mit der id
"kanton_hauptort" ein.
*/
```



The screenshot shows a web browser window titled "JavaScript-Objekt | Lösung der Übung 3". The URL in the address bar is "539197-26.web1.fh-htwchur.ch/...". The page content is as follows:

JavaScript-Objekt | Lösung der Übung 3

03_liste_aus_eigenschaften_erstellen

ungeordnete Liste aus dynamisch in JavaScript aus Untereigenschaften des Objekts erstellen und in HTML einfügen

- Kantonsname: Bern
- Amtsprachen:
 - Deutsch
 - Französisch
- Hauptort: Bern
- Fläche: 5959.24
- Einwohner: 1034977
- Arbeitslosenquote: 2,9 %

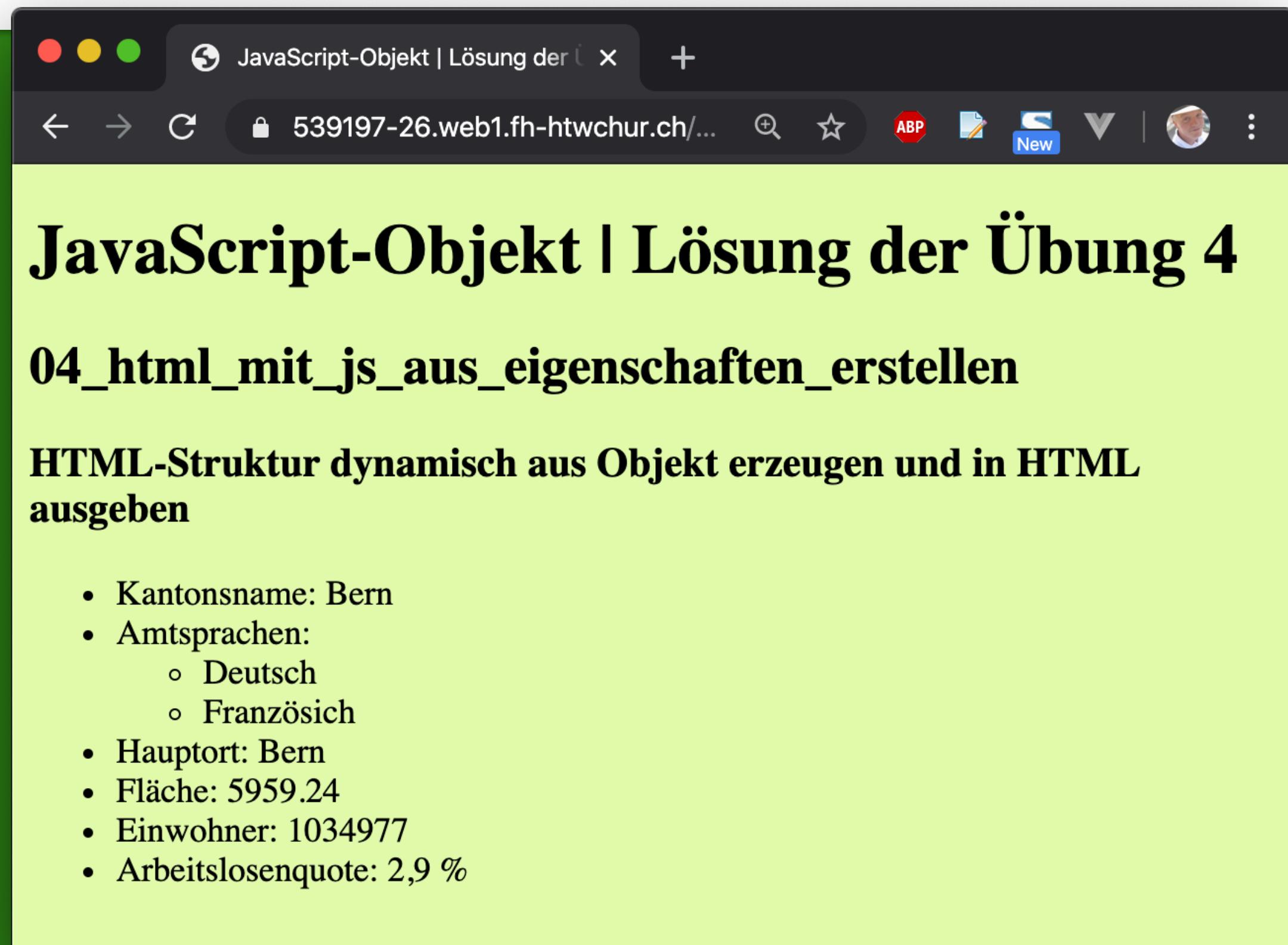
HTML mit JavaScript aus Eigenschaften erstellen

Wir können auch alle Angaben aus dem Objekt durch JavaScript in HTML darstellen.

So können wir flexibel auf unterschiedliche Angaben innerhalb verschiedener Objekte reagieren.

JS-Objekte – Übung 4

```
/* JS-Objekte | Übung 4 | 04_html_mit_js_aus_eigenschaften_erstellen
 * 1. Erstellen Sie die HTML-Struktur mit Hilfe von JavaScript ...
 * 2. und schreiben Sie die Eigenschaftswerte aus "kanton1" in die entsprechenden Elemente
 * 3. Fügen Sie zuletzt die ganze, in JS erstellte HTML-Struktur als ChildNode in das HTML-
Element mit der id "container" ein.
 */
```

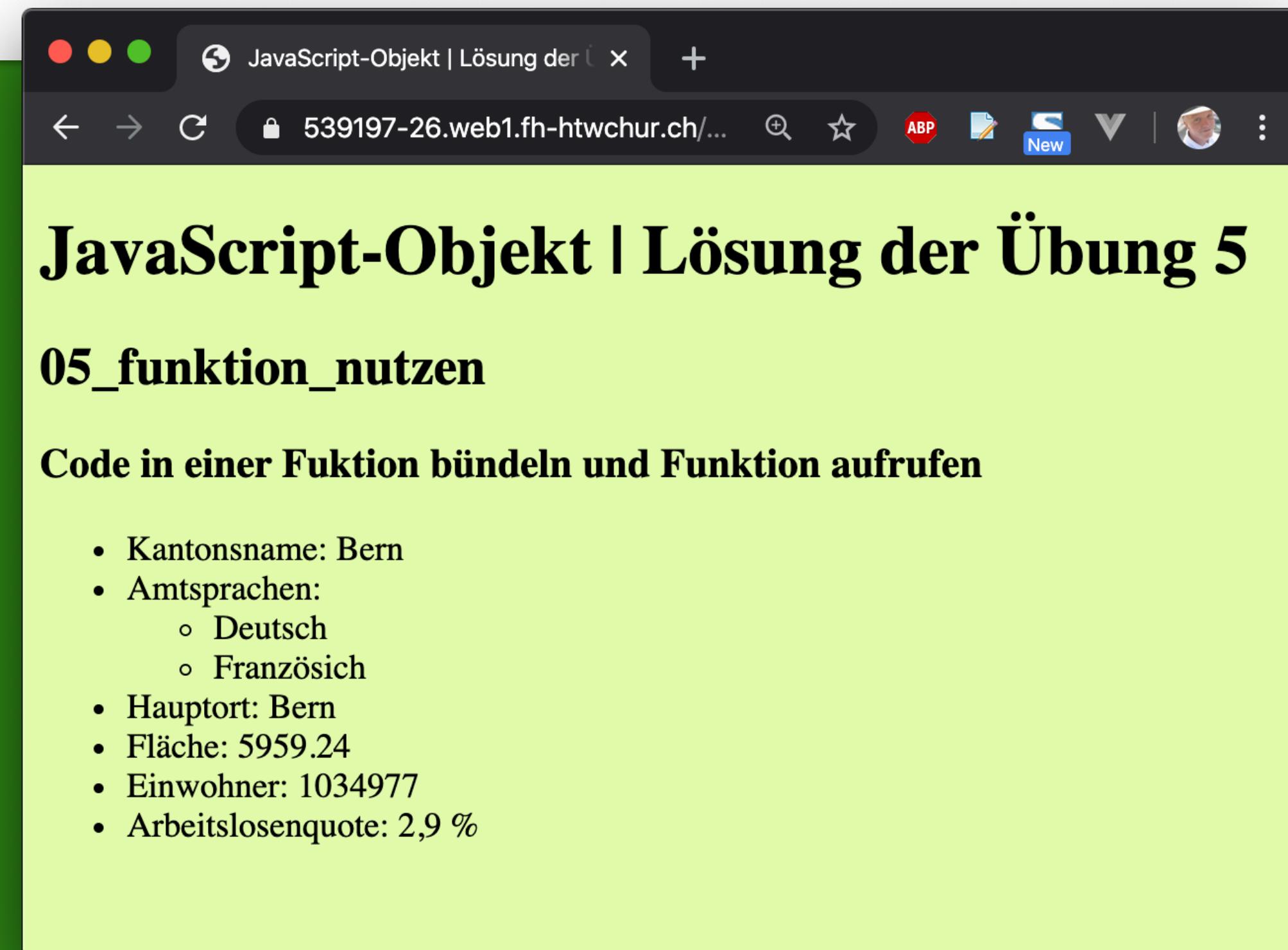


eine Funktion zur Darstellung nutzen

Noch flexibler wird es, wenn wir alle Befehle zur Erzeugung und Darstellung des Arrays in einer Funktion bündeln.
(Vorbereitung für Übung 6)

JS-Objekte – Übung 5

