



Ajax en Fetch API

Veerle Ongenae



Overzicht

- Ajax
 - Concept



Ajax

- <https://www.youtube.com/watch?v=RDo3hBL1rfA>
- AJAX = Asynchronous JavaScript and XML
- Bestaande standaarden anders gebruiken
 - XMLHttpRequest-object
 - Vanuit javascript HTTP-berichten sturen en ontvangen
 - Javascript + DOM
 - CSS
 - XML
- Sneller webpagina's
 - Indruk: lokaal werken



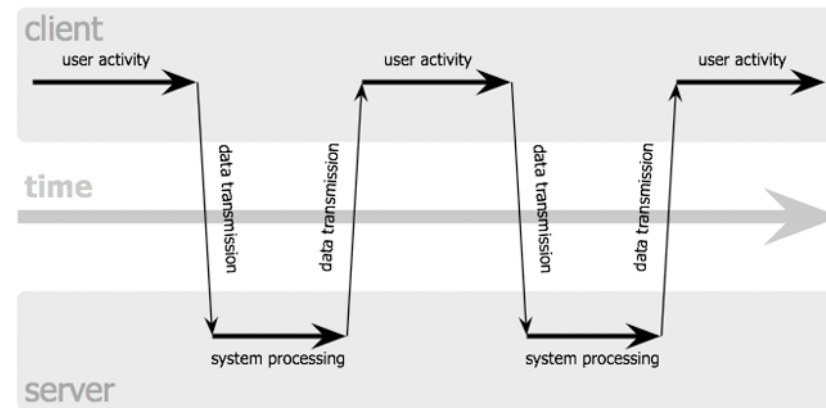
Essentie Ajax

- Klassieke webpagina
 - HTTP-bericht naar server
 - Browser wacht op antwoord
 - Browser toont HTTP-antwoord
 - Nieuw document vervangt volledig vorig document
- Ajax
 - Delen van het HTML-document kunnen vervangen worden
 - Gebruiker kan verder werken (asynchrone communicatie met de server)

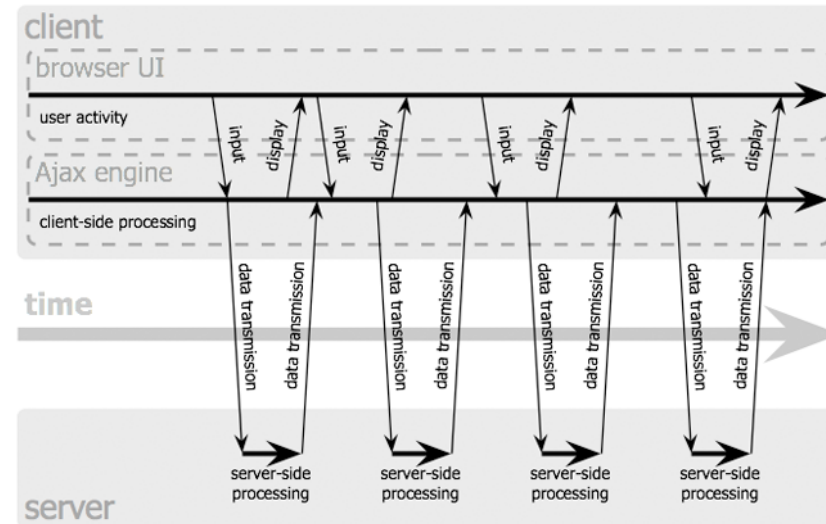


Essentie Ajax

classic web application model (synchronous)



Ajax web application model (asynchronous)

