



# **JavaScript 2**

## **Lektion => 4**

Utbildare: Mahmud Al Hakim

**NACKADEMIN**

# Lektionstillfällets mål

## Mål med lektionen

- Ajax
- XML
- JSON
- Introduktion till API
- Att läsa: sid. 368–384
- Arbetsmetod
  - Teori och praktik varvas under lektionen

NACKADEMIN

# Kort summering av föregående lektion

- Vi har gått igenom några jQuery Plugins.

- Vi har utvecklat en enkel inköpslista

- Källkoden finns här

<https://github.com/mahmudalhakim/JavaScript-2/blob/master/Lektion-03/inkopslista.html>

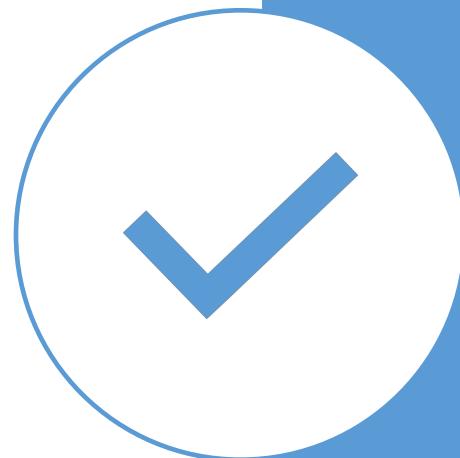
- Demo finns här

<https://mahmudalhakim.github.io/JavaScript-2/Lektion-03/inkopslista.html>

# Callback function

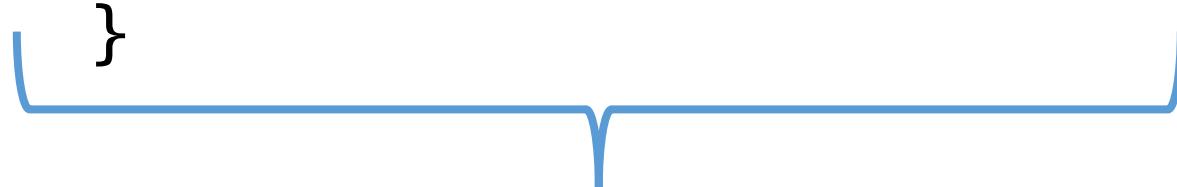
“A callback function is a function passed into another function as an argument, which is then invoked inside the outer function to complete some kind of routine or action”.

[https://developer.mozilla.org/en-US/docs/Glossary/Callback\\_function](https://developer.mozilla.org/en-US/docs/Glossary/Callback_function)



# Callback function

```
function greeting(name) {  
    alert('Hello ' + name);  
}  
  
function processUserInput(callback) {  
    var name = prompt('Please enter your name.');//  
    callback(name);  
}  
  
processUserInput(greeting);
```



[https://developer.mozilla.org/en-US/docs/Glossary/Callback function](https://developer.mozilla.org/en-US/docs/Glossary/Callback_function)

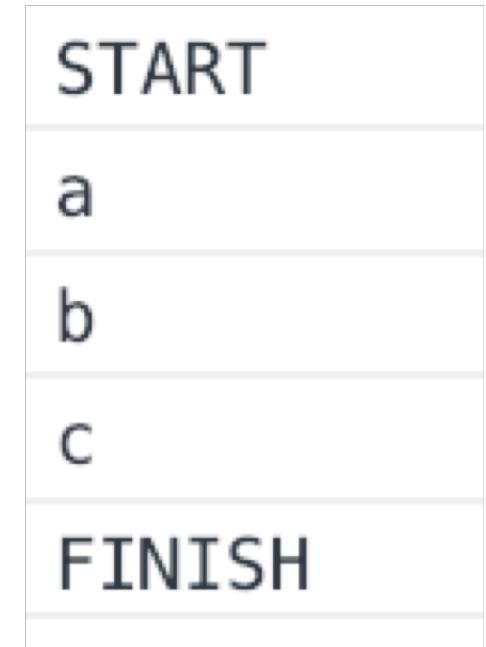
# Synchronous callback

```
var array1 = ['a', 'b', 'c'];

console.log('START');

array1.forEach(function(element) {
    console.log(element);
});

console.log('FINISH');
```