```
/* JS-Objekte | Übung 5 | 05_funktionen_nutzen
* 1. Packen Sie den kompletten Aufbau in eine Funktion mit dem Namen "kanton_anzeigen()"
mit dem Funktionsparameter "kanton_param").
* Dem Funktionsparameter "kanton_param" übergeben Sie später beim Funktionsaufruf das
Objekt "kanton1".
* 2. Ersetzen Sie innerhalb der Funktion "kanton1" durch "kanton_param".
* 3. Schreiben Sie "anzeige" als Kind-Element in das HTML-Element mit der id="container".
* 4. Rufen Sie die Funktion auf mit dem JS-Objekt "kanton1" als Parameter auf.
*/
```



mehrere Objekte mit einer Funktion anzeigen

Mit Hilfe der Funktion können wir weitere Objekte ganz einfach anzeigen.

JS-Objekte – Übung 6

```
/* JS-Objekte | Lösung der Übung 6 | 06_mehrere_objekte_mit_funktionen_anzeigen
* 1. Erstellen Sie für den Kanton Graubünden ein JavaScript-Objekt mit dem Namen "kanton2"
* Das Objekt soll die selben Eigenschaften haben wie "kanton1"
* Die Informationen entnehmen Sie der Seite:
*     https://de.wikipedia.org/wiki/Kanton\_Graub%C3%BCnden
* 2. Rufen Sie zusätzlich die Funktion auf mit dem JS-Objekt "kanton2" als Parameter auf.
*/
```



JS-Objekte – Aufgabe 1

```
/* JS-Objekte | Aufgabe 1 | 01_bus_objekt_erstellen
/* Bilden Sie den Verlauf der Churer Buslineien 6 und 9 als JS-Objekt ab.
/* Jedes Bus-Objekt muss die Eigenschaften liniennummer, kurzbeschreibung und haltestellen
beinhalten.
/* Den Linienverlauf der Buslinien entnehmen Sie den Seiten
/*      https://churbus.ch/linie/linie-6 und
/*      https://churbus.ch/linie/linie-9
/* *****
```

Keine Anzeige im Browser

JS-Objekte – Aufgabe 2

