



JavaScript

JS-biblioteket jQuery & Ajax

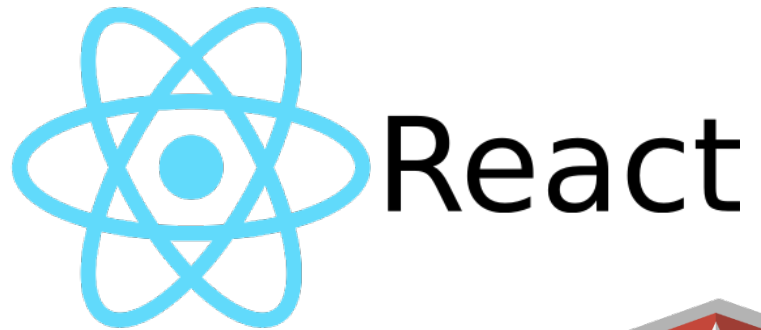
# Dagens föreläsning

- jQuery, varför?
- jQuery, syntax
- Lär känna din HTML-kod
- DOM – Nyckeln till interaktiva webbplatser
- Case, förstå sin HTML-kod
  - Metoder för att lösa problem
- Ajax, vad är det?
- Ajax, syntax
- Case, ajax



# Frameworks & Libraries

# Ramverk?



Bibliotek

dōjō  
toolkit



mootools



## CODING

CSS

HTML

JavaScript

Techniques

## DESIGN

Web Design

Responsive

Typography

Inspiration

## MOBILE

iPhone & iPad

Android

Design Patterns

## GRAPHICS

Photoshop

Fireworks

Wallpapers

Freebies

## UX DESIGN

# Not An Imposter: Fighting Front-End Fatigue

By [David Berner](#)

🕒 November 17th, 2016

📁 Techniques, Workflow

💬 63 Comments

I recently spoke with a back-end developer friend about **how many hours I spend coding or learning about code outside of work**. He showed me a passage from an Uncle Bob book, “Clean Code”, which compares the hours musicians spend with their instruments in preparation for a concert to developers rehearsing code to perform at work.

I like the analogy but I'm not sure I fully subscribe to it; it's that type of thinking that can cause burnout in the first place. I think it's great if you want to further your craft and broaden your skill set, but to be doing it every hour of the day isn't sustainable.

**Front-end fatigue is very real.** I've seen a number of posts on JavaScript fatigue but I think the problem extends further than that specific language.

<https://www.smashingmagazine.com/2016/11/not-an-imposter-fighting-front-end-fatigue/>

— I love that technology is evolving so rapidly. Equally, I can appreciate how it can be

Advertisement



Search on Smashing Maga

e.g. JavaScript

Advertisement



PSD2HTML.com



Subscribe to our email newsletter and valuable resources, sent every Tuesday.

email address

234,898

Subscribers  
powered by MailChimp

Advertisement



Smashing Magazine

Smashing Jobs



# JOB ADVERTS EXPLAINED

JOB TITLE: Senior Full-stack Agile architect + ninja dev

JOB DESCRIPTION:

We're looking for a maker. Someone to shake up the industry and innovate in the digital space.

REQUIREMENTS:

JS (ES6, ES7), React, Flux, Angular 1/2, nodeJS, SASS, OOCSS, BEM, PostCSS, Agile, PHP, .NET, C#, Ruby, SVG, SEO, canvas, WebGL, animation, photoshop, WebAssembly, 3D, VR, AR ++

WHAT IT SAYS

JOB TITLE: Web developer

JOB DESCRIPTION:

We're looking for someone semi-normal to get us out of the mess we're in.

REQUIREMENTS

fundamental knowledge.  
Ability to learn.