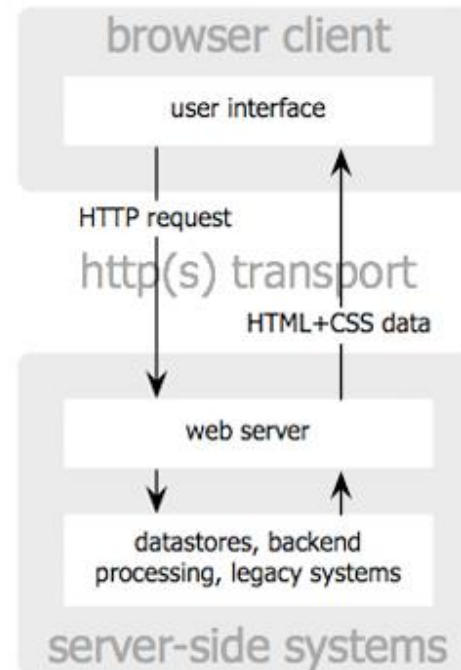


Jesse James Garrett / adaptivepath.com



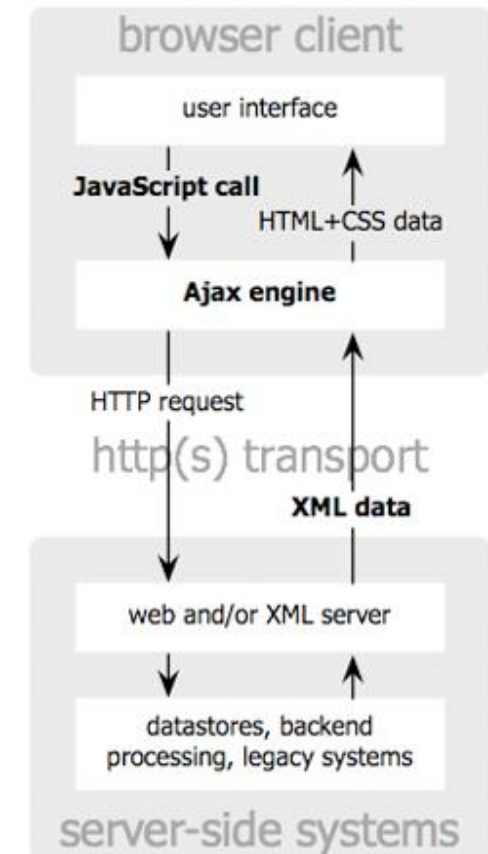
Realisatie Ajax

- Ajax-engine in browser
- HTTP-antwoord
 - XML, JSON, ...