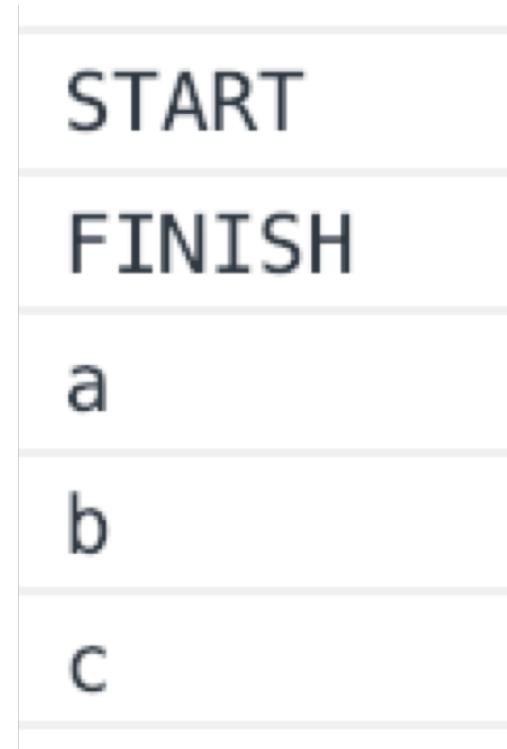
[https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/Array/forEach](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/forEach)

# Asynchronous callback

```
console.log('START');
```

```
setTimeout(function() {
  for (const i of array1) {
    console.log(i);
  }
}, 1000);
```

```
console.log('FINISH');
```



# Vad är Ajax?

- Ajax står för ”Asynchronous JavaScript and XML”.
- Termen Ajax myntades 2005 av Jesse James Garrett.  
[https://en.wikipedia.org/wiki/Jesse\\_James\\_Garrett](https://en.wikipedia.org/wiki/Jesse_James_Garrett)
- Ajax är ett samlingsnamn för flera olika tekniker som kan användas för att bygga applikationer för webben med bättre interaktivitet.
- Kända tjänster som använder Ajax:  
Gmail, Google Maps och Facebook.