WHAT IT MEANS





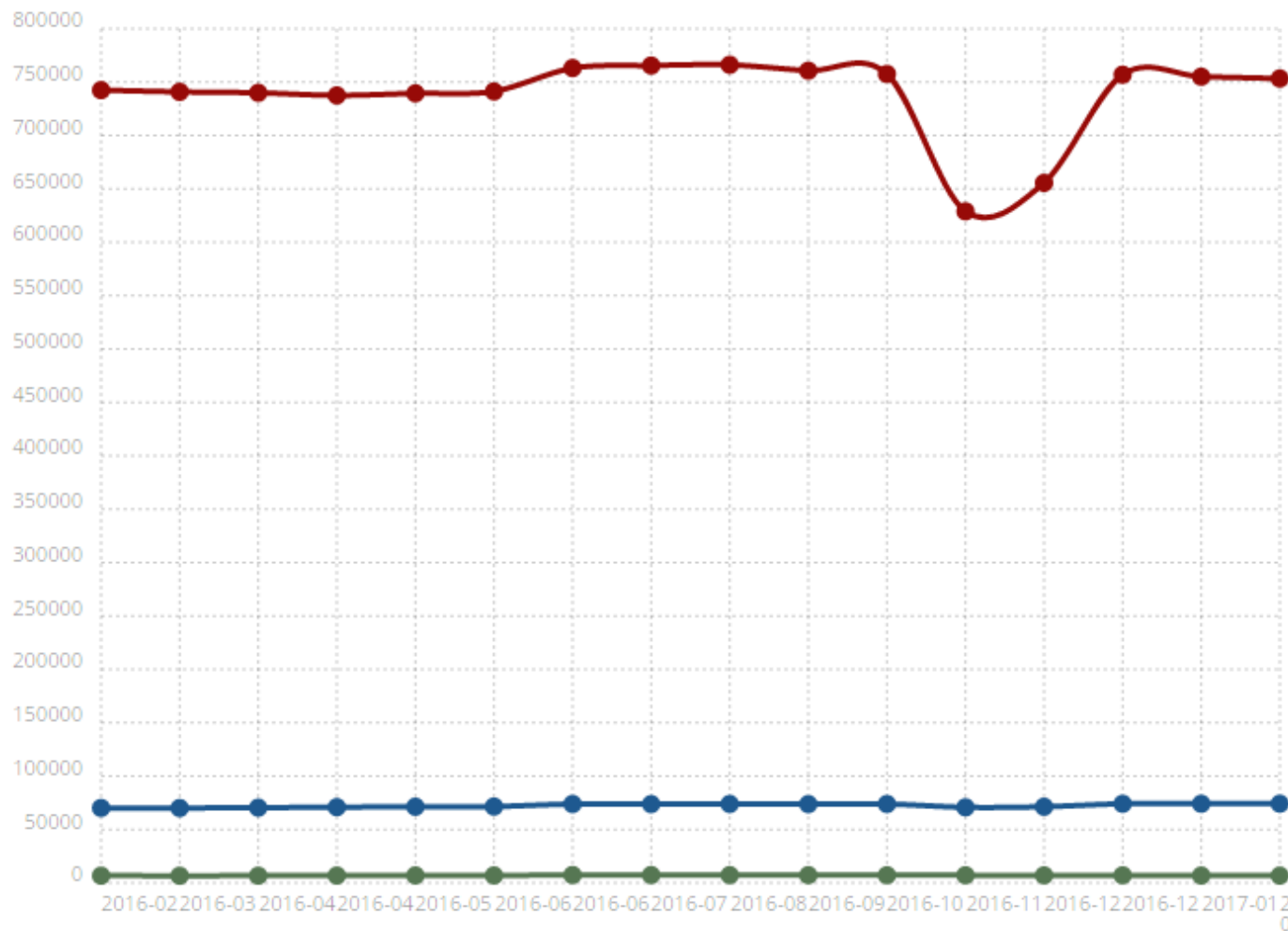
***jQuery***

Erfarenheter hittills utav  
jQuery?

# jQuery Usage Statistics

Websites using jQuery

↓ [Download Lead List](#)



## Chart Data

Source	Legend	Chart
Top 10k	●	<input checked="" type="checkbox"/>
Top 100k	●	<input checked="" type="checkbox"/>
Top 1m	●	<input checked="" type="checkbox"/>
Internet	●	<input type="checkbox"/>

## Coverage Totals

<b>Quantcast Top 10k</b> 6,836 of 10,000	<b>68.4%</b>
<b>Quantcast Top 100k</b> 74,214 of 100,000	<b>74.2%</b>
<b>BuiltWith Top Million</b> 753,322 of 942,003	<b>80%</b>
<b>Entire Internet</b> 67,146,475 of 365,373,952	<b>18.4%</b>