**classic
web application model**

Jesse James Garrett / adaptivepath.com



**Ajax
web application model**

<http://www.adaptivepath.com/ideas/essays/archives/000385.php>

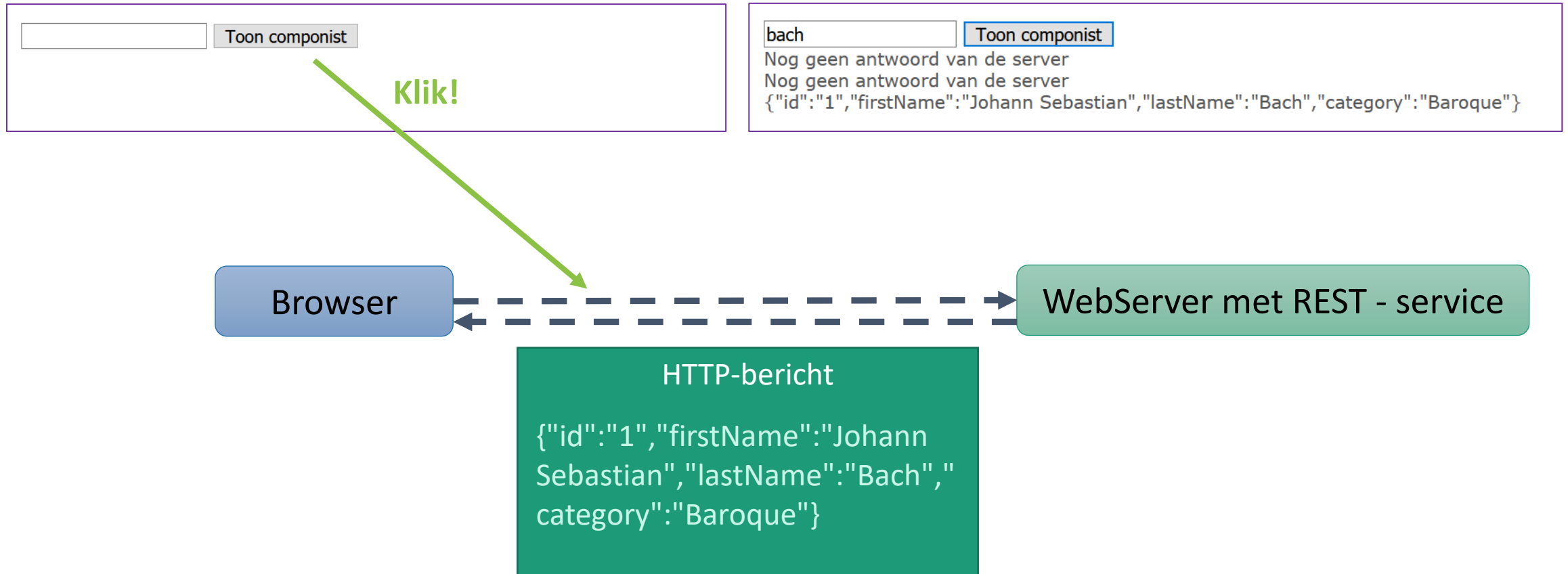


Overzicht

- Ajax
 - Concept
 - Voorbeeld



Voorbeeld – componist.html



Voorbeeld – componist.html

Toon componist

bach

Toon componist

Nog geen antwoord van de server
Nog geen antwoord van de server
{ "id": "1", "firstName": "Johann Sebastian", "lastName": "Bach", "category": "Baroque" }

```
<input type="text" id="componist">
```

```
<input type="button" id="knop" value="Toon componist" >
```

```
<div id="resultaat"></div>
```



Voorbeeld – componist.js

Toon componist

Nog geen antwoord van de server

Nog geen antwoord van de server

`{"id": "1", "firstName": "Johann Sebastian", "lastName": "Bach", "category": "Baroque"}`

```
function haalComponist() {...}

function ontvang(req) {...}

function haalNaam() {...}

function init() {
  let knop = document.getElementById( elementId: "knop" );
  knop.onclick=haalComponist;
}

init();
```



Componist.js – bericht sturen naar server

```
function haalComponist() {
  let aanvraag = new XMLHttpRequest();
  if (aanvraag !== null) {
    aanvraag.open( method: "GET",
      url: "http://localhost:8080/componist?naam="+haalNaam(), async: true);
    aanvraag.onreadystatechange = () => ontvang(aanvraag); // uit te voeren functie bij antwoord
    aanvraag.send( body: null); // geen corpus
  }
}

function haalNaam() {
  let componistInvoer = document.getElementById( elementId: "componist");
  return componistInvoer.value;
}

function ontvang(req) {...}
```


Nog geen antwoord van de server

Nog geen antwoord van de server

```
{ "id": "1", "firstName": "Johann Sebastian", "lastName": "Bach", "category": "Baroque" }
```

localhost:8080/componist?naam=b X +

id: "1"

firstName: "Johann Sebastian"

lastName: "Bach"

category: "Baroque"

Browser

WebServer met REST - service



Componist.js – antwoord server verwerken

Toon componist

Nog geen antwoord van de server

Nog geen antwoord van de server

```
{ "id": "1", "firstName": "Johann Sebastian", "lastName": "Bach", "category": "Baroque" }
```

```
function ontvang(req) {  
  let bericht = document.getElementById( elementId: "resultaat");  
  let tekst;  
  if (req.readyState === 4 && req.status === 200) { // aanvraag gelukt  
    tekst = document.createTextNode(req.responseText); // info ophalen  
  } else {  
    tekst = document.createTextNode( data: "Nog geen antwoord van de server");  
  }  
  bericht.appendChild(tekst);  
  bericht.appendChild(document.createElement( tagName: "br"));  
}
```

Browser

WebServer met REST - service

HTTP-bericht

```
{ "id": "1", "firstName": "Johann  
Sebastian", "lastName": "Bach", "category": "Baroque" }
```