# Varför Ajax?

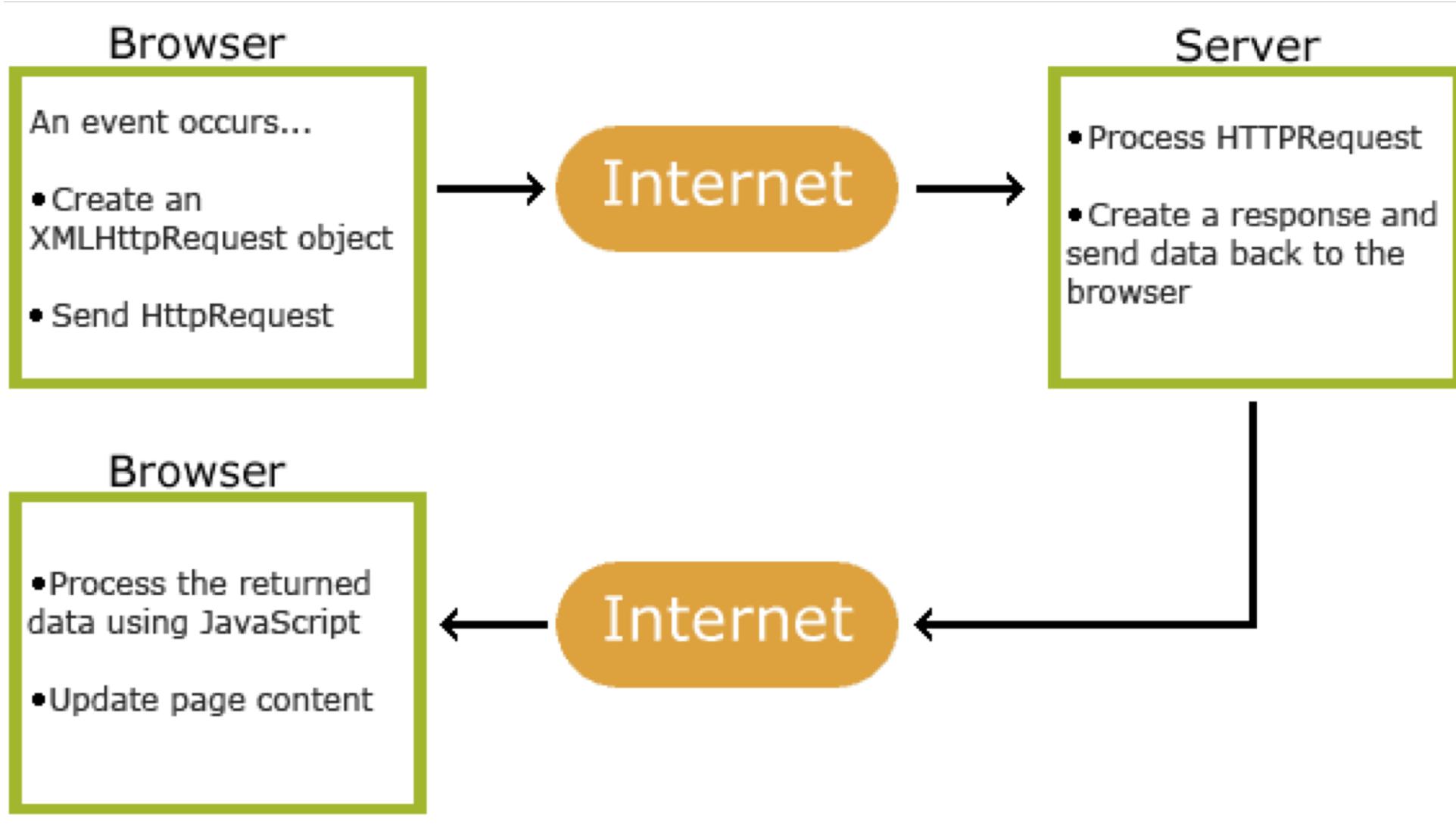
- Ajax kan underlätta och snabba på interaktionen med webbapplikationer på många sätt.
  1. Validering av formulär i realtid.
  2. Automatisk textkomplettering.
  3. Sofistikerade kontroller och effekter i användargränssnittet.
  4. Partiell submit.
  5. Sidor som liknar vanliga skrivbordsapplikationer.

Källa: <https://sv.wikipedia.org/wiki/AJAX>

AJAX is a developer's dream, because you can:

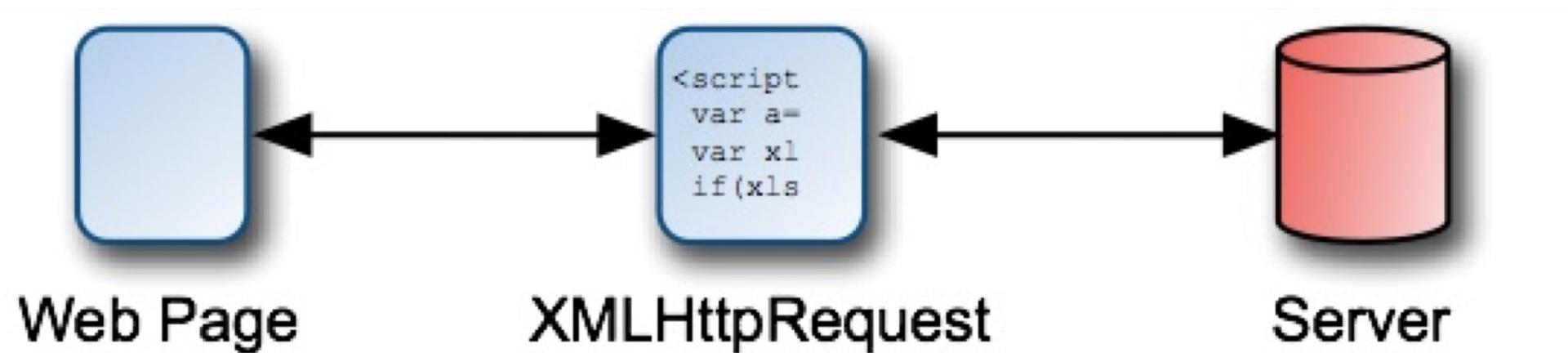
- Read data from a web server - after the page has loaded
- Update a web page without reloading the page
- Send data to a web server - in the background

