

JavaScript-Objekte | JSON | AJAX

Einführung

Die Themen JavaScript-Objekte, JSON und AJAX 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

JSON ?

JSON ist ein textbasiertes Daten-Austauschformat.

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

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

Beispiel

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

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

JavaScript-Objekt

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

JSON

JSON = JavaScript **O**bject **N**otation

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

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.

AJAX ?

Daten mit Hilfe von JavaScript dynamisch vom Server laden.

Lernziele: AJAX

Mit AJAX Daten vom Server laden können
Kontrollieren, ob Ladevorgang erfolgreich war
geladene Daten in der Webseite darstellen

Ziel

Daten vom Server laden und auf der Webseite anzeigen

JSON

AJAX

JS-Objekte

Hinweise

Immer auf einem Server testen!

AJAX benötigt zwingend einen Server!

Alle Übungs- und Lösungsdateien auf [Github](#).

Joint Degree Bachelor Studiengang

Multimedia Production



Fachhochschule Graubünden
University of Applied Sciences



Berner
Fachhochschule

JavaScript-Objekte

(JS-Objekte)

https://www.w3schools.com/js/js_objects.asp

<https://developer.mozilla.org/de/docs/Learn/JavaScript/Objects/Basics>

Eigenschaften von JS-Objekten

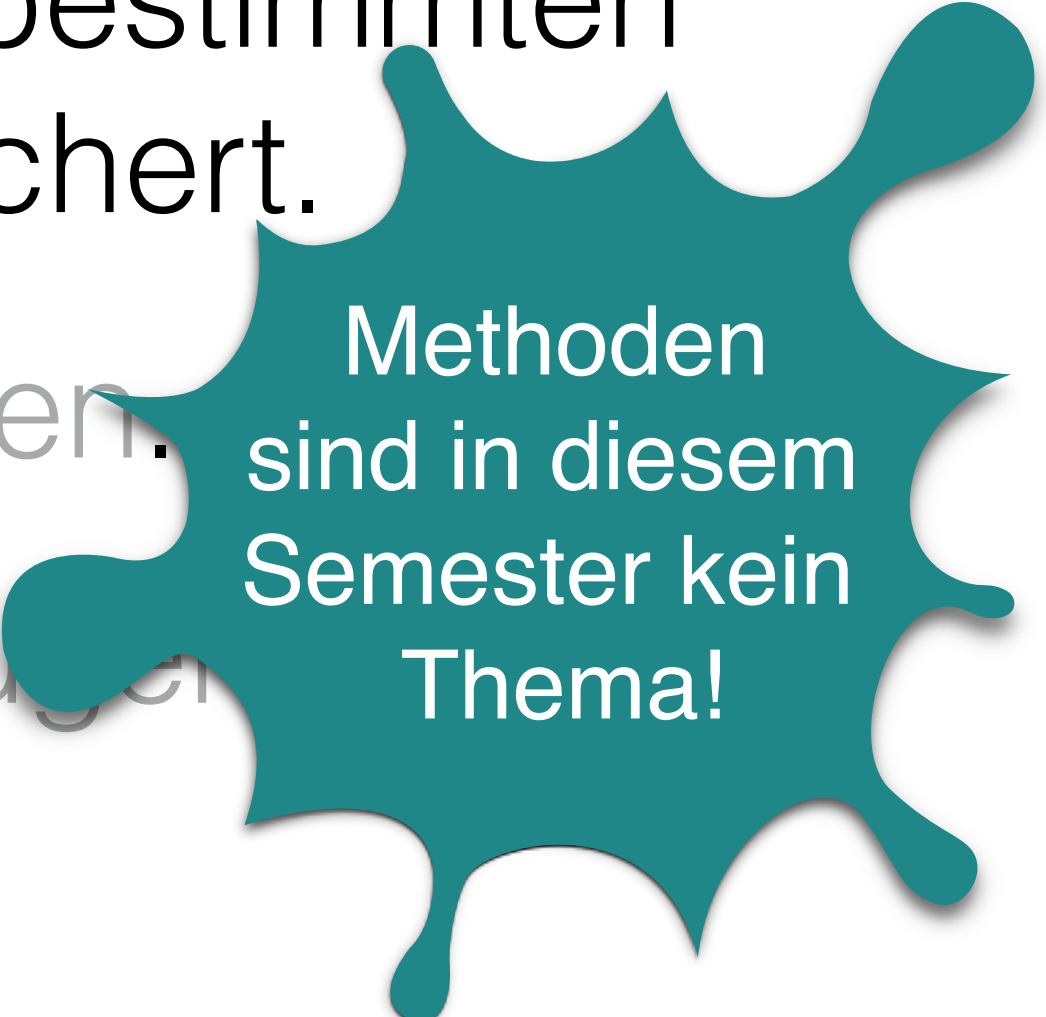
Ein Objekt funktioniert als eine Zuordnungsliste, die unter bestimmten Namen weitere Unterobjekte, auch Member genannt, speichert.