```
/* JS-Objekte | Aufgabe 2 | 02_bus_objekt_anzeigen
/* Erstellen Sie eine Funktion "busobjekt_in_html_anzeigen()" mit dem Funktionsparameter
"bus_parameter",
/* Die Funktion gibt ein alle Angaben zur Buslinie als HTML zurück.
/* Wenden Sie die Funktion mit den JS-Objekten an und geben Sie das Ergebnis als Child-
Elemente des HTML-Elements mit der id "container" im Browser aus.
/* ***** */
```



Feedback

anonymes Feedback

AJAX

AJAX war nie neu

AJAX war nie eine neue Erfindung.

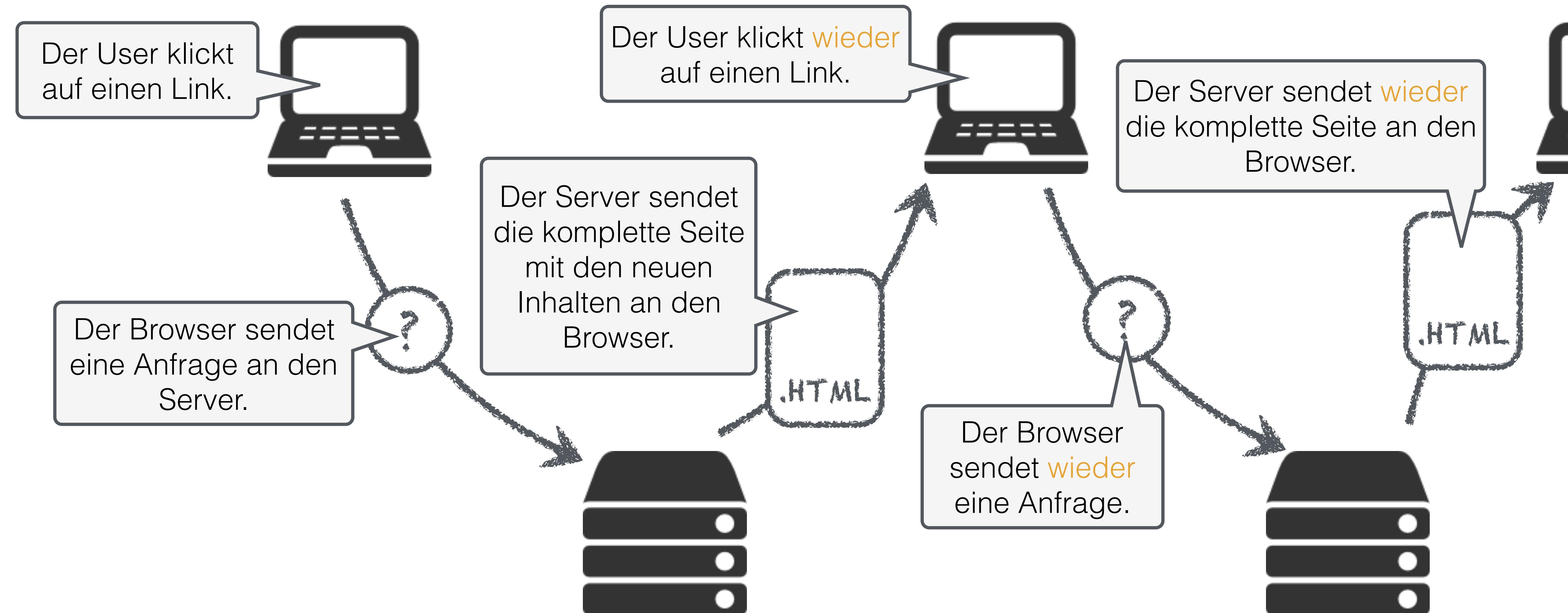
Die Techniken von AJAX existieren schon seit ca. 1998.

Zusammenfassung der Techniken erst ca. 2004.

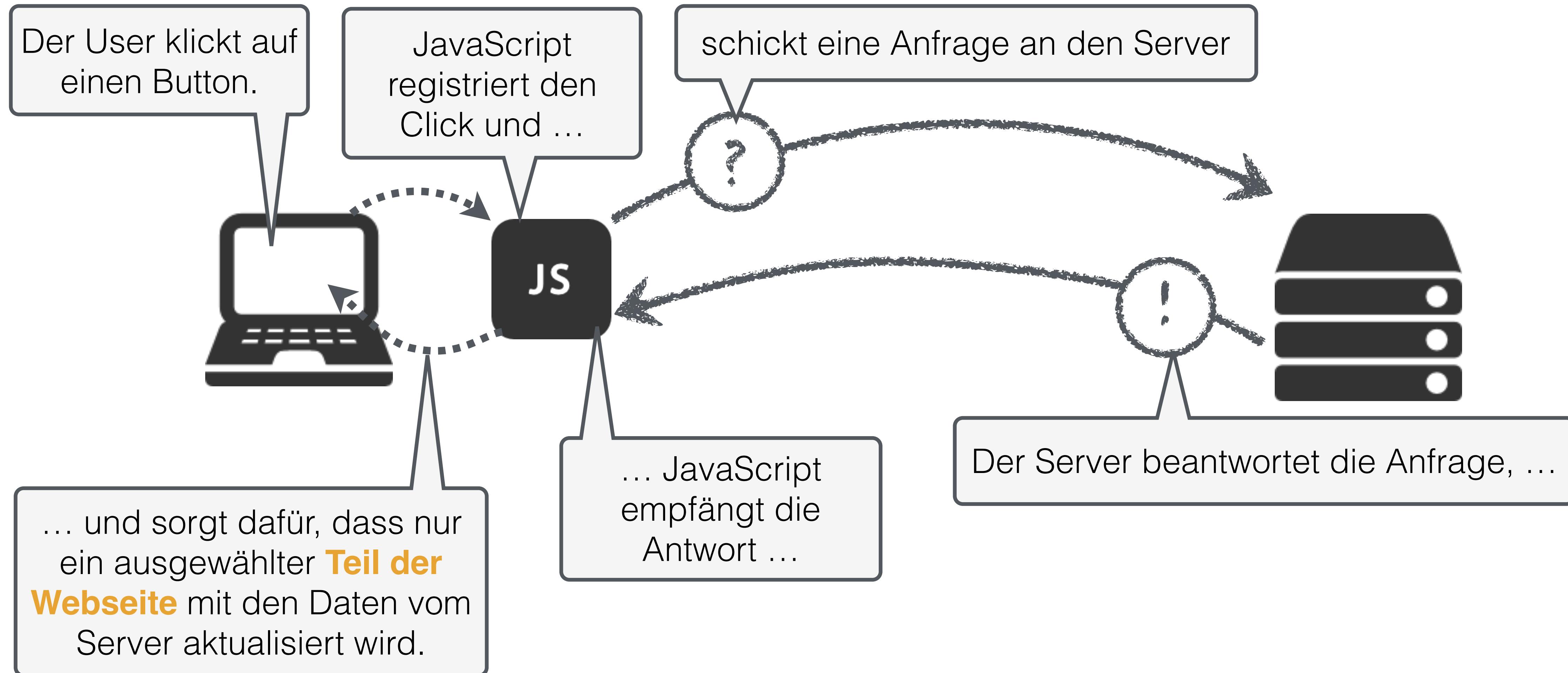
Was bedeutet AJAX

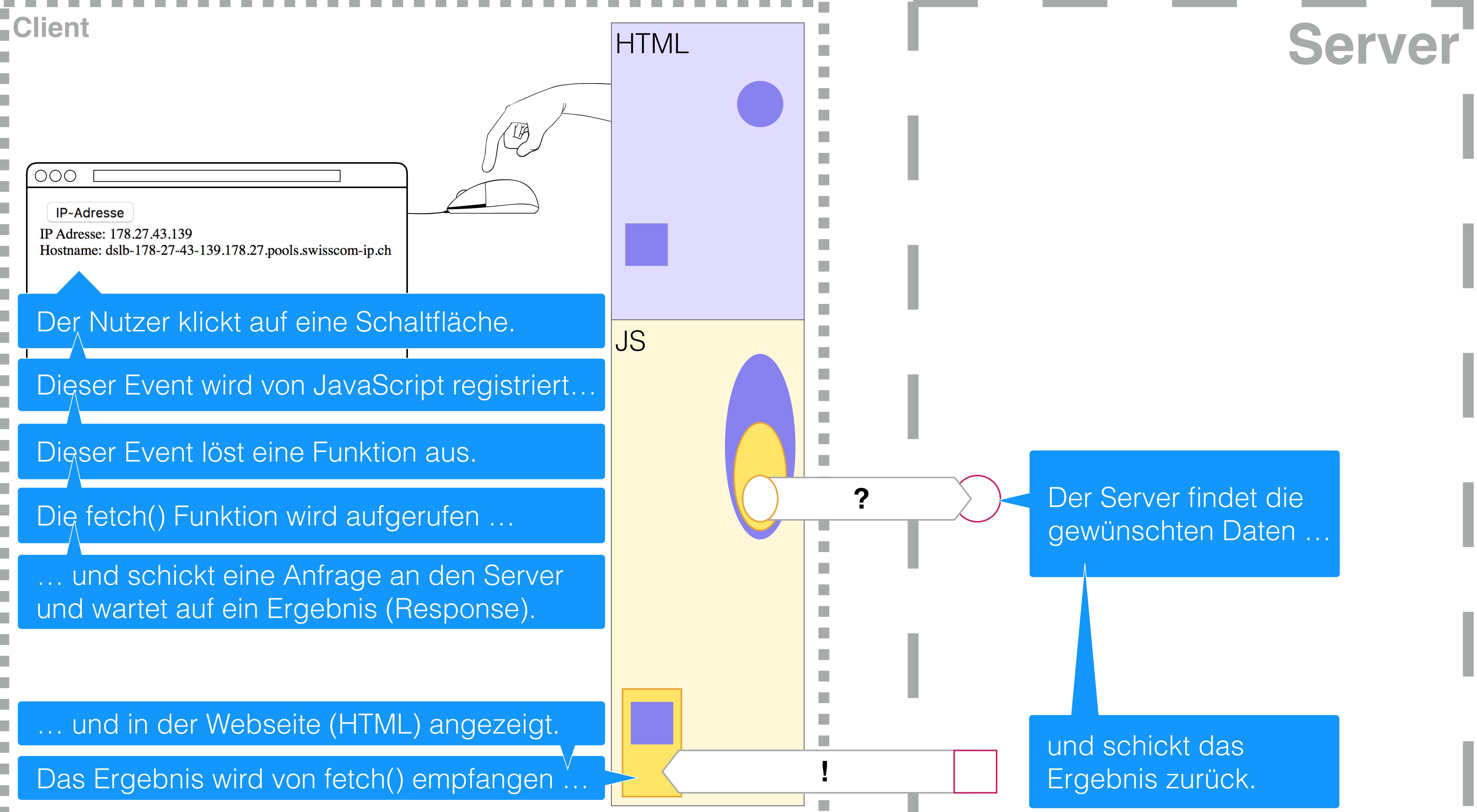
Aynchronous
JavaScript
And
XML

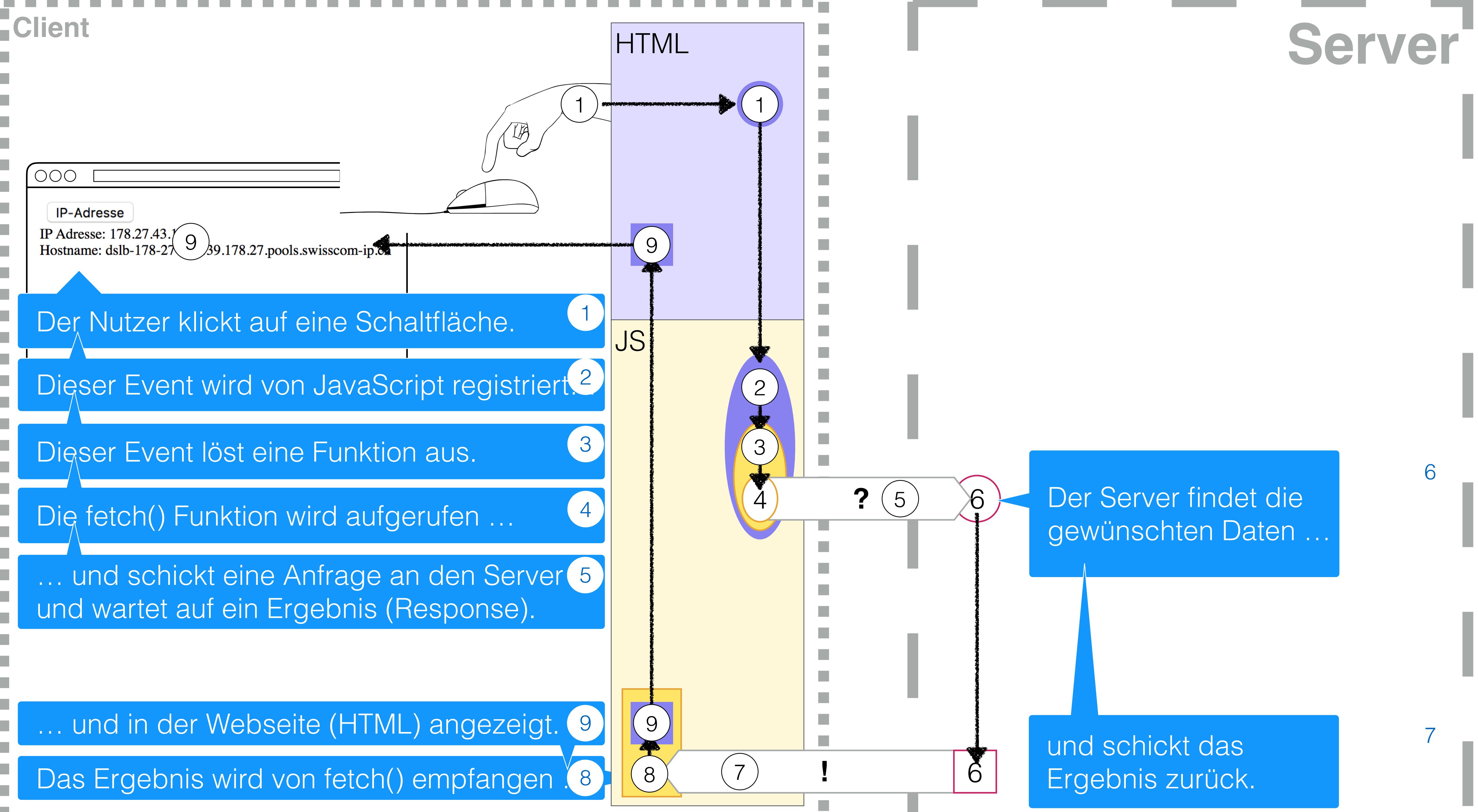
Serverkommunikation bisher



Serverkommunikation mit AJAX







fetch() Funktionsweise

Mit der `fetch()`-Funktion können wir mit JavaScript Dateien von einem Server laden.

`fetch()` ist ein relativ neuer Befehl, und er macht uns das Leben sehr viel einfacher.

Die `fetch()`-Funktion arbeitet mit sog. Promises (Versprechen).

Als Parameter benötigt `fetch()` lediglich die URL der zu ladenden Datei.

Promises

Die `fetch()`-Funktion arbeitet mit sog. Promises (Versprechen).

Promises sind eine erweiterbare Kette von Funktionen.

Promises am Beispiel von fetch()

fetch() benötigt als Parameter die URL der zu ladenden Datei.

Wenn fetch() mit der URL aufgerufen wurde, verspricht fetch() eine Antwort (response) zu liefern.

Wir müssen angeben, welches Datenformat zu erwarten ist.

Wenn die Antwort vom Server eintrifft, verspricht fetch() Daten zu liefern.
Mit diesen Daten können wir jetzt Arbeiten.

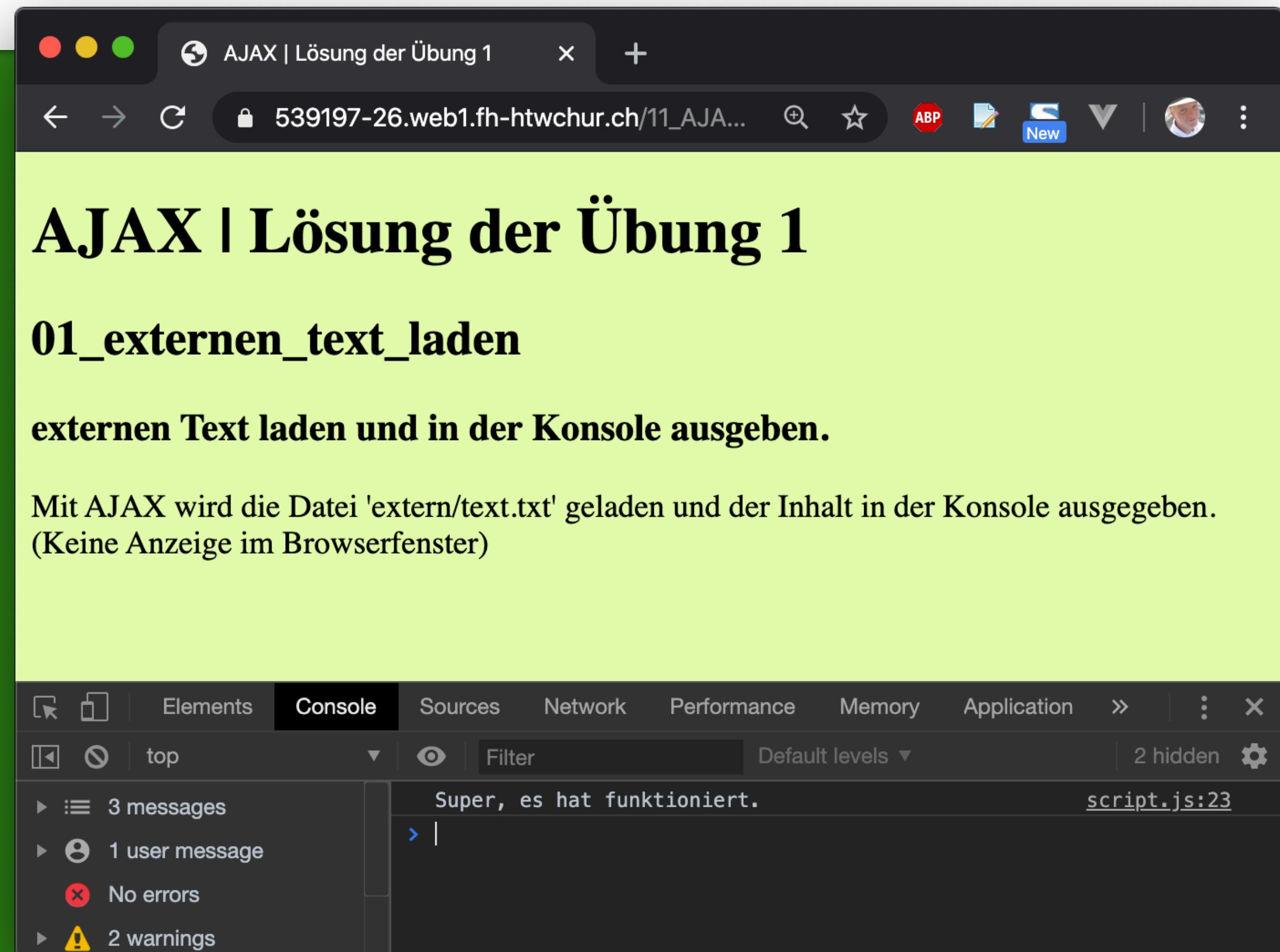
Sollte etwas nicht funktionieren,
können wir den Fehler einfangen (catchen).

fetch() Funktionsweise