<http://trends.builtwith.com/javascript/jQuery>



1. jQuery erbjuder ”enklare” kod



2. jQuery erbjuder många bra funktioner

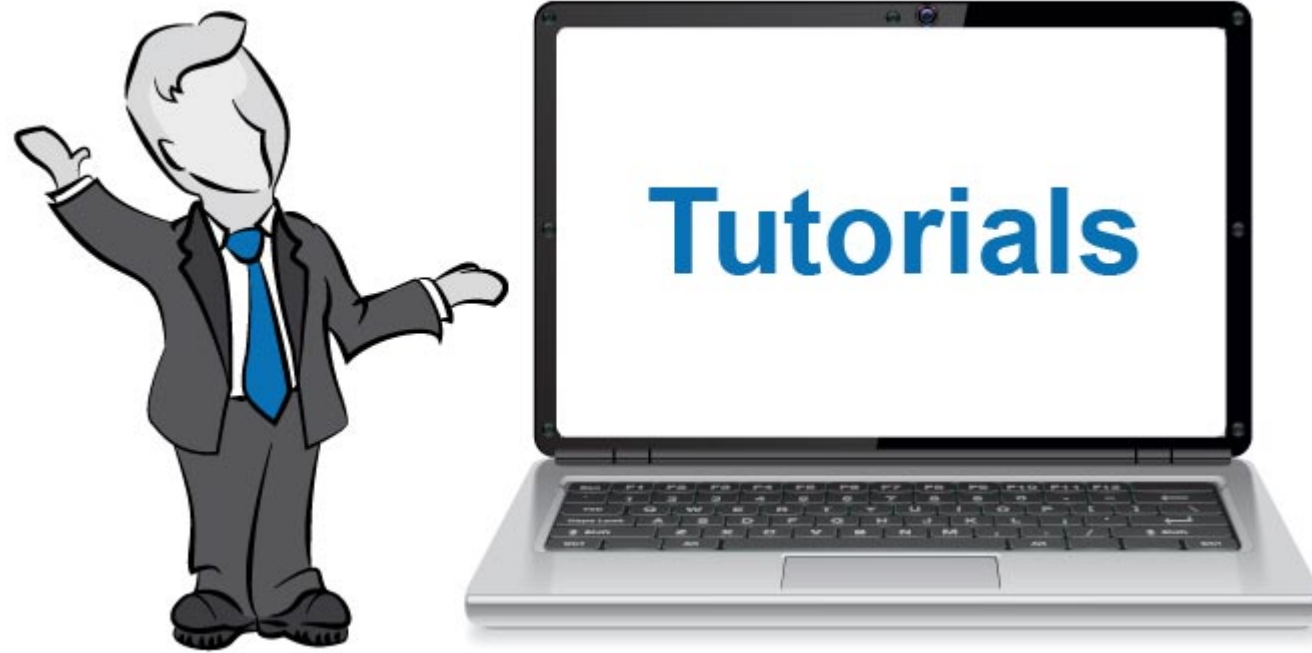


3. jQuery är cross-platform





+ Många plugins



+ Många guider

# Exempel på guider

- <https://www.codecademy.com/learn/learn-jquery>
- <https://learn.jquery.com/>
- <http://www.w3schools.com/jquery/>
- <https://www.codeschool.com/courses/try-jquery>
- <http://jqfundamentals.com/chapter/jquery-basics>



<http://api.jquery.com/>

# \$ = jQuery

\$ är en referens till jQuery-objektet

# jQuery

- Vi kallar på jQuery när vi vill hitta element på vår webbsida.

## jQuery()

Return a collection of matched elements either found in the DOM based on passed argument(s) or created by passing an HTML string.

```
// Letar upp alla länkar i vårt dokument
var links = $("a");
// Letar upp alla element med klassen .blue i vårt dokument
var classBlut = $(".blue");
// Letar upp elementet med id start
var start = $("#start");
```

```
// Letar upp alla länkar i vårt dokument
var links = jQuery("a");
// Letar upp alla element med klassen .blue i vårt dokument
var classBlut = jQuery(".blue");
// Letar upp elementet med id start
var start = jQuery("#start");
```



# Jämför med vanligt JavaScript

```
// Letar upp alla länkar i vårt dokument
var links = $("a");
var links = document.querySelectorAll("a");

// Letar upp alla element med klassen .blue i vårt dokument
var classBlue = $(".blue");
var classBlue = document.querySelectorAll(".blue");

// Letar upp elementet med id start
var start = $("#start");
var start = document.querySelector("#start");
```

# jQuery-funktioner på element

```
// Letar upp elementet med id start
var start = $("#start");
// Gömmer start
start.hide();
// Visar start
start.show();
// Läger till klassen "center" på start
start.addClass("center");
// Tar bort start
start.remove();
```

# jQuery – Mer än bara hitta element

- jQuery har fler väldigt nyttiga funktioner vi kan använda, t.ex.

```
// Skapar ett ajax-anrop  
$.ajax();  
// Hämtar information från en URL  
$.get();  
// Loopar igenom en samling element  
$.each();
```

# jQuery != JavaScript

- Vi kan inte använda "vanliga" JavaScript-funktioner på jQuery-objekt, och vice versa.

```
// Inte korrekt
$("#start").setAttribute("class", "center");
document.querySelector("#start").attr("class", "center");

// Korrekt
document.querySelector("#start").setAttribute("class", "center");
$("#start").attr("class", "center");
```

# jQuery - Exempel

- 1) Testa hur det fungerar
- 2) Att göra-lista! (igen...)