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.



Methoden
sind in diesem
Semester kein
Thema!

Syntax

Ein Objekt wird immer durch geschweifte Klammern eingeschlossen.

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

```
let beispiel = {  
}
```

Syntax - Objekt

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

Eigenschaftswert

Eigenschaftswert

Eigenschaftswert

oder Bezeichner

oder Schlüssel

oder Key

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

Eigenschaftsnamen werden **nicht** in Anführungszeichen geschrieben.

Bei Eigenschaftsnamen wird Gross- und Kleinschreibung unterschieden.

Konvention: Eigenschaftsnamen werden kleingeschrieben.

Syntax

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

The diagram illustrates the syntax for object literals. It shows a code snippet with three properties: 'ganzzahl' with value 42, 'fliesskommazahl' with value 13.37, and 'eigenschaftsname' with value 'Eigenschaftswert'. Callout boxes highlight the following rules: a comma is required after the first two properties, but no comma is allowed after the final property 'eigenschaftsname'.

Mehrere Eigenschaften werden durch Kommata getrennt.

Nach dem letzten Eigenschaftswert darf kein Komma stehen.

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 Eigenschaftsnamen 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

Übung

gutes Beispiel

Geschweifte Klammern öffnen ...

```
let gutesBeispiel = {
```

```
  ganzzahl : 42,
```

Punkt als Dezimaltrenner, ohne Anführungszeichen

```
  fliesskommazahl : 13.37,
```

Zeichenketten können in einfachen oder doppelten Anführungszeichen stehen.

```
  zeichenkette : 'Hallo Welt',
```

```
  unterobjekt : {
```

Geschweifte Klammern öffnen ...

```
    vorname : 'Urs',
```

```
    nachname : 'Thöny'
```

```
  },
```

... und schliessen das Unterobjekt.

```
  einArray : ['foo', 'bar'],
```

Ein Array steht auch im JS-Objekt immer in eckigen Klammern.

```
  undefinierbar : undefined,
```

`undefined` ist korrekt, wird jedoch selten als Wert vergeben.

```
  nix : null,
```

`null` ist korrekt, wird jedoch selten als Wert vergeben.

```
  entweder0der : true
```

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

```
}
```

... und schliessen das JS-Objekt



SCHLECHTES Beispiel



Eigenschaftsnamen 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,  
  entweder0der : 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.

Joint Degree Bachelor Studiengang

Multimedia Production



HTW Chur

Hochschule für Technik und Wirtschaft
University of Applied Sciences



Dot.Syntax
Zugriff auf Eigenschaften

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 **Eigenschaftsnamen** an.

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

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

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


Übung

Joint Degree Bachelor Studiengang

Multimedia Production



Fachhochschule Graubünden
University of Applied Sciences



Berner
Fachhochschule

JSON

<https://www.json.org/json-de.html>

<https://developer.mozilla.org/de/docs/Learn/JavaScript/Objects/JSON>

JSON

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

JSON

JSON ist ein textbasiertes Datenformat.

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

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

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

JSON = JavaScript **O**bject **N**otation

JSON-Regeln

Wie ein JavaScript-Objekt mit vier Unterschieden:

1. Eigenschaftsnamen 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.

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

JavaScript-Objekt

Mögliche Datentypen in JSON

- Zeichenketten
- Zahlen
- (JSON-) Objekte
- Arrays
- Boolsche Werte
- "null"

Übung

Joint Degree Bachelor Studiengang

Multimedia Production



Fachhochschule Graubünden
University of Applied Sciences



Berner
Fachhochschule

AJAX

<https://www.mediaevent.de/javascript/fetch.html>

AJAX war nie neu

AJAX war nie eine neue Erfindung, sondern eine Kombination verschiedener Webtechniken.

Schon 1998 existierten diese Techniken.