```
// mit fetch() das Laden einer externen Datei starten.  
// Als Parameter benötigt fetch() den Pfad zur externen Datei.  
fetch('extern/text.txt')  
  // Der fetch() Aufruf erwartet eine Antwort (response)  
  .then((response) => {  
    // Definieren, welches Format die Antwort hat (wichtig für den nächsten Teil)  
    // hier text  
    return response.text();  
  })  
  // Wenn die Antwort eintrifft ...  
  .then((data) => {  
    // ... wird sie weiterverarbeitet (hier: Ausgabe in die Konsole)  
    console.log(data);  
  })  
  // Nur wenn etwas nicht funktioniert hat ...  
  .catch(function(error) {  
    // ... wird eine Fehlermeldung ausgegeben.  
    console.log('Error: ' + error.message);  
  });
```

AJAX – Übung 1

```
/* AJAX | Übung 1 | 01_externen_text_laden
/* 1. Laden Sie den Inhalt aus der Datei 'extern/text.txt' mit Hilfe der fetch()-API
/* 2. Geben Sie im ersten Promise an, welches Datenformat als Antwort erwartet wird.
/* 3. Geben Sie den geladenen Text in der Konsole aus.
/* Kontrollieren Sie das Ergebnis in der Konsole ihres Browsers.
/* *****
```

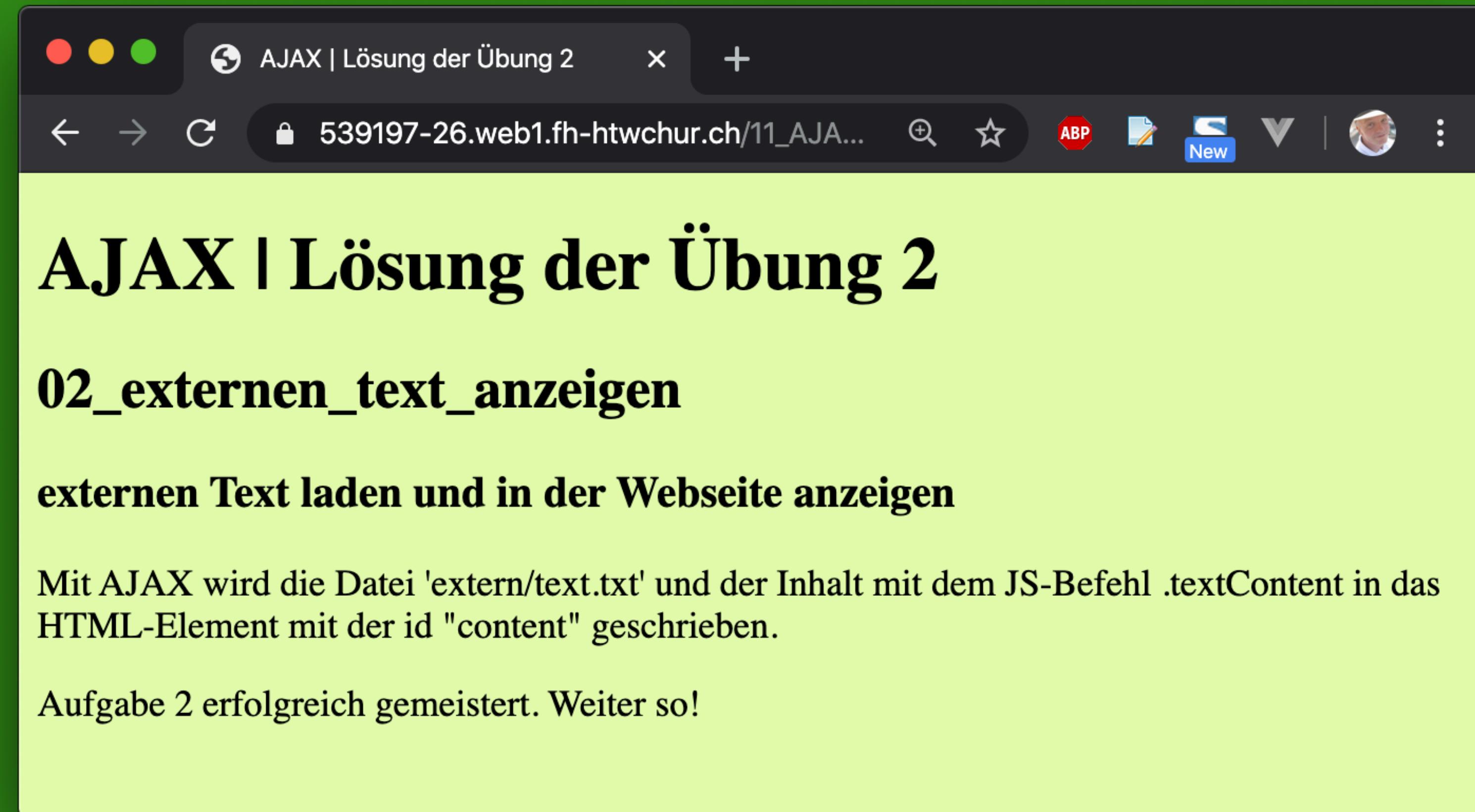


externen Text anzeigen

Den empfangenen Text können wir ganz einfach im Browserfenster ausgeben. Wir müssen ihn lediglich als Text in ein vorhandenes HTML-Element schreiben.

AJAX – Übung 2

```
/* AJAX | Übung 2 | 02_externen_text_anzeigen
/* 1. Geben Sie den geladenen Inhalt als textContent in dem HTML-Element mit der ID
'content' aus.
/* *****
```



externes HTML anzeigen

Wenn in der geladenen Datei HTML-Code steht, können wir diesen in die Webseite einbauen. Wir müssen dazu den geladenen HTML-Code mit `.innerHTML()` in ein vorhandenes Element laden.

AJAX – Übung 3

```
/* AJAX | Übung 3 | 03_externes_html_anzeigen
/* 1. Laden Sie den Inhalt aus der Datei 'extern/html_kanton1.txt' mit Hilfe der fetch()-API
/* 2. Geben Sie den geladenen Inhalt als innerHTML in dem HTML-Element mit der ID 'content'
aus.
/* *****
```



Inhalte interaktiv austauschen

Solch kleine Datenmengen beim laden der Seite mit `fetch()` abzurufen erscheint zunächst nicht sinnvoll.

Besser ist es, die Inhalte erst auf Anforderungen des Users zu laden.

Dazu müssen wir auf Ereignisse, die der User macht reagieren.

Es bietet sich an Buttons zu erstellen, die bei `'click'` den Ladeprozess auslösen.

Die Herausforderung besteht darin die Angabe über die zu ladende Datei, z. B. den Dateinamen, innerhalb des HTML-Codes des jeweiligen Buttons mitzugeben.

Hier hilft uns das data-*Attribut.

data-*

Mit **data-***-Attributen (Custom Data Attributes) haben Sie die Möglichkeit, Elementen eigene Attribute mitzugeben, die dann per Script ausgewertet oder mit CSS genutzt werden können.

In diesen Attributen können Sie Informationen, die nicht visuell präsentiert werden sollen, zur Verfügung stellen.

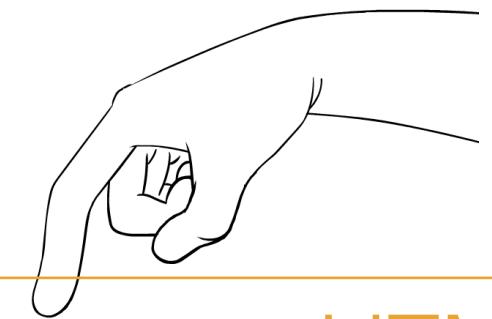
this in EventHandlern

Funktionen, die einen Event wie z. B. einen Mausklick verarbeiten, nennt man EventHandler.

Jeder Event hat einen sog. Auslöser, wie z. B. den Button, auf den geklickt wurde.

Innerhalb eines EventHandlers können wir mit dem Schlüsselwort **this** auf die Eigenschaften des Auslösers zugreifen.

data-* und this kombinieren



```
<div class="buttons">  
  <button class="btn" data-dateiname="html_kanton1">Bern</button><br>  
  <button class="btn" data-dateiname="html_kanton2">Graubünden</button><br>  
</div>
```

Bei Klick auf diesen Button ...

// Eine Liste (Array) mit allen Elementen mit der class "btn" wird in der Variablen "buttons" gespeichert.