# How Ajax Works



# XMLHttpRequest

- XMLHttpRequest är ett JavaScript-objekt.
- XMLHttpRequest används i applikationer som använder sig av Ajax.
- XMLHttpRequest tillåter en webbsida att göra anrop till webbservern utan att sidan laddas om.
- Med hjälp av XMLHttpRequest kan du överföra XML, JSON och annan textbaserad information mellan en klient och en server



# Skapa ett XMLHttpRequest-objekt

```
let xhr = new XMLHttpRequest();
console.log(xhr);
```

▼ XMLHttpRequest ⓘ

- onabort: null
- onerror: null
- onload: null
- onloadend: null
- onloadstart: null
- onprogress: null
- onreadystatechange: null
- ontimeout: null
- readyState: 0
- response: ""
- responseText: ""
- responseType: ""
- responseURL: ""
- responseXML: null
- status: 0
- statusText: ""
- timeout: 0

► upload: XMLHttpRequestUpload

withCredentials: false

► \_\_proto\_\_: XMLHttpRequest

# Skicka en request till en server

- För att skicka en request till en server behöver du först ange typ och url med hjälp av metoden open().

```
xhr.open('GET', 'demo.txt', true);
```

- Sedan behöver du skicka requesten med metoden send()

```
xhr.send();
```

# CORS – Cross-Origin Resource Sharing

- För att kunna kommunicera med servern så måste du ha en lokal HTTP-server .
- VS Code har ett tillägg som heter ”Live Server”.
- Live Server skapar en server lokalt som du kan nå via följande adress <http://127.0.0.1:5500>



**Live Server** ritwickdey.liveserver

Ritwick Dey | 3 435 558 | ★★★★★

Launch a development local Server with live reload fe...

[Disable ▾](#) [Uninstall](#)

blocked by CORS policy: Cross origin requests are only supported for protocol schemes: http, data, chrome, chrome-extension, https.

# Egenskapen onreadystatechange

Property	Description
onreadystatechange	Defines a function to be called when the readyState property changes
readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
status	200: "OK" 403: "Forbidden" 404: "Page not found" For a complete list go to the <a href="#">Http Messages Reference</a>
statusText	Returns the status-text (e.g. "OK" or "Not Found")

# Onreadystatechange – Exempel

```
xhr.onreadystatechange = function() {  
    console.log(this.readyState);  
    if (this.readyState === 4 && this.status === 200) {  
        document.getElementById('demo').innerHTML =  
            this.responseText;  
    }  
};
```

# Skapa en load-funktion



```
function load(url, callback) {  
    let xhr = new XMLHttpRequest();  
  
    xhr.onreadystatechange = function() {  
        if (this.readyState == 4 && this.status == 200) {  
            callback(this);  
        }  
    };  
  
    xhr.open('GET', url, true);  
    xhr.send();  
}
```

# Hämta ett dokument med load()

```
<button id="loadBtn1">Hämta ett textdokument</button>
<div id="demo1"></div>

let loadBtn1 = document.getElementById('loadBtn1');
loadBtn1.addEventListener("click", function() {
    load('demo1.txt', demo1);
});

function demo1(xhr) {
    document.getElementById('demo1').innerHTML = xhr.responseText;
}
```