Overzicht

- Ajax
 - Concept
 - Voorbeeld
 - API



XMLHttpRequest-object

- Communicatie met server
 - Verstuurt HTTP-berichten
 - Ontvangt HTTP-berichten
- Oproepen functie indien antwoord ontvangen (**call back**)
- **HTTP-antwoordbericht**
 - Inhoud
 - Tekst
 - XML
 - JSON → `JSON.parse(request.responseText)`
 - Omzetting tekst → JSON-object
 - Javascript Object Notation



Opstellen HTTP-aanvraag

- Methode **open**

- Connectie met server

```
open(aanvraagMethode, URL)
open(aanvraagMethode, URL, asynchroon)
open(aanvraagMethode, URL, asynchroon, gebruikersnaam)
open(aanvraagMethode, URL, asynchroon, gebruikersnaam, wachtwoord)
```

- aanvraagMethode: GET, POST, ...
- URL: bron met data
- asynchroon
 - Default: true
 - Er wordt niet gewacht op antwoord van de server



Callback-functie instellen

- onreadystatechange

- Uit te voeren functie bij verandering “readystatechange”

- readyState

- 0: request not initialized
- 1: server connection established
- 2: request received
- 3: processing request
- 4: request finished and response is ready



Aanvraag versturen naar server

- **send**(corpus)
 - Corpus
 - inhoud HTTP-bericht
 - Parameters
 - null
- Na instellen callback-functie



Antwoordbericht - eigenschappen

- **status**
 - HTTP-antwoord-status
- **responseText**
 - Corpus antwoord
 - string
- Alternatieven
 - **responseXML** → document



Overzicht

- Ajax
 - Concept
 - Voorbeeld
 - API
- Fetch API
 - Principe



Fetch API

- API om bronnen op te halen in een client script
- Vergelijkbaar met XMLHttpRequest
- Gebruikt promises

⇒ AJAX werd Ajax

- Term
- Client haalt data van de server om webpagina aan te passen



Fetch - principe

url

opties

- optioneel
- JS-object
 - headers
 - method
 - ...

antwoord-object

```
fetch(  
  'http://domain/service',  
  { method: 'GET' }  
)  
  .then( response => response.json() )  
  .then( json => console.log(json) )  
  .catch( error => console.error('error:', error) );
```

omzetten naar JS



Overzicht

- Ajax
 - Concept
 - Voorbeeld
 - API
- Fetch API
 - Principe
 - Parameters



Parameters doorsturen naar de server

- **HTTP-parameters**: informatie van een HTML-formulier (form-tag) doorsturen naar de server (webapp, REST-service)
- Antwoord afhankelijk van doorgestuurde data
- Een aantal opties
 - GET: **querystring** in de URL
 - `https://host:poort/pad?naam=waarde&naam1=waarde1&...`
 - POST: in **body**
 - `naam=waarde&naam1=waarde1&...` (= idem indien formulier)
 - `{"naam":"waarde", "naam1":"waarde1", ... }` (=JSON-object)





Voorbeeld Fetch API: querystring en formulier

localhost:8080/componist.html

bach Toon componist

```
{ "id": "1", "firstName": "Johann Sebastian", "lastName": "Bach", "category": "Baroque" }  
{ "id": "1", "firstName": "Johann Sebastian", "lastName": "Bach", "category": "Baroque" }
```

Network

Status	Methode	Domein	Bestand	Initiator	Type	Overgebracht	Grootte
200	GET	localhost:8080	componist.html	BrowserTabChild.js...	html	773 B	455 B
200	GET	localhost:8080	componist.css	stylesheet	css	1,19 KB	904 B
200	GET	localhost:8080	componistFetch.js	script	js	1,70 KB	1,38 KB
404	GET	localhost:8080	favicon.ico	FaviconLoader.jsm:...	json	gebufferd	133 B
200	POST	localhost:8080	componist	componistFetch.js:...	json	242 B	80 B
200	GET	localhost:8080	componist?naam=bach	componistFetch.js:...	json	242 B	80 B