```
let buttons = document.querySelectorAll('.btn');
```

... wird der EventHandler ausgelöst ...

// Die Liste "buttons" wird in einer Schleife durchlaufen.

```
for(let i = 0; i < buttons.length; i++){
```

// Jedem Listen-Element (jedem Button) wird ein Click-EventListener zugeordnet.

```
  buttons[i].addEventListener("click", function(){
```

// In der Variablen aktuellerDateiname speichern wir mit this.getAttribute("data-dateiname") ...

// ... den Inhalt des Attributs data-dateiname aus dem Button, der gerade geklickt wurde ...

// ... und dadurch den EventListener ausgelöst hat. (Dateiname ohne Extension)

```
    let aktuellerDateiname = this.getAttribute("data-dateiname");
```

... und in der Variablen aktuellerDateiname der Wert *html_kanton2* gespeichert

```
// Beim Click auf den Button wird die Funktion zeige_externen_inhalt() Aufgerufen.
```

// Als Parameter wird der Inhalt des data-dateiname-Attributs, des jeweiligen Buttons mitgegeben.

```
  zeige_externen_inhalt(aktuellerDateiname);
```

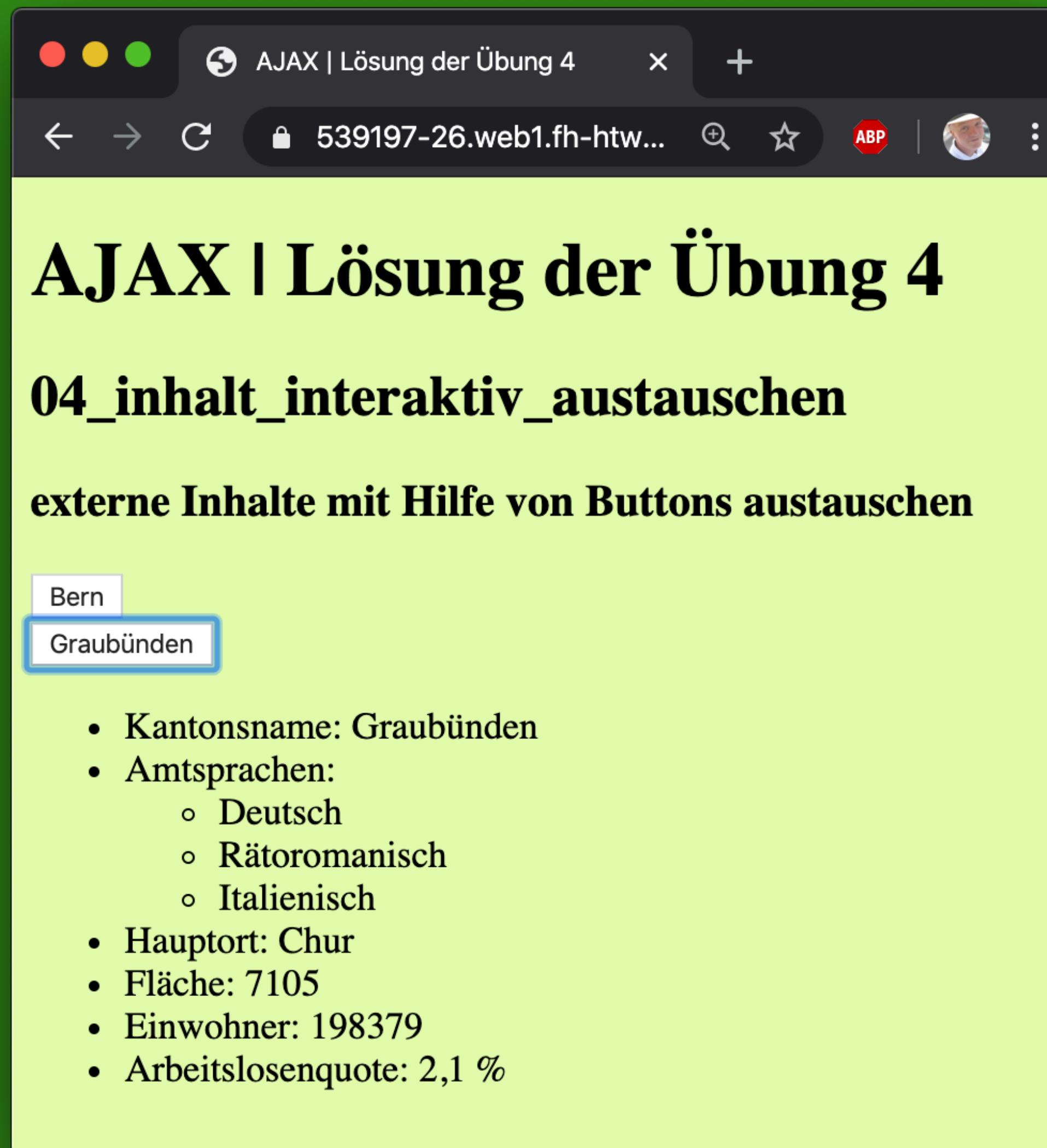
```
}
```

AJAX – Übung 4

```
/* AJAX | Übung 4 | 04_inhalt_interaktiv_austauschen
/* 1. Siehe: zugehörige HTML-Datei. (../index.html)
/* 2. Packen Sie den fetch()-Aufruf in eine Funktion mit dem Namen zeige_externen_inhalt()
/* 3. Als Parameter (dateiname) erhält die Funktion den Dateinamen der externen Datei.
(Dateiname ohne Extension)
/* 4. Setzen Sie den Pfad der zu ladenden Datei aus
/*      dem Ordnernamen mit folgendem Schrägstrich,
/*      dem Parameter "dateiname",
/*      der Extension mit vorstehendem Punkt zusammen,
/*      und speichern Sie die so entstandene Zeichenkette in der Variablen "url".
/* 5. Rufen Sie fetch() mit der Variablen url als Parameter auf
/* 6. Geben Sie den geladenen Inhalt als innerHTML in dem HTML-Element mit der ID 'anzeige'
aus.