# Hämta flera dokument

```
let loadBtn2 = document.getElementById('loadBtn2');
loadBtn2.addEventListener('click', function() {
    load('demo1.txt', demo1);
    load('demo2.txt', demo2);
});

function demo1(xhr) {
    document.getElementById('demo1').innerHTML = xhr.responseText;
}

function demo2(xhr) {
    document.getElementById('demo2').innerHTML = xhr.responseText;
}
```

# XML – EXTENSIBLE MARKUP LANGUAGE

- XML, är ett universellt och utbyggbart märkspråk och en förenklad efterträdare till SGML "Standard Generalized Markup Language".
- Syftet med XML är att kunna utväxla data mellan olika informationssystem.
- För att kunna hantera ett XML-dokument, krävs att sändare och mottagare av informationen har kommit överens om vilka element och attribut som ska kunna användas. Därför måste man ha en gemensam dokumentmall.
- Källa: <https://sv.wikipedia.org/wiki/XML>

# XML – Exempel

```
<?xml version="1.0" encoding="UTF-8"?>
<CATALOG>
  <CD>
    <TITLE>Empire Burlesque</TITLE>
    <ARTIST>Bob Dylan</ARTIST>
    <COUNTRY>USA</COUNTRY>
    <COMPANY>Columbia</COMPANY>
    <PRICE>10.90</PRICE>
    <YEAR>1985</YEAR>
  </CD>
</CATALOG>
```

[https://www.w3schools.com/js/cd\\_catalog.xml](https://www.w3schools.com/js/cd_catalog.xml)

# Hämta XML via Ajax

```
<button id="loadBtn3">Hämta data från XML</button>
<div id="demo3"></div>
```

```
let loadBtn3 = document.getElementById('loadBtn3');
loadBtn3.addEventListener('click', function() {
  load('cd_catalog.xml', demo3);
});
```

# Hämta XML via Ajax – fortsättning

```
function demo3(xhr) {  
    let xml = xhr.responseXML;  
    let ARTIST = xml.getElementsByTagName('ARTIST');  
  
    let list = "<ul>";  
    for (let i = 0; i < ARTIST.length; i++) {  
        list += "<li>" + ARTIST[i].textContent + "</li>";  
    }  
    list += "</ul>";  
  
    document.getElementById('demo3').innerHTML = list;  
}
```

# Övning – Skapa följande tabell från en XML-fil

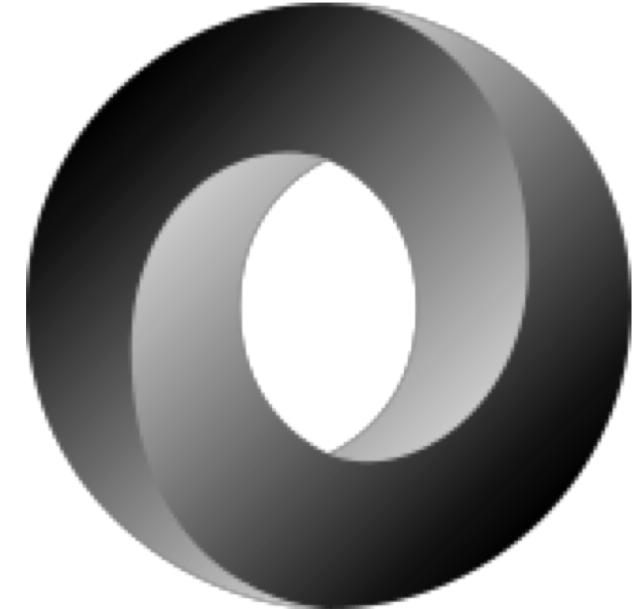
*Arbeta i grupper ca 30 minuter*

Artist	Title
Bob Dylan	Empire Burlesque
Bonnie Tyler	Hide your heart
Dolly Parton	Greatest Hits
Gary Moore	Still got the blues
Eros Ramazzotti	Eros