JSON

```
{  
  "id": "1",  
  "firstName": "Johann Sebastian",  
  "lastName": "Bach",  
  "category": "Baroque"  
}
```

Aanvraag

naam: "bach"

1 naam=bach





Voorbeeld Fetch API: querystring en formulier – bericht naar server

```
function haalComponist() {  
  // parameters  
  let params = new URLSearchParams( init: {naam: haalNaam()});  
  // POST-aanvraag  
  fetch( input: "http://localhost:8080/componist",  
    init: {  
      method: 'post',  
      headers: {  
        "Content-type": "application/x-www-form-urlencoded; charset=UTF-8"  
      },  
      body: params  
    })  
    .then(ontvangTekst).then(response => response.text()).then(toonTekst).catch(fout);  
  // GET - aanvraag  
  let url = new URL( url: "http://localhost:8080/componist")  
  url.search = params  
  fetch(url).then(ontvangTekst).then(response => response.text()).then(toonTekst).catch(fout);  
}
```

```
{ "id": "1", "firstName": "Johann Sebastian", "lastName": "Bach", "category": "Baroque" }  
{ "id": "1", "firstName": "Johann Sebastian", "lastName": "Bach", "category": "Baroque" }
```

querystring

url

opties

antwoord
afhandelen





Voorbeeld Fetch API: antwoord afhandelen

bach

Toon componist

```
{ "id": "1", "firstName": "Johann Sebastian", "lastName": "Bach", "category": "Baroque" }  
{ "id": "1", "firstName": "Johann Sebastian", "lastName": "Bach", "category": "Baroque" }
```

```
fetch(url).then(ontvangTekst).then(response => response.text()).then(toonTekst).catch(fout);
```

promise

statuscode geen 200

body => tekst

resultaat verwerken in HTML

```
function ontvangTekst(response) {  
  if (!response.ok) {  
    throw new Error("Fout" + response.status);  
  }  
  // aanvraag gelukt  
  return response;  
}
```

```
function toonTekst(data) {  
  let resultaat = document.getElementById( elementId: "resultaat");  
  resultaat.appendChild(document.createTextNode(data));  
  resultaat.appendChild(document.createElement( tagName: "br"));  
}
```

foutboodschap

```
function fout(error) {  
  // aanvraag mislukt  
  let resultaat = document.getElementById( elementId: "resultaat");  
  resultaat.appendChild(document.createTextNode(error)); // info ophalen  
}
```



Parameters doorsturen naar de server

- Een aantal opties

- GET: **querystring** in de URL
 - `https://host:poort/pad?naam=waarde&naam1=waarde1&...`
- POST: in **body**
 - `naam=waarde&naam1=waarde1&...` (= idem indienen formulier)
 - `{"naam":"waarde", "naam1":"waarde1", ... }` (=JSON-object)

- Antwoord

- Tekst
- JSON



Voorbeeld: Fetch API - JSON

webbrowser

Naam componist:

Naam componist:

Bedrich Smetana
Georges Bizet
Louis-Hector Berlioz
Benjamin Britten
Leonard Bernstein
Bela Bartok
Johann Sebastian Bach
Johannes Brahms
Ludwig van Beethoven

onzichtbare
tabel

<http://netbeans.org/kb/docs/web/ajax-quickstart.html>

localhost:8080/vulaan?letters=be

← → ↻ 🏠

📄 localhost:8080/vulaan?letters=be

JSON Onbewerkte gegevens Headers

Opslaan Kopiëren Alles samenvouwen Alles uitvouwen 🔍 JSON filteren

▼ 0:

id: "8"

firstName: "Ludwig van"

lastName: "Beethoven"

category: "Classical"

▼ 1:

id: "20"

firstName: "Louis-Hector"

lastName: "Berlioz"

category: "Romantic"

▼ 2:

id: "37"

firstName: "Bedrich"

lastName: "Smetana"

category: "Romantic"

▼ 3:

id: "41"

firstName: "Bela"

lastName: "Bartok"

category: "Post-Romantic"