Lär känna din HTML-kod



Vad är syftet med  
HTML?

Varför har vi element  
i HTML?

Varför har vi attribut  
i HTML?

*Anatomy of an HTML element*

The diagram illustrates the structure of an HTML element using the example `<p class="nice">Hello world!</p>`. Brackets are used to group parts of the code with labels:

- Opening tag**: A bracket above the code spans from `<p` to `>`.
- Closing tag**: A bracket above the code spans from `</p>` to `>`.
- An attribute and its value**: A bracket below the code spans from `class="nice"` to `"nice"`.
- Enclosed text content**: A bracket below the code spans from `Hello world!` to `!`.

`<p class="nice">Hello world!</p>`

# Vad finns det för attribut?

- Många! T.ex.
  - id
  - class
  - title
  - rel
  - src
  - href
  - type
  - ... och många fler.
- Beroende vilken typ av element det är - finns det olika attribut som är användbara.



# jQuery - Exempel

Att göra flikar!



# HTML 5 – Attributet *data-\**

- Ibland så räcker inte de attributen som vi har att tillgå till. Tänk om jag har behov som dessa inte motsvarar?
- Attributet *data-\** ger oss möjlighet att skapa egna attribut där \* är ett valfritt namn, t.ex. skulle vi kunna skapa följande attribut för element:

```
<ul>  
  <li data-animal-type="bird">Owl</li>  
  <li data-animal-type="fish">Salmon</li>  
  <li data-animal-type="spider">Tarantula</li>  
</ul>
```

# jQuery – Hämta attribut

```
// Hämtar värdet för attributet "data-id"
$("#my-element").attr("data-id");

// Sätter värdet för attributet "data-id"
$("#my-element").attr("data-id", "new value");

// Hämtar värdet från attributet "value" (formulärselement)
$("#my-element").val();

// Sätter värdet från attributet "value" (formulärselement)
$("#my-element").val("new value");
```



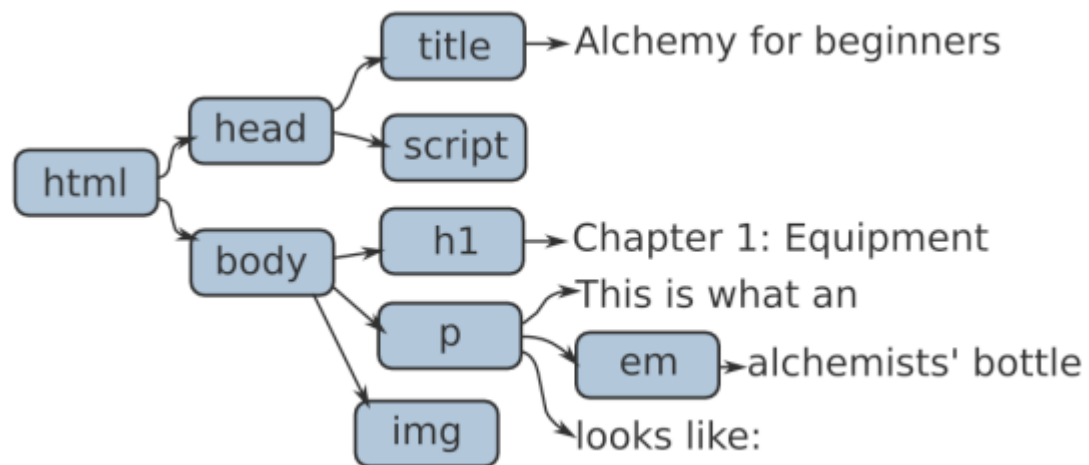
# Demo – *data*-\*

Filtrera data!

# DOM

Behöver vi ha koll på nu!