/* 7. Speichern Sie mit Hilfe von .querySelectorAll() alle Buttons mit der class="btn" in
einer Variablen als Array.
/* 8. Durchlaufen Sie das Array mit einer for-Schleife ...
/* 9. ... und geben Sie mit Hilfe von .addEventListener() jedem Array-Element (Button) für
den EventListener "click" eine Funktion mit, ...
/* 10. ... die den Wert des Attributs data-dateiname des geklickten Buttons in einer Variable
speichert und ...
/* 11. ... unsere Funktion zeige_externen_inhalt() aufruft. Als Parameter erhält die Funktion
den Wert der soeben erstellten Variablen.
/* *****
```

AJAX – Übung 4



A screenshot of a web browser window titled "AJAX | Lösung der Übung 4". The URL is "539197-26.web1.fh-htw...". The page content is identical to the one on the right, but the "Graubünden" button is highlighted with a blue border.

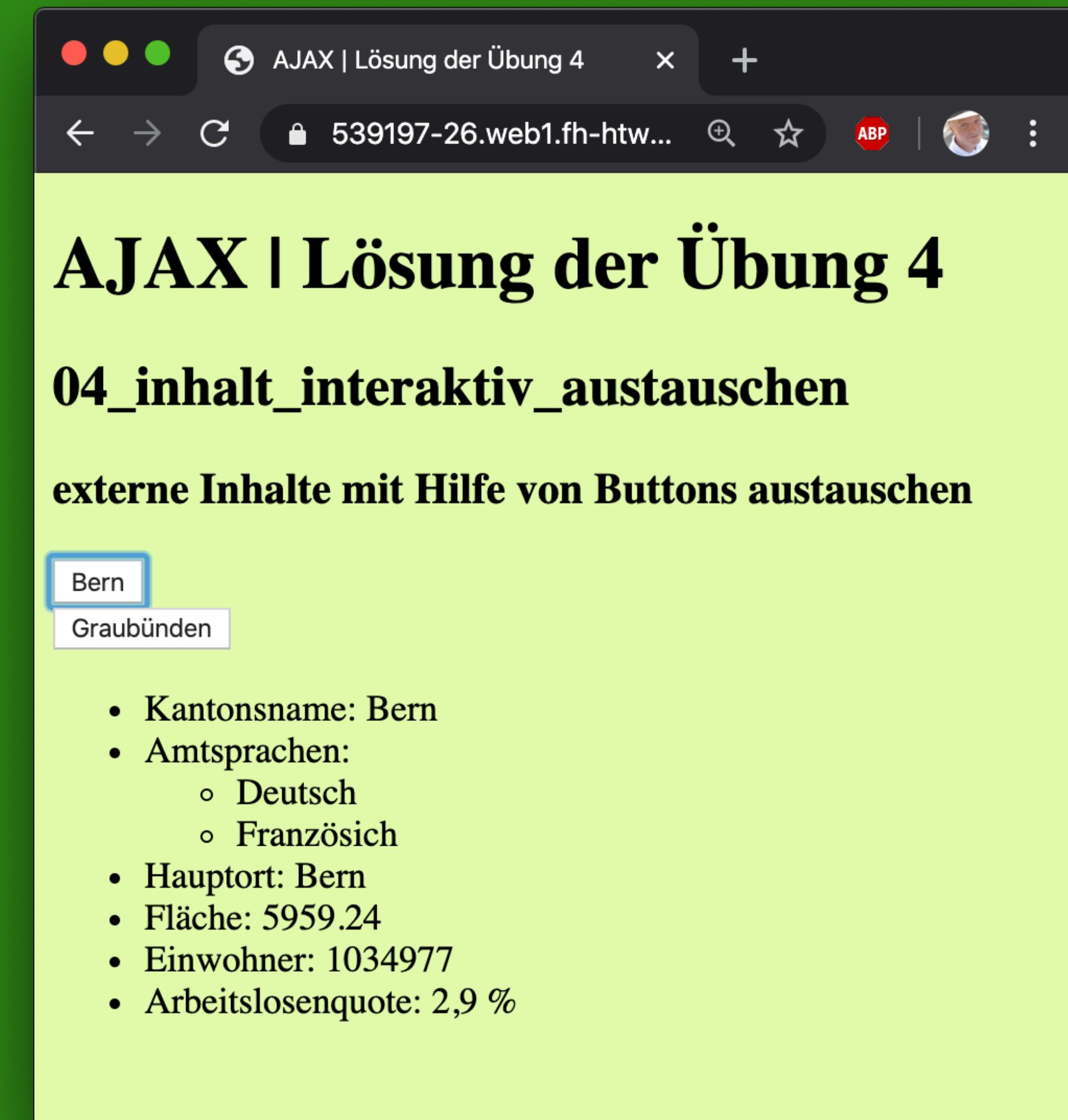
AJAX | Lösung der Übung 4

04_inhalt_interaktiv_austauschen

externe Inhalte mit Hilfe von Buttons austauschen

Bern
Graubünden

- Kantonsname: Graubünden
- Amtssprachen:
 - Deutsch
 - Rätoromanisch
 - Italienisch
- Hauptort: Chur
- Fläche: 7105
- Einwohner: 198379
- Arbeitslosenquote: 2,1 %



A screenshot of a web browser window titled "AJAX | Lösung der Übung 4". The URL is "539197-26.web1.fh-htw...". The page content is identical to the one on the left, but the "Bern" button is highlighted with a blue border.

AJAX | Lösung der Übung 4

04_inhalt_interaktiv_austauschen

externe Inhalte mit Hilfe von Buttons austauschen

Bern
Graubünden

- Kantonsname: Bern
- Amtssprachen:
 - Deutsch
 - Französich
- Hauptort: Bern
- Fläche: 5959.24
- Einwohner: 1034977
- Arbeitslosenquote: 2,9 %

AJAX– Aufgabe 1

```
/* AJAX | Aufgabe 1 | 01_bus_austauschen
/* Erstellen Sie in der zugehörigen HTML–Datei für jede Churer Buslinei einen Button.
/* Funktionalisieren Sie die Buttons der zugehörigen Datei so,
/* Dass beim Klick auf einen Button die passende Text–Datei geladen
/* und der Inhalt als HTML dargestellt wird.
/* Die txt–Dateien befinden sich im Verzeichnis "extern".
/* ***** */
```

The image displays a row of nine screenshots of a web application interface, each showing a different bus route or location. The interface has a green header bar with the title 'Übung: Inhalte mit Ajax austauschen'. Below the header, there is a list of stops for each route. Each screenshot is labeled with a number from 1 to 9.

- Screenshot 1:** Lachen – Bahnhofplatz – Plankis – Domat/Ems – Rhäzüns. Stop list: 1 Lachen – Bahnhofplatz – Plankis – Domat/Ems – Rhäzüns, 2 Obere Au – Bahnhofplatz – Fürstewald, 2 Obere Au – Bahnhofplatz – Kleinwaldlegg, 3 Austrasse – Bahnhofplatz – Stelleweg – Haldenstein, 3 Austrasse – Bahnhofplatz – Trimmis – Untervaz, 4 Austrasse – Bahnhofplatz – Seniorencentrum Cadonau, 6 Bahnhofplatz – City West, 9 Bahnhofplatz – Meiersboden.
- Screenshot 2:** Lachen – Bahnhofplatz – Plankis – Felsberg. Stop list: 1 Lachen – Bahnhofplatz – Plankis – Felsberg, 2 Obere Au – Bahnhofplatz – Fürstewald, 2 Obere Au – Bahnhofplatz – Kleinwaldlegg, 3 Austrasse – Bahnhofplatz – Stelleweg – Haldenstein, 3 Austrasse – Bahnhofplatz – Trimmis – Untervaz, 4 Austrasse – Bahnhofplatz – Seniorencentrum Cadonau, 6 Bahnhofplatz – City West, 9 Bahnhofplatz – Meiersboden.
- Screenshot 3:** Obere Au – Bahnhofplatz – Fürstewald. Stop list: 1 Obere Au P&R, 2 Obere Au, 3 Austrasse – Bahnhofplatz – Stelleweg – Haldenstein, 3 Austrasse – Bahnhofplatz – Trimmis – Untervaz, 4 Austrasse – Bahnhofplatz – Seniorencentrum Cadonau, 6 Bahnhofplatz – City West, 9 Bahnhofplatz – Meiersboden.
- Screenshot 4:** Obere Au – Bahnhofplatz – Kleinwaldlegg. Stop list: 1 Obere Au P&R, 2 Obere Au, 3 Austrasse – Bahnhofplatz – Stelleweg – Haldenstein, 3 Austrasse – Bahnhofplatz – Trimmis – Untervaz, 4 Austrasse – Bahnhofplatz – Seniorencentrum Cadonau, 6 Bahnhofplatz – City West, 9 Bahnhofplatz – Meiersboden.
- Screenshot 5:** Austrasse – Bahnhofplatz – Stelleweg – Haldenstein. Stop list: 1 Austrasse, 2 Obere Au – Bahnhofplatz – Kleinwaldlegg, 2 Obere Au – Bahnhofplatz – Fürstewald, 2 Obere Au – Bahnhofplatz – Stelleweg – Haldenstein, 3 Austrasse – Bahnhofplatz – Trimmis – Untervaz, 4 Austrasse – Bahnhofplatz – Seniorencentrum Cadonau, 6 Bahnhofplatz – City West, 9 Bahnhofplatz – Meiersboden.
- Screenshot 6:** Austrasse – Bahnhofplatz – Trimmis – Untervaz. Stop list: 1 Austrasse, 2 Obere Au – Bahnhofplatz – Kleinwaldlegg, 2 Obere Au – Bahnhofplatz – Fürstewald, 2 Obere Au – Bahnhofplatz – Stelleweg – Haldenstein, 3 Austrasse – Bahnhofplatz – Trimmis – Untervaz, 4 Austrasse – Bahnhofplatz – Seniorencentrum Cadonau, 6 Bahnhofplatz – City West, 9 Bahnhofplatz – Meiersboden.
- Screenshot 7:** Austrasse – Bahnhofplatz – Seniorencentrum Cadonau. Stop list: 1 Austrasse, 2 Obere Au – Bahnhofplatz – Kleinwaldlegg, 2 Obere Au – Bahnhofplatz – Fürstewald, 2 Obere Au – Bahnhofplatz – Stelleweg – Haldenstein, 3 Austrasse – Bahnhofplatz – Trimmis – Untervaz, 4 Austrasse – Bahnhofplatz – Seniorencentrum Cadonau, 6 Bahnhofplatz – City West, 9 Bahnhofplatz – Meiersboden.
- Screenshot 8:** Bahnhofplatz – City West. Stop list: 1 Bahnhofplatz, 2 Obere Au – Bahnhofplatz – Kleinwaldlegg, 2 Obere Au – Bahnhofplatz – Fürstewald, 2 Obere Au – Bahnhofplatz – Stelleweg – Haldenstein, 3 Austrasse – Bahnhofplatz – Trimmis – Untervaz, 4 Austrasse – Bahnhofplatz – Seniorencentrum Cadonau, 6 Bahnhofplatz – City West, 9 Bahnhofplatz – Meiersboden.
- Screenshot 9:** Bahnhofplatz – Meiersboden. Stop list: 1 Bahnhofplatz, 2 Obere Au – Bahnhofplatz – Kleinwaldlegg, 2 Obere Au – Bahnhofplatz – Fürstewald, 2 Obere Au – Bahnhofplatz – Stelleweg – Haldenstein, 3 Austrasse – Bahnhofplatz – Trimmis – Untervaz, 4 Austrasse – Bahnhofplatz – Seniorencentrum Cadonau, 6 Bahnhofplatz – City West, 9 Bahnhofplatz – Meiersboden.

JSON

JSON

JSON (JavaScript Object Notation) ist ein schlankes Datenaustauschformat, das für Menschen einfach zu lesen und zu schreiben und für Maschinen einfach zu parsen (Analysieren von Datenstrukturen) und zu generieren ist.

JSON

JSON ist ein universelles Daten-Austauschformat. Es wird von fast allen Programmiersprachen unterstützt.

Die Daten werden nach einfachen Regeln in einer Textdatei strukturiert zusammengefasst.

JSON ist ein textbasiertes Daten-Austauschformat.

Daten werden in einer bestimmten Text-Struktur von einem Sender (Server) zu einem Empfänger (Webseite) übertragen.

JSON-Regeln

Wie ein JavaScript-Objekt mit vier Unterschieden:

1. Eigenschaftnamen müssen in doppelten Anführungszeichen stehen.
2. Zeichenketten müssen in doppelten Anführungszeichen stehen.
Einfache Anführungszeichen funktionieren nicht.
3. Den Eigenschaftswert **undefined** gibt es in JSON nicht.
4. In JSON können keine Methoden definiert werden.

JSON – Übung 1