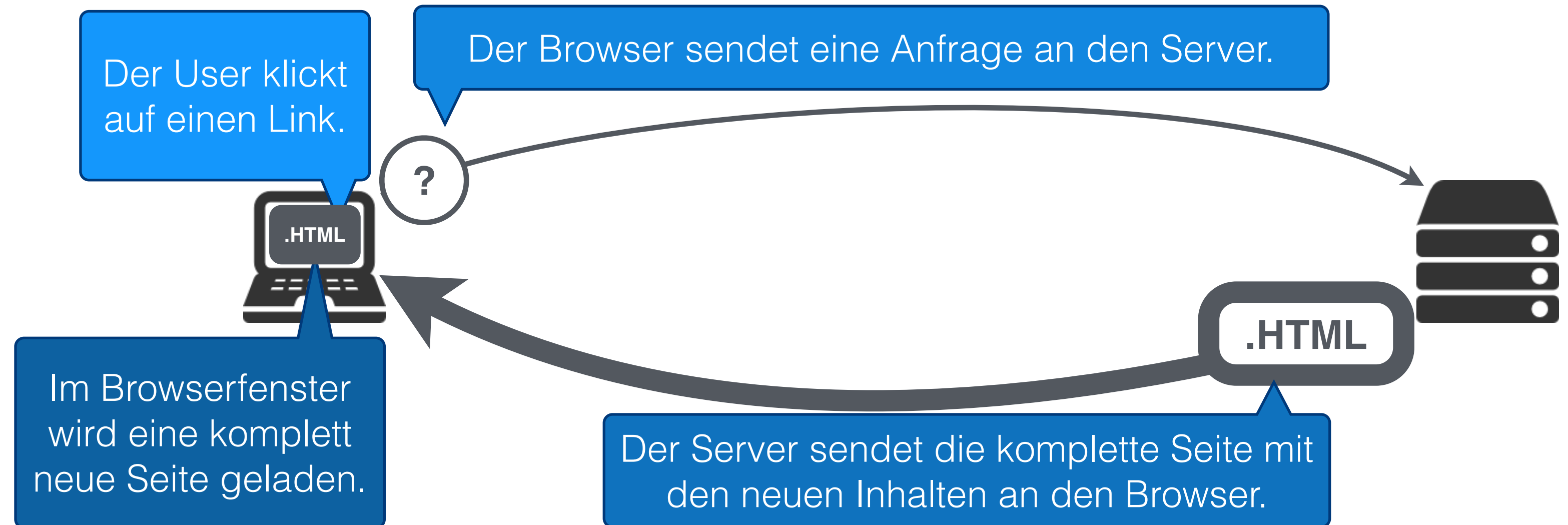
Der Begriff AJAX gibt es seit ca. 2004.