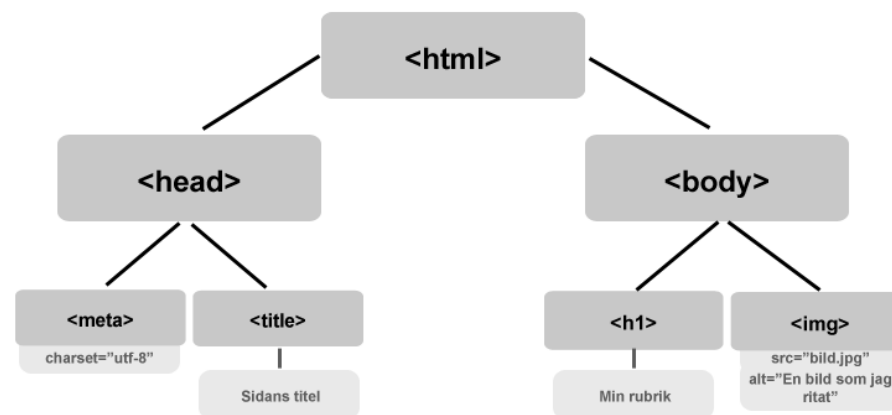
# DOM



- Trädstruktur som motsvarar taggarnas ordning/nästling
- Varje element är en "nod" i trädet
- Relationer mellan noderna beskrivs med förälder/barn och syskon

# Källkod vs. noder

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Sidans titel</title>
  </head>
  <body>
    <h1>Min rubrik</h1>
    
  </body>
</html>
```



# jQuery & DOM

- jQuery har många bra funktioner för att navigera i DOM
  - Som dessutom fungerar cross-browser!
- <https://api.jquery.com/category/traversing/tree-traversal/>

```
// Nästa element
$("#my-element").next();
// Föregående element
$("#my-element").prev();
// Förälderelement
$("#my-element").parent();
// Elementets barn
$("#my-element").children();
// Letar upp ett element (bland barnen)
$("#my-element").find();
```

# jQuery – lägga till innehåll