▼ 4:

id: "42"

firstName: "Leonard"

lastName: "Bernstein"

category: "Post-Romantic"

▼ 5:


id: "43"

firstName: "Benjamin"

lastName: "Britten"

category: "Post-Romantic"

REST-service





Voorbeeld Fetch API – JSON - events

```
function getElementY(element) {...}  
  
function init() {  
  // listener voor tekstvak  
  tekstvakComponist = document.getElementById( elementId: "naam_componist");  
  tekstvakComponist.addEventListener( type: "keyup", vulAan);  
  
  // positie tabel bepalen op basis van positie rij  
  completeTable = document.getElementById( elementId: "suggesties");  
  autoRow = document.getElementById( elementId: "suggestie_kolom");  
  completeTable.style.top = getElementY(autoRow) + "px";  
}  
  
init();
```

event eventhandler

naam_componist

Naam componist:

Bedrich Smetana
Georges Bizet
Louis-Hector Berlioz
Benjamin Britten
Leonard Bernstein
Bela Bartok
Johann Sebastian Bach
Johannes Brahms
Ludwig van Beethoven

suggesties





Voorbeeld Fetch API - JSON – HTTP-vraag naar server

HTTP-bericht naar server
<http://localhost:8080/vulaan?letters=ba>

```
// JSON parsen om informatie componisten op te halen
function parseMessages(componisten) {
    maakTabelLeeg();
    if (componisten !== undefined) {
        for (let componist of componisten) {
            appendComposer(componist);
        }
    }
}

function maakTabelLeeg() {
    let tabel = document.getElementById( elementId: "suggesties");
    tabel.textContent = ""; // tabel met mogelijkheden leeg maken
}

// een rij toevoegen aan de de tabel met mogelijke componisten
function appendComposer(componist) {...}

function vulAan() {
    let naam = document.getElementById( elementId: "naam_componist");
    let params = new URLSearchParams( init: {letters: naam.value});
    let url = new URL( url: "http://localhost:8080/vulaan");
    url.search = params;
    fetch(url).then(response => response.json()).then(parseMessages)
        .catch(error => console.log(error));
}
```

eventhandler

body => JS-object

Overzicht

- Ajax
 - Concept
 - Voorbeeld
 - API
- Fetch API
 - Principe
 - Parameters
- Crossdomain calls