```
/* JSON | Übung 1 | 01_JSON_erreichen
/* Erstellen Sie in der Datei extern/kanton1.json
/* aus dem unten stehenden Code für ein JS-Objekt
/* einen valieden JSON-Code
/*
/* Unterschiede zu JS-Objekten:
/*   Es sind nur doppelte Anführungszeichen erlaubt
/*   Alle Eigenschaftnamen (links vom Doppelpunkt) müssen in Anführungszeichen stehen.
/* ****
{

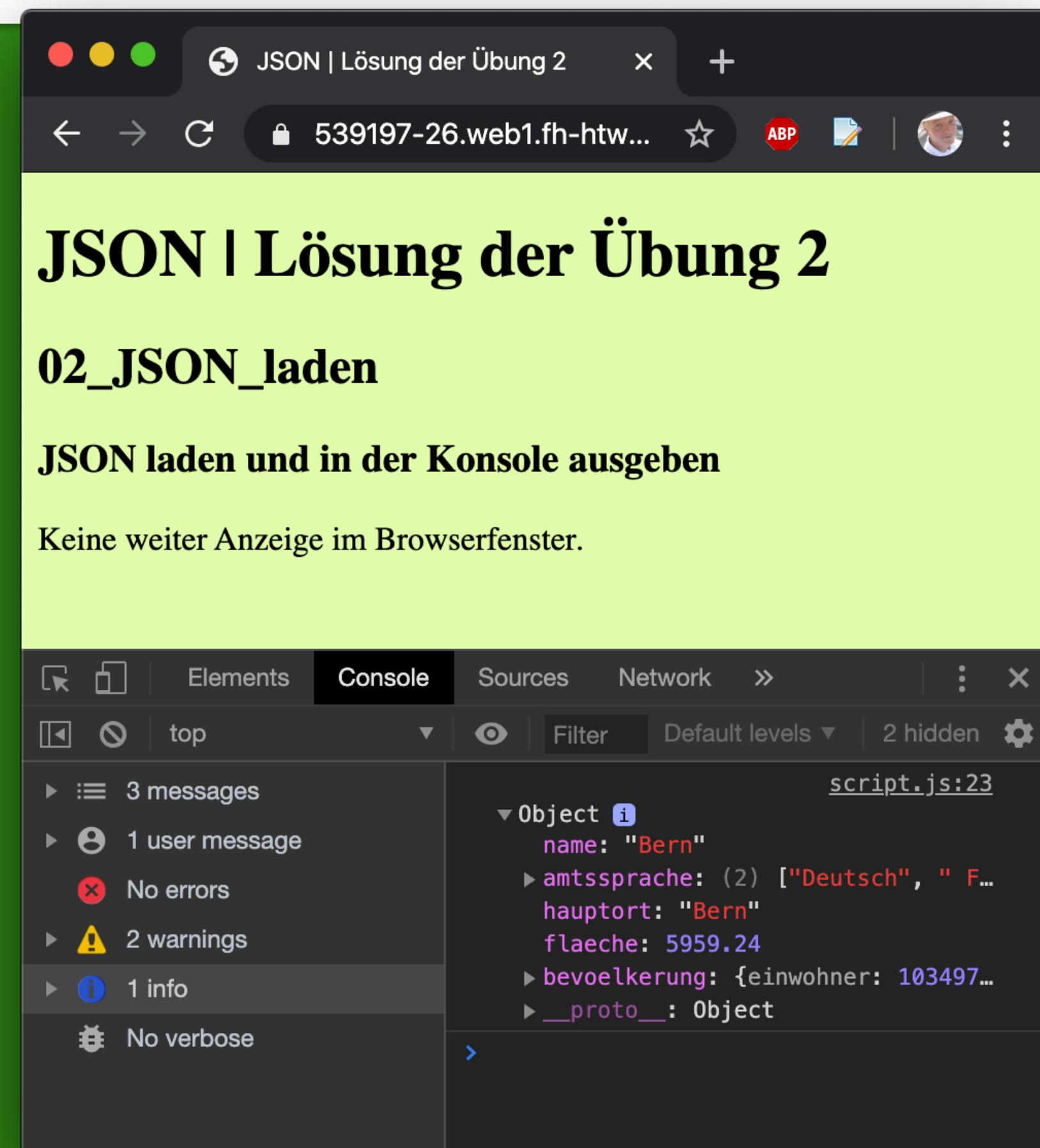
  name : "Bern",
  amtssprache : ["Deutsch", " Französisch"],
  hauptort : "Bern",
  flaeche : 5959.24,
  bevoelkerung : {
    einwohner : 1034977,
    quote : "2,9 %"
  }
}
```

fetch() – JSON response

```
// Mit fetch() -Funktion die externe Datei 'extern/person1.json' laden.  
fetch('extern/person1.json')  
  .then((response) => {  
    // Definiert, welches Format die Antwort hat (wichtig für den nächsten Teil)  
    // !!!!! hier JSON !!!!!!  
    // fetch erwartet jetzt eine validen JSON-Zeichenkette  
    // Die empfangenen JSON-Daten werden von JS direkt in ein Objekt umgewandelt.  
    return response.json();  
})  
  .then((data) => {  
    // Da JavaScript oben mitgeteilt haben, dass der empfangene Text eine JSON-Zeichenkette ist,  
    // können wir die empfangenen Daten direkt als JavaScript-Objekt-Variable speichern.  
    // Ausgabe von data in der Konsole des Browsers.  
    console.log(data);  
})  
  .catch(function(error) {  
    console.log('Error: ' + error.message);  
});
```

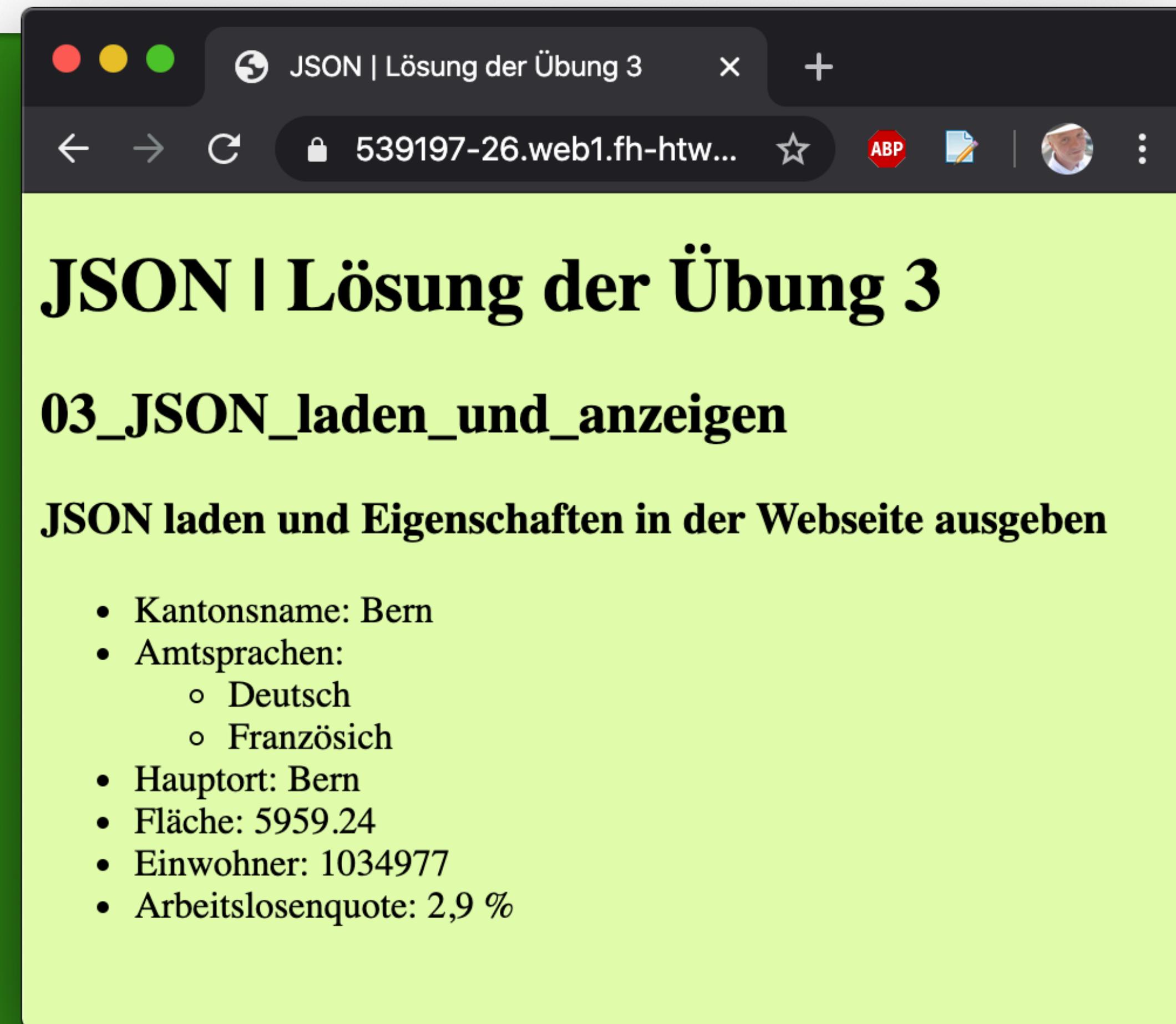
JSON – Übung 2