```
// Lägger till innehåll som första barn till elementet
$("#my-element").prepend("<p>En ny paragraf</p>");
// Lägger till innehåll som sista barn till elementet
$("#my-element").append("<p>En ny paragraf</p>");
// Lägger till innehåll före elementet
$("#my-element").before("<p>En ny paragraf</p>");
// Lägger till innehåll efter elementet
$("#my-element").after("<p>En ny paragraf</p>");
```





# Navigera i DOM

# Bygga modals!

Typ, snygga popup på bilder.



JavaScript

# Ajax

*Asynchronous JavaScript and XML*

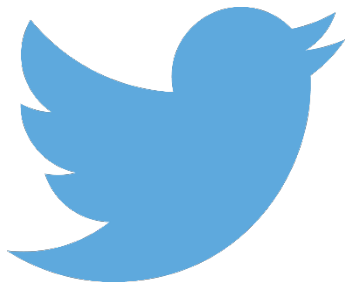
# Agenda - ajax

- Vad innebär ajax?
- Hur använder man ajax?
- Exempel på ajax
- Att bygga en enkel tjänst med ajax



*”**AJAX** (en. Asynchronous JavaScript and XML) är ett samlingsnamn för flera olika tekniker som kan användas för att bygga applikationer för World Wide Web med bättre interaktivitet än tidigare webbapplikationer.”*

Aggregators Wikis Folksonomy User Centered Joy of Use  
Blogs Participation Six Degrees Usability Widgets  
Pagerank XFN Recommendation Social Software FOAF Browser  
Videocasting Podcasting Sharing Collaboration Perpetual Beta Simplicity AJAX  
Audio IM Video Convergence Web 2.0 Design CSS Pay Per Click  
UMTS Mobility Atom XHTML SVG Ruby on Rails VC Trust Affiliation  
OpenAPIs RSS Semantic Web Standards SEO Economy  
OpenID Remixability REST Standardization The Long Tail  
DataDriven Accessibility XML  
Modularity SOAP Microformats Syndication





# Exempel: Google

<https://www.google.se>



# ”...för flera olika tekniker”

- XMLHttpRequest-objektet, som tillåter JavaScript på en webbsida att göra anrop till en webserver utan att sidan laddas om.
  - Detta kan var både inom den egna webbplatsen, eller till externa webbplatser
- DOM (Document Object Model) vilket tillåter JavaScript att skriva om innehåll på den aktuella sidan.
- HTML och CSS vilket beskriver utseende för, och märker, innehållet på sidan.
- XML, JSON, m.m. som används för att formatera data som transporteras mellan klient och server. Vanlig text kan också användas som format.



**Problem:** hämta/skicka information utanför vår html-sida

Lösning: Ajax (oavsett om det är information från vår egen webbplats, eller från annan webbplats)



*”Att interaktivt ändra innehållet på vår webbplats”*

Från olika data-källor

# Exempelområden

- Autocomplete (<http://google.com>)
  - Vid sökningar av diverse information
- Flerstegsformulär (<http://demo.tutorialzine.com/2011/11/chained-ajax-selects-jquery/>)
- ”Realtidssystem”
  - T.ex. chattsystem, auto-uppdateringar, etc.
- Direktvalidering av formulär
- Autosparning av information
- Röstningssystem
- Etc.

# Fördelar med Ajax

- Snabbar upp användandet av webbplatser
  - Ladda bara om den information vi behöver – inte allt
- Färre siduppdateringar på webbplatsen
- Reducerar nätverkstrafik
- Minskar serverbelastning
- Ger ett mer interaktivt GUI

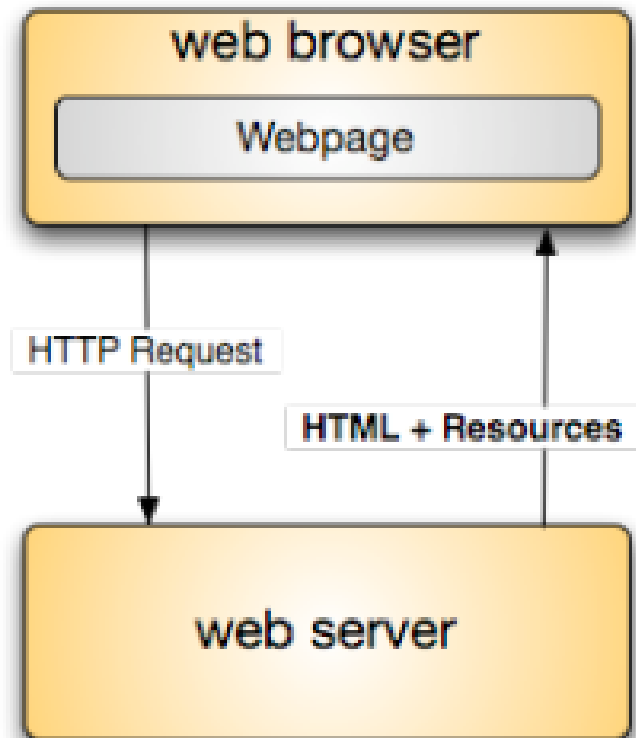


# Nackdelar med Ajax

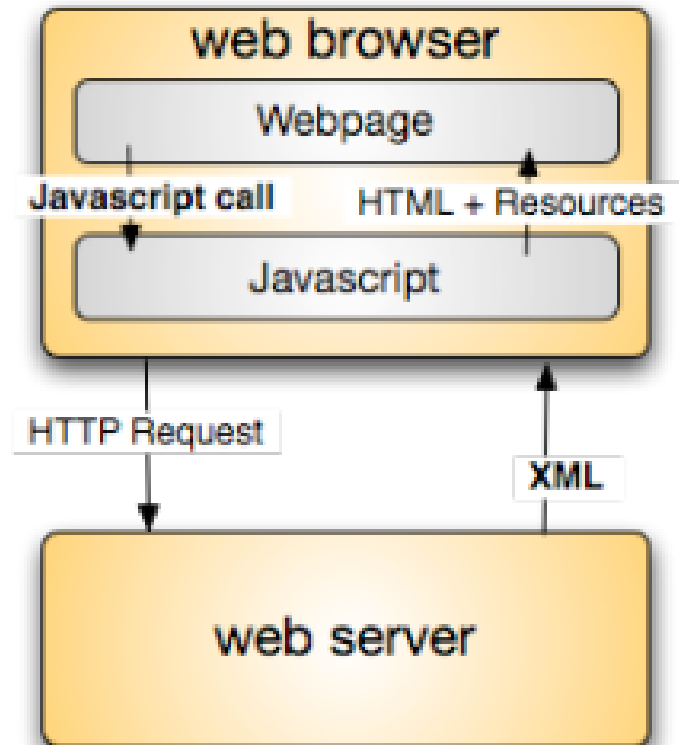
- Bokmärken slutar fungera
- Sökmotorer indexerar inte innehållet som laddas genom ajax
- "Bakåtknappen" slutar att fungera
- JavaScript kan enkelt inaktiveras från klientsidan



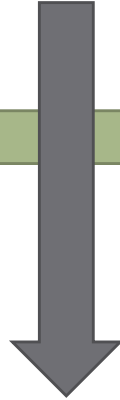
## Traditional web model



## AJAX web model



moviefinder.com/index.html



moviefinder.com/result.html

Klassisk webbsida

## Moviefinder - King of movies!

Sök efter film

Titel:

Titel här... (minst 3 tecken)

Sök film!

Sidomladdning

## Moviefinder - King of movies!

Sök efter film

Titel:

Star Wars

Sök film!

### Sökresultat



Star Wars: Episode IV - A New Hope

1977



Star Wars: Episode V - The Empire Strikes Back

1980



Star Wars: Episode VI - Return of the Jedi

1983



Star Wars: Episode I - The Phantom Menace

1999



moviefinder.com/index.html

## Moviefinder - King of movies!

Sök efter film

Titel:

Titel här... (minst 3 tecken)

Sök film!



## Moviefinder - King of movies!

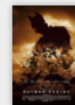
Sök efter film

Titel:

Batman

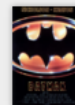
Sök film!

### Sökresultat



Batman Begins

2005



Batman

1989



Batman Returns

1992

Interaktiv webbsida m.h.a. Ajax

## Category: Shorthand Methods

These methods perform the more common types of Ajax requests in less code.

### **jQuery.get()**

Load data from the server using a HTTP GET request.

### **jQuerygetJSON()**

Load JSON-encoded data from the server using a GET HTTP request.

### **jQuery.getScript()**

Load a JavaScript file from the server using a GET HTTP request, then execute it.

### **jQuery.post()**

Load data from the server using a HTTP POST request.

### **.load()**

Load data from the server and place the returned HTML into the matched element.

jQuerys Ajax-metoder

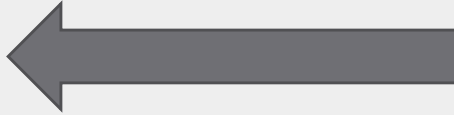
## Category: Low-Level Interface

---

These methods can be used to make arbitrary Ajax requests.

### **jQuery.ajax()**

Perform an asynchronous HTTP (Ajax) request.



### **jQuery.ajaxPrefilter()**

Handle custom Ajax options or modify existing options before each request is sent and before they are processed by \$.ajax().

### **jQuery.ajaxSetup()**

Set default values for future Ajax requests. Its use is not recommended.

### **jQuery.ajaxTransport()**

Creates an object that handles the actual transmission of Ajax data.

jQuerys Ajax-metoder

# Exempel på ett ajax-anrop

"data" är ett objekt  
med svaret som vi  
laddat in

"data" är ett objekt  
med ev. fel som  
förekommit