Opmerking cross domain calls

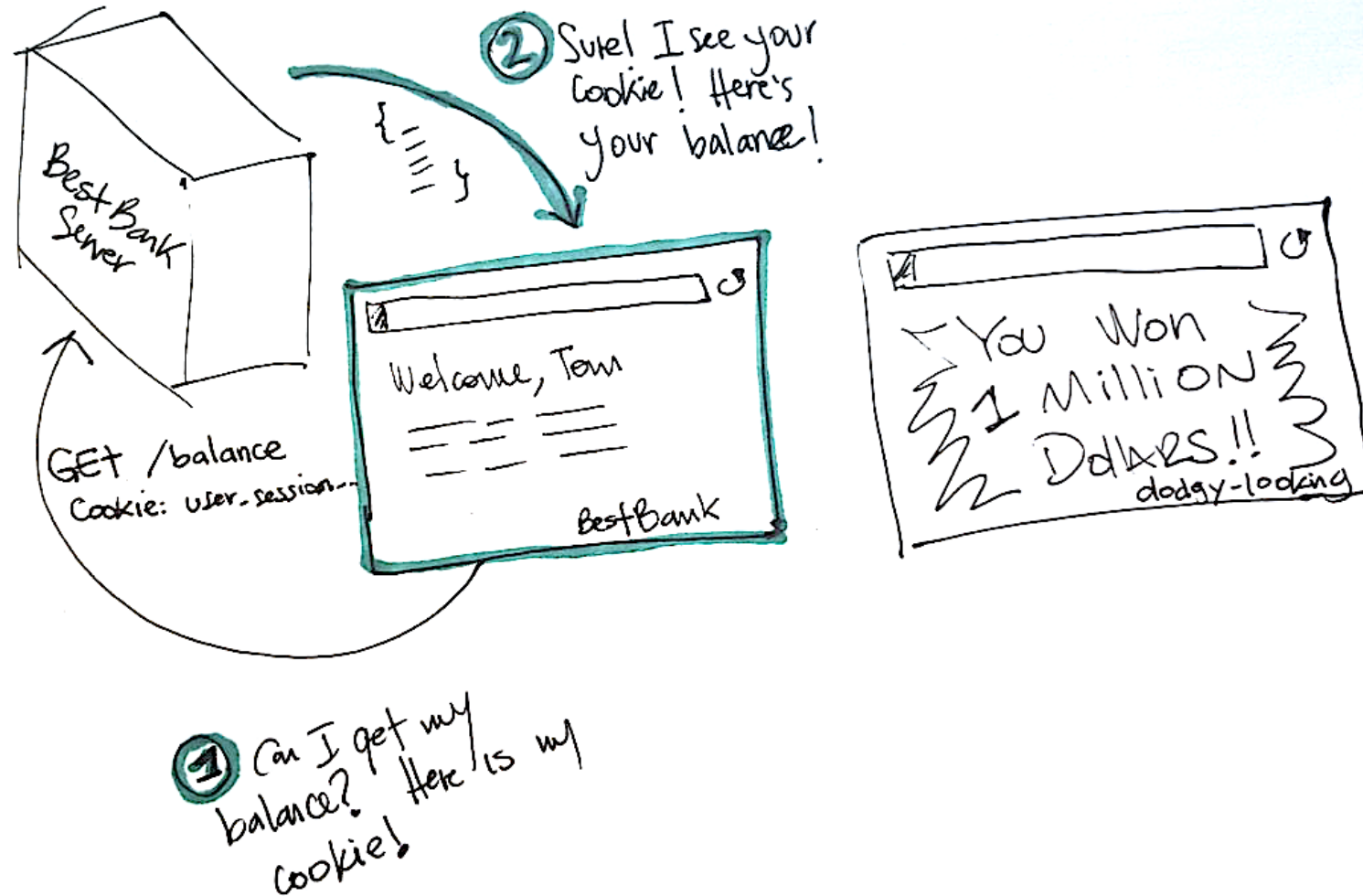
- Ajax-aanvragen naar ander domein dan huidige webpagina
 - Webpagina
 - <http://localhost:8080/index.html>
 - Ajax-aanvraag
 - <http://localhost:50633/ComponistService.svc/GetComponisten?letters=ba>

! Cross-origin-aanvraag geblokkeerd: de Same Origin Policy staat het lezen van de externe bron op <http://localhost:50633/ComponistService.svc/GetComponisten?letters=ba> niet toe. Dit is te verhelpen door de bron naar hetzelfde domein te verplaatsen of door CORS in te schakelen.