Was bedeutet AJAX

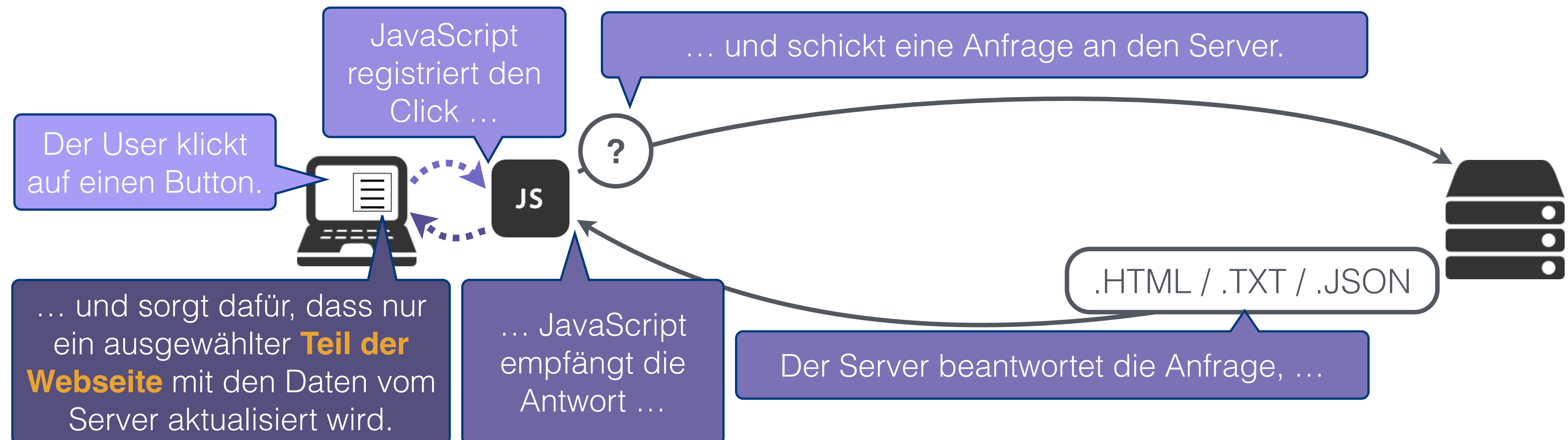
Aynchronous
JavaScript
And
XML

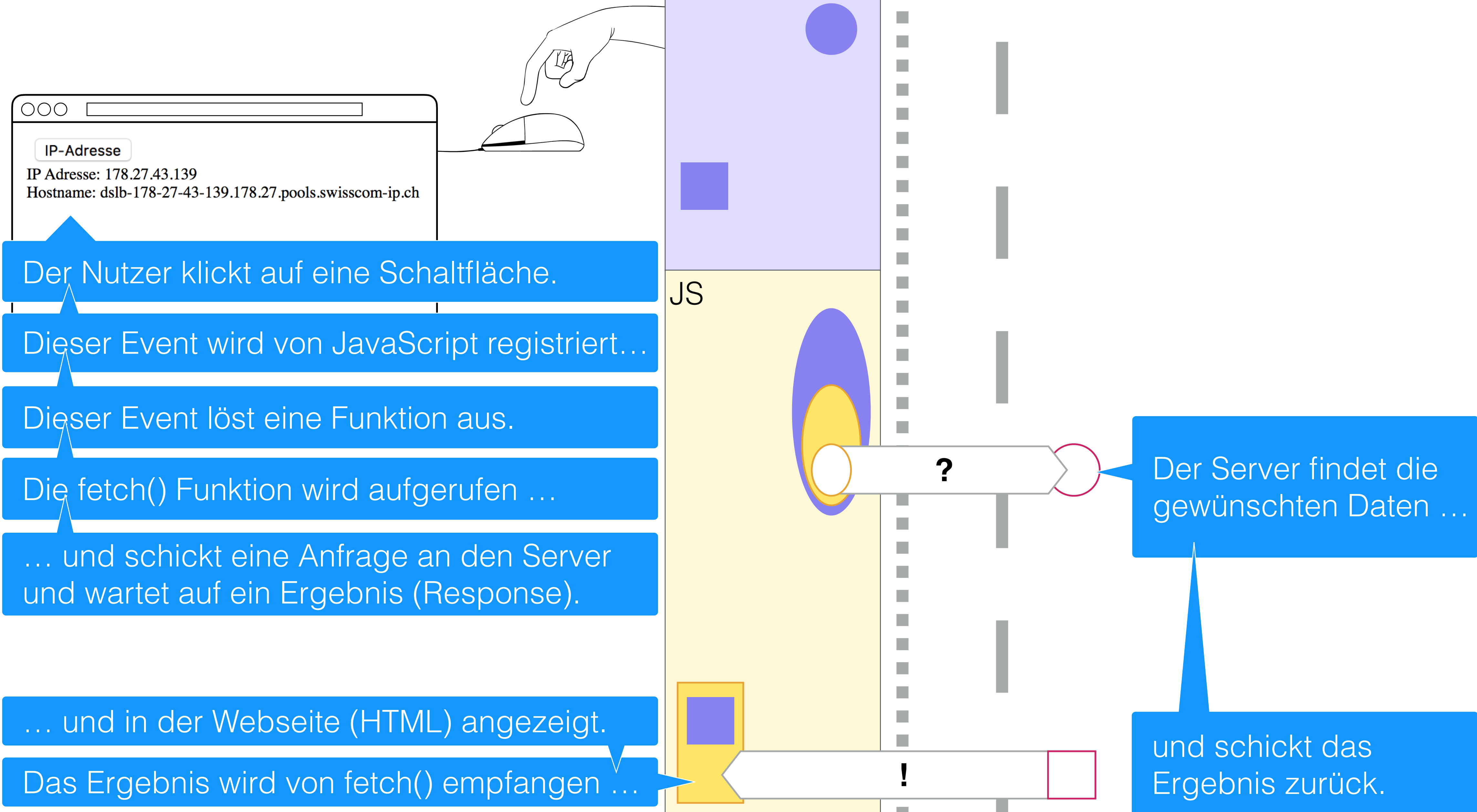
Serverkommunikation

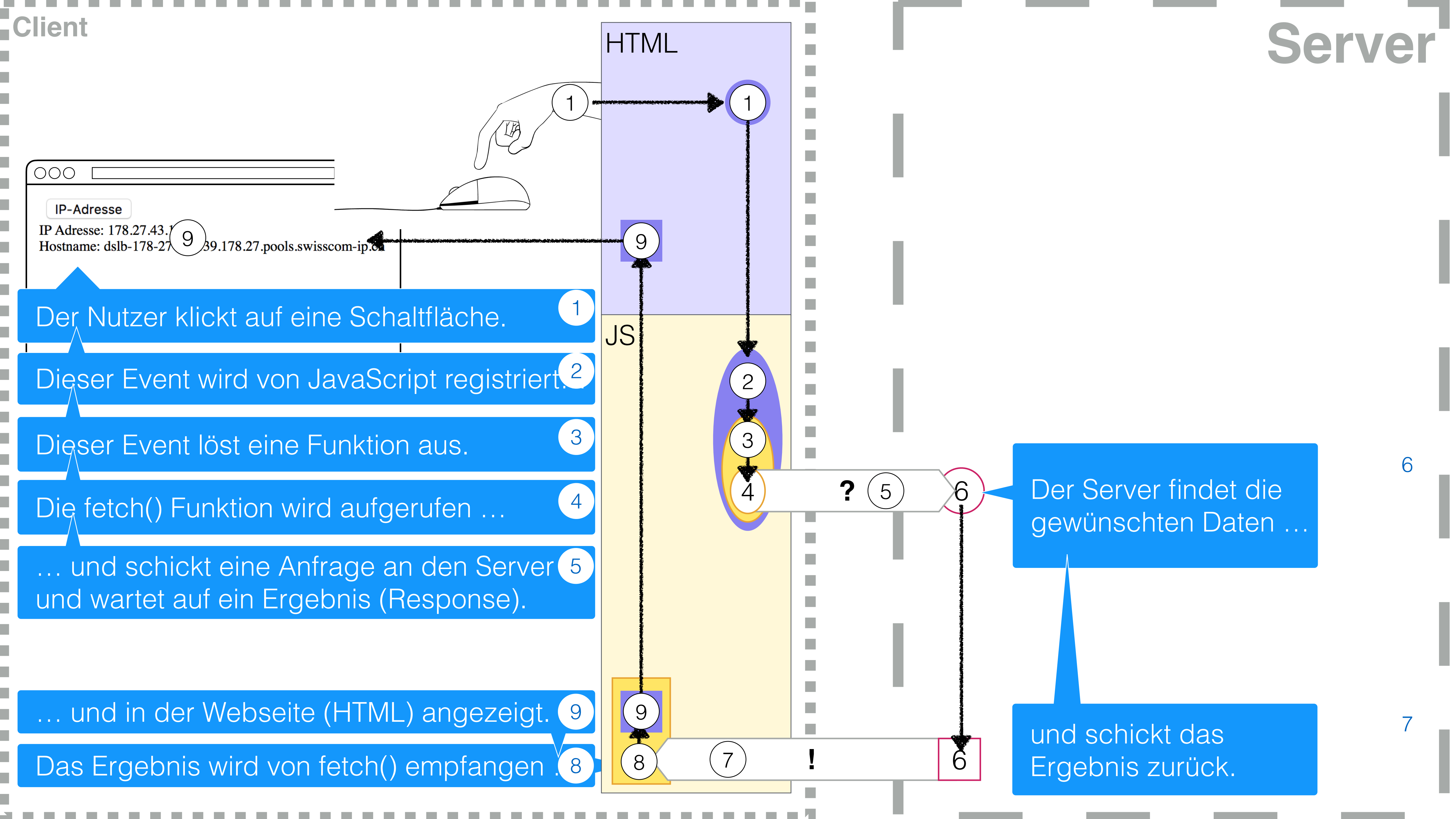
bisher



mit AJAX







Übung

Joint Degree Bachelor Studiengang

Multimedia Production



Fachhochschule Graubünden
University of Applied Sciences



AJAX mit fetch()

<https://www.mediaevent.de/javascript/fetch.html>

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()

Start

```
fetch('extern/text.txt')
```

dann

```
.then((response) => {  
  return response.text();  
})
```

dann

```
.then((data) => {  
  console.log(data);  
})
```

sonst

```
.catch(function(error) {  
  console.log('Error: ' + error.message);  
});
```

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);  
  });
```

Übung

Joint Degree Bachelor Studiengang

Multimedia Production



Fachhochschule Graubünden
University of Applied Sciences



Berner
Fachhochschule