```
1  
2  
3 $.ajax({  
4     url: "/"  
5 }) .done(function(data) {  
6     console.log("Ajax-anrop klart, lyckades!");  
7 }) .fail(function(data) {  
8     console.log("Ajax-anrop klart, misslyckades!");  
9 });  
10  
11  
12
```

# Asynkront

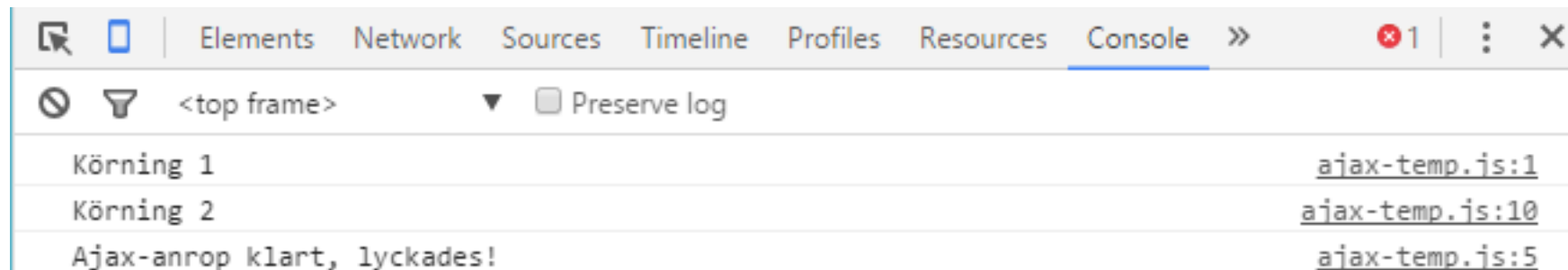
Bra att veta!

# Körning av ajax-anropet