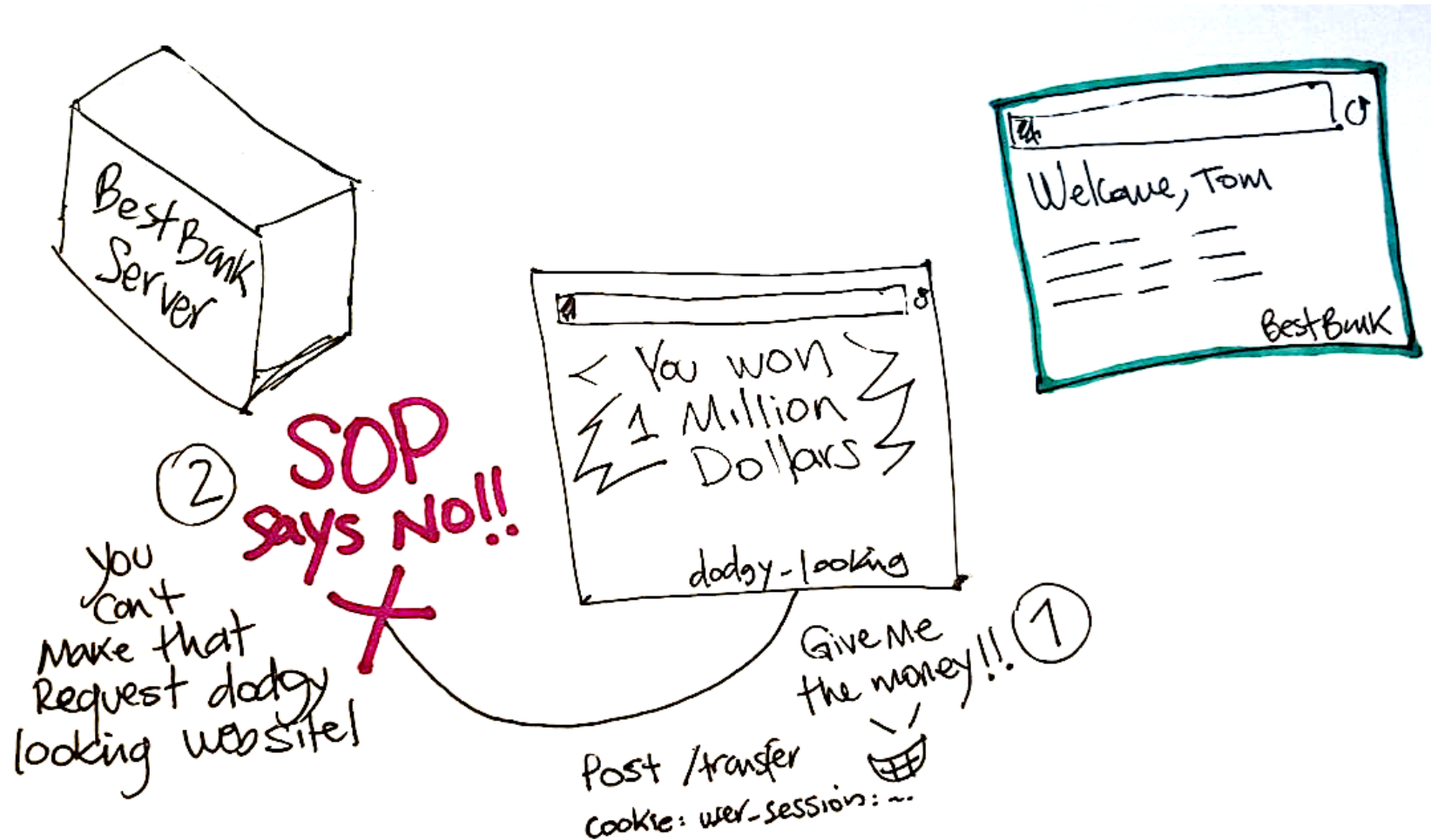
- Mogelijk oplossing
 - Server side proxy
 - Webpagina → server
 - Server → ander domein
 - Ander domein → server
 - Server → webpagina
 - CORS
 - Eventueel JSONP



Same origin policy – waarom?



Same origin policy – waarom?



Cross-origin resource sharing (CORS)

- Webpagina
 - Aanvraag naar bronnen van een ander domein
 - Niet toegelaten
- CORS
 - Server kan toelating geven om geraadpleegd te worden door een ander domein
 - Nieuwe HTTP-headers



Cross-origin resource sharing (CORS)

browser

Origin: <http://www.example-social-network.com>

server

Access-Control-Allow-Origin: <http://www.example-social-network.com>

- <https://www.youtube.com/watch?v=Vusw5r9umFM>



JSONP

- JSON with padding
- Browsers laten aanvraag naar ander domein wel toe voor <script>-tag
 - src-attribuut → resultaat URL = functie → functie uitgevoerd
- Opmerking
 - Beveiligingsrisico's: malafide services of "gehackte" services
- https://www.youtube.com/watch?v=QjtO9TQCJ_8



Overzicht

- Ajax
 - Concept
 - Voorbeeld
 - API
- Fetch API
 - Principe
 - Parameters
- Crossdomain calls