[https://www.w3schools.com/js/cd\\_catalog.xml](https://www.w3schools.com/js/cd_catalog.xml)

# JSON

- JSON står för **JavaScript Object Notation**.
- JSON är ett kompakt, textbaserat format som används för att utbyta data.
- JSON är ett mindre utrymmeskrävande alternativ till XML.
- OBS! JSON är enbart ett format för data. JSON säger inget om hur data ska presenteras för användaren.
- <https://www.json.org/>



# JSON – Exempel

(spara som demo.json)

```
{  
  "id": 1,  
  "name": "Mahmud Al Hakim",  
  "username": "mahmud",  
  "email": "mahmud@example.com",  
  "phone": "076-1659999",  
  "company": {  
    "name": "Web Academy AB",  
    "address": "Hemfridsvägen, 192 67 Sollentuna",  
    "website": "http://webacademy.se"  
  }  
}
```

# Ladda JSON via Ajax

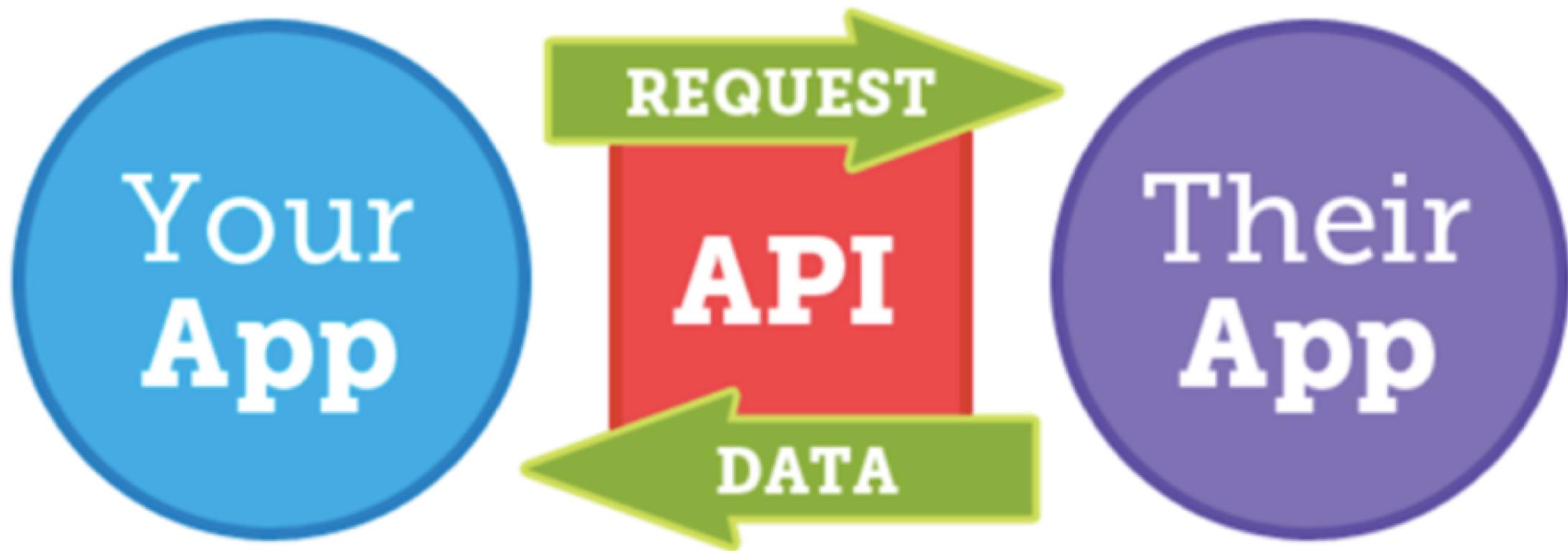
```
let loadBtn = document.getElementById('loadBtn');
loadBtn.addEventListener('click', function() {
    load('demo.json', demo);
});

function demo(xhr) {
    let user = JSON.parse(xhr.responseText);
    console.log(user);
    document.getElementById('demo').innerHTML = user.name;
}
```