```
/* JSON | Übung 2 | 02_JSON_laden
/* 1. Laden Sie mit fetch() die Datei 'extern/kanton1.json'.
/* 2. Stellen sie im response ein, welchen Datentyp fetch() zu erwarten hat.
/* 3. Geben Sie mit console.log() die empfangenen Daten in der Konsole Ihres Browsers aus.
/* *****
```



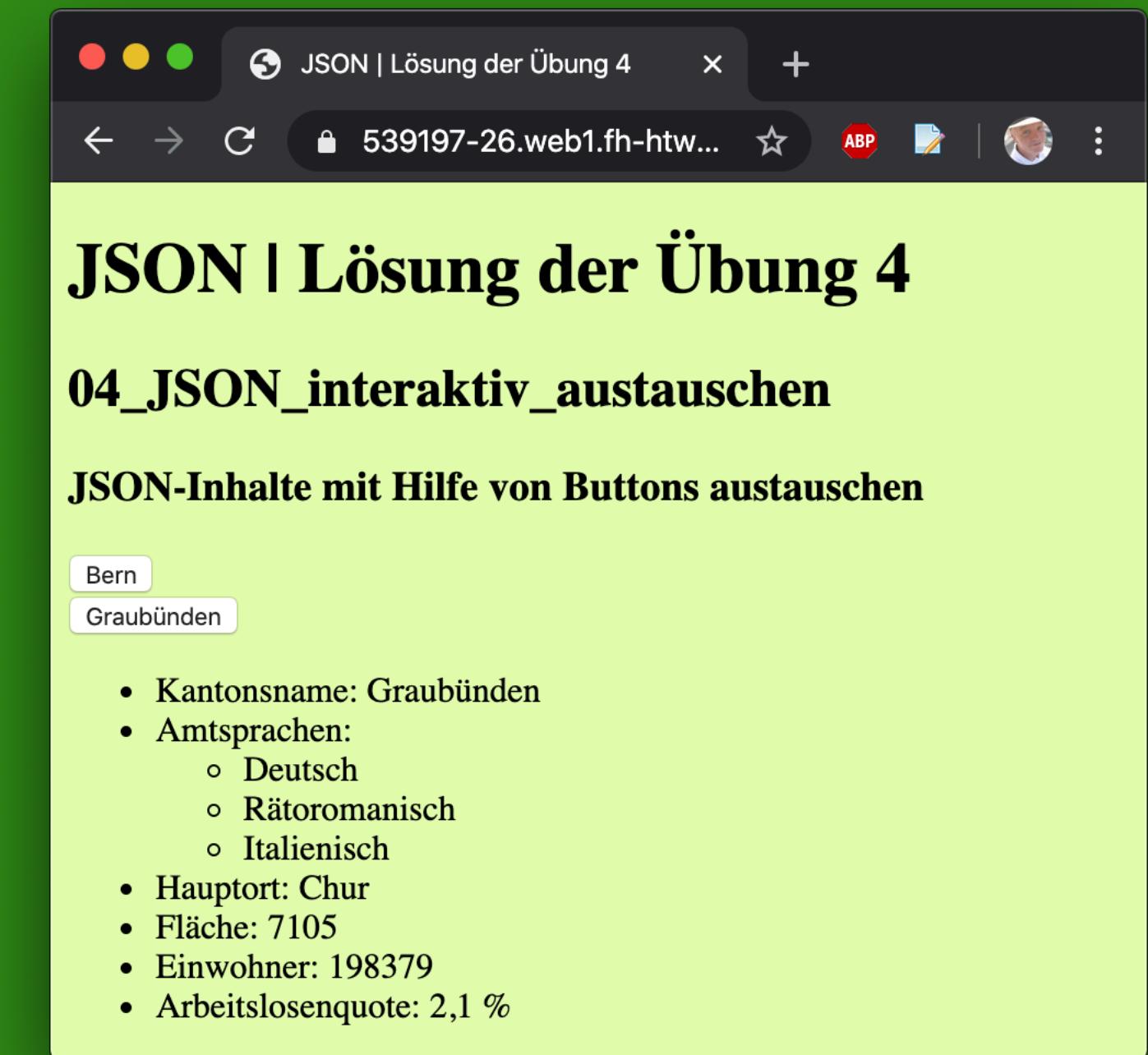
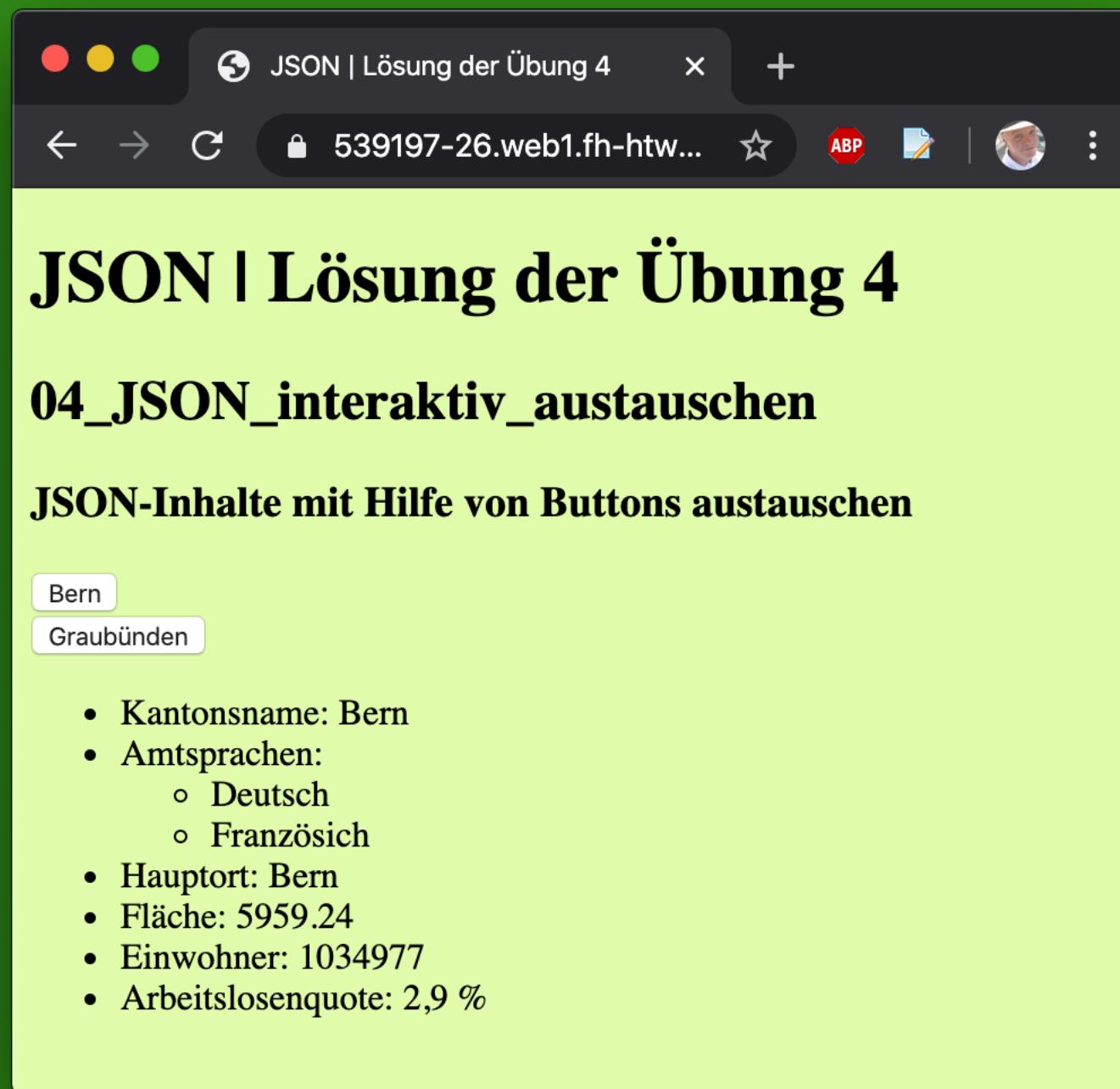
JSON – Übung 3

```
/* JSON | Übung 3 | 03_JSON_laden_und_anzeigen
/* 1. Laden Sie mit fetch() die Datei 'extern/kanton1.json'.
/* 2. Stellen sie im response ein, welchen Datentyp fetch() zu erwarten hat.
/* 3. Geben Sie mit console.log() die empfangenen Daten in der Konsole Ihres Browsers aus.
/* *****
```



JSON – Übung 4

```
/* JSON | Übung 4 | 04_JSON_interaktiv_austauschen
/* 1. Laden Sie mit fetch() die Datei 'extern/kanton1.json'.
/* 2. Stellen sie im response ein, welchen Datentyp fetch() zu erwarten hat.
/* 3. Geben Sie mit console.log() die empfangenen Daten in der Konsole Ihres Browsers aus.
/* *****
```



JSON– Übung 4

```
* JSON | Aufgabe 1 | 01_JSON_Busplan
/* Funktionalisieren Sie die Buttons der zugehörigen Datei so,
/* Dass beim Klick auf einen Button die passende JSON–Datei geladen,
/* der Inhalt aufbereitet und als HTML–Dargestellt wird.
/* Die JSON–Dateien befinden sich im Verzeichnis "extern".
/* *****
```

Feedback

anonymes Feedback