# Visa JSON som string med JSON.stringify

```
function demo(xhr) {  
  
    let user = JSON.parse(xhr.responseText);  
  
    document.getElementById('demo6').innerHTML =  
        JSON.stringify(user);  
}
```

# Vad är ett API?



# API – Application Programming Interface

- De flesta programvaror i dagens läge är applikationer som knyter samman annan mjukvarufunktion i olika former och skapar en meningsfull helhet, och denna sammanknytning sker med hjälp av API:er.
- API:er ger alltså möjlighet att på ett strukturerat sätt återanvända redan utvecklad och kvalitetssäkrad mjukvara som har kapslats in i någon form av kodbibliotek.
- Källa:  
[https://sv.wikipedia.org/wiki/Application\\_Programming\\_Interface](https://sv.wikipedia.org/wiki/Application_Programming_Interface)

# REST – Representational State Transfer

- REST eller RESTful webbtjänst är ett IT-arkitekturbegrepp som beskriver hur tjänster för maskin-till-maskin-kommunikation kan tillhandahållas via webbteknologi.
- Tillämpningsexempel kan vara att en mobil applikation, en bot eller en webbserver som fungerar som agent kan accessa innehållet i en databas via en webbplats som är designad för maskiner snarare än för mänskor.
- Vanligen överförs data på JSON-format mellan maskinerna.
- Källa: [https://sv.wikipedia.org/wiki/Representational\\_State\\_Transfer](https://sv.wikipedia.org/wiki/Representational_State_Transfer)

# NamnAPI.se

[Varför](#)[Hur man använder NamnAPI](#)[Exempel](#)[Om](#)[Kommentarer](#)

v1

Like 15

Share

Tweeta

## Varför

Olika namnexempel är ofta förekommande i exempelvis programmeringslitteratur, men de stannar oftast som ett exempel fast än man så ofta behöver dem i olika sorters applikationer eller också annat som är "offline".

Inte allt för sällan när man exempelvis ska göra en databas så behöver man fylla den med exempelkunder för att testa, extremt tråkigt är det att sitta och komma på ett massor med namn som man behöver föra in manuellt.

NamnAPI råder bot på de här problemen.

I NamnAPIs databas så finns de 100 vanligaste tilltalsnamnen för både kvinnor och män, och dessutom de 100 vanligaste efternamnen. Totalt så finns det alltså 20 000 olika kombinationer.

# NamnAPI.se Några endpoints i JSON-format

- <http://api.namnapi.se/v2/names.json>
- <http://api.namnapi.se/v2/names.json?limit=5>
- <http://api.namnapi.se/v2/names.json?gender=male>
- <http://api.namnapi.se/v2/names.json?gender=male&type=firstname>
- <http://api.namnapi.se/v2/names.json?gender=male&type=surname>
- <http://api.namnapi.se/v2/names.json?gender=female&type=firstname>

# Visa ett slumpmässigt namn från NamnAPI.se

```
let loadBtn = document.getElementById('loadBtn');

loadBtn.addEventListener('click', function() {
  let url =
    'https://cors.io/?http://api.namnapi.se/v2/names.json?limit=1';

  load(url, demo);

});
```

## Vad är cors.io

```
let url =
```

```
'https://cors.io/?http://api.namnapi.se/v2/names.json?limit=1';
```

blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource.

- Varför https://cors.io/? före en endpoint?
- Detta är en Proxy för att undvika webbläsarens CORS-blockering.
- Lär mer: <https://developer.chrome.com/extensions/xhr>

## Visa ett slumpmässigt namn från NamnAPI.se – Fortsättning

```
function demo(xhr) {  
    let json = JSON.parse(xhr.responseText);  
    let array = json.names;  
    let name = array[0];  
    console.log(name);  
    document.getElementById('demo').innerText =  
        name.firstname + ' ' + name.surname ;  
}
```

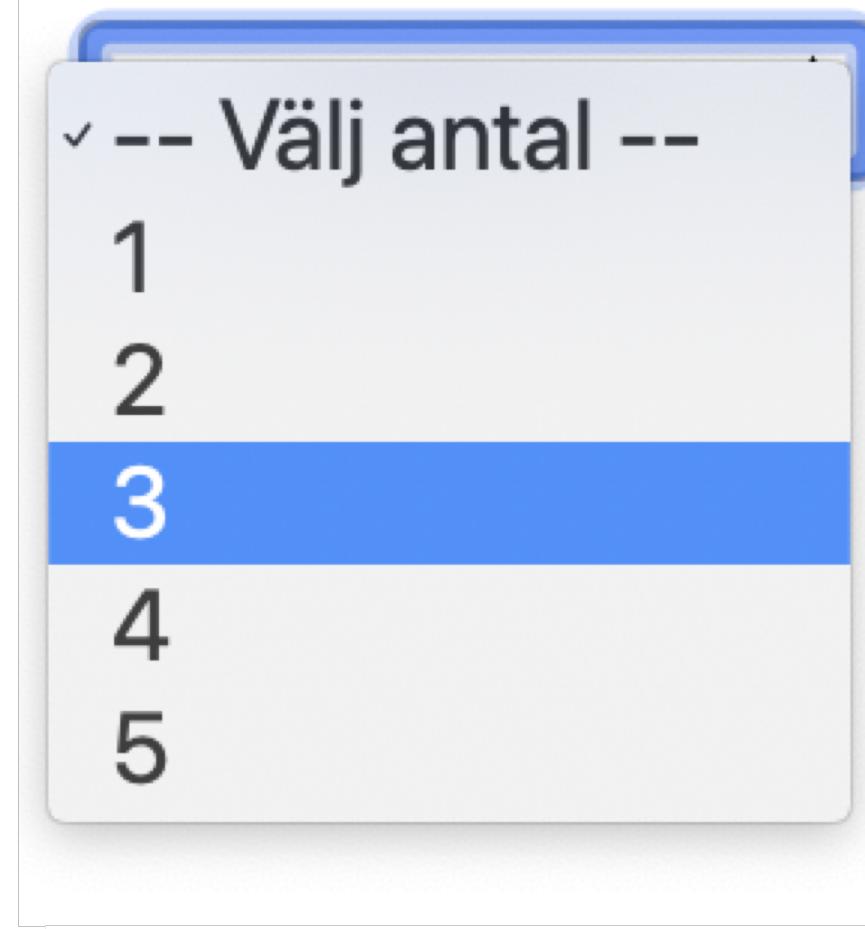
# Övning 1

- Visa 10 slumpmässiga namn från [www.NamnAPI.se](http://www.NamnAPI.se)

Birgit Lindgren  
Moa Sundberg  
Thomas Löfgren  
Rune Pettersson  
Joel Holm  
Kenneth Sundqvist  
Bertil Hansson  
Hanna Persson  
Håkan Engström  
Claes Ek

# Övning 2

- Skapa en dropdown-lista som ger besökaren möjlighet att välja antal slumpmässiga namn från [www.NamnAPI.se](http://www.NamnAPI.se)
- Tips:  
Skapa dropdown-listan med HTML-taggen Select och 5 st. option-taggar.  
Lägg till event-lyssnaren change till select.



**Louise Danielsson  
Sebastian Henriksson  
Therese Pålsson**

# Summering av dagens lektion

- Ajax
- XML
- JSON
- Introduktion till API
- Att läsa: 368–384
- Reflektioner kring dagens lektion?

**NACKADEMIN**

# Framåtblick inför nästa lektion

- Under nästa lektion kommer vi att jobba mer med Ajax och APIer.