```
1 console.log("Körning 1");
2
3 $.ajax({
4     url: "/"
5 }).done(function(data) {
6     console.log("Ajax-anrop klart, lyckades!");
7 }).fail(function(data) {
8     console.log("Ajax-anrop klart, misslyckades!");
9 });
10
11 console.log("Körning 2");
12
```

# Körning av ajax-anropet

```
1 console.log("Körning 1");
2
3 $.ajax({
4     url: "/"
5 }).done(function(data) {
6     console.log("Ajax-anrop klart, lyckades!");
7 }).fail(function(data) {
8     console.log("Ajax-anrop klart, misslyckades!");
9 });
10
11 console.log("Körning 2");
12
```



# Varför asynkront?

- Vi vill inte behöva vänta på slöa servrar! (eller andra saker som kan hindra får sidan från att vara snabb)





# Callback-funktioner

- Istället för att göra saker i den ordning som det står (synkront), vill vi istället göra saker när vårt ajax-anrop får ett svar (asynkront).

```
1 console.log("Körning 1");
2
3 $.ajax({
4     url: "/"
5 }).done(function(data) {
6     console.log("Ajax-anrop klart, lyckades!");
7 }).fail(function(data) {
8     console.log("Ajax-anrop klart, misslyckades!");
9 });
10
11 console.log("Körning 2");
12
```

# Utan anonyma funktioner

---

```
console.log("Körning 1");

$.ajax({
  url: "/"
}).done(onSuccess).fail(onFail);

function onSuccess(data){
  console.log("Ajax-anrop klart, lyckades!");
}

function onFail(data){
  console.log("Ajax-anrop klart, misslyckades!");
}

console.log("Körning 2");
```

# Ett standard ajax-anrop

- Med jQuery.

```
console.log("Körning 1");
$.ajax({
  url: "/", // Adressen för anropet
  type: "GET", // Vilken HTTP-metod vi ska använda
  dataType: "JSON", // Hur vi ska tolka vårt svar
  data: { // Eventuella parametrar vi vill skicka med
    arg1: "value1",
    arg2: "value2",
    arg2: "value3"
  }
}).done(function(data) {
  console.log("Ajax-anrop klart, lyckades!");
}).fail(function(data) {
  console.log("Ajax-anrop klart, misslyckades!");
});

console.log("Körning 2");
```

# API – Ett trevligt sätt att hämta data på.

- Vi vill använda redan färdig data i våra applikationer, t.ex.
- Musik (t.ex. <https://developer.spotify.com/>)
- Film (t.ex. <https://www.omdbapi.com/>)
- Chuck Norris jokes (<http://www.icndb.com/api/>)
- ... och massvis andra data källor.
- Vi kan även skapa egna data-källor vilket vi ska börja med! =)

Exempel:  
Raggningsrepliker

# Raggningshjälpen!

---

Ge mig ett tips!

---

# Raggningshjälpen!

---

Ge mig ett tips!

---

*Tjena kexet, sitter du här och smular?*

# Förutsättningar

- Vi har en sida som ger oss slumpvalda ragningsrepliker i ren text
  - <http://localhost:1234/date.php>
- Vi vill nu varje gång vi klickar på knappen ”Ge mig ett tips”, genom JavaScript (och jQuery) ladda in denna sidas innehåll – och visa upp det

## Ragningshjälpen!

Ge mig ett tips!

*Tjena kexet, sitter du här och smular?*



Låt oss testa!



# Använda externa API

<http://www.icndb.com/api/>

# Chuck Norris Jokes!

---

Give me a joke!



# Chuck Norris Jokes!

---

Give me a joke!

*Chuck Norris got his drivers license at the age of 16. Seconds.*

# JSON

JavaScript Object Notation

```
var jsonData = {  
  "employees": [  
    {"firstName": "John", "lastName": "Doe"},  
    {"firstName": "Anna", "lastName": "Smith"},  
    {"firstName": "Peter", "lastName": "Jones"}  
  ]  
};
```

```
console.log(jsonData);  
console.log(jsonData.employees);  
console.log(jsonData.employees[0]);  
console.log(jsonData.employees[0].firstName);  
console.log(jsonData.employees[0].lastName);
```

---

► Object {employees: Array[3]}

---

► [Object, Object, Object]

---

Object {firstName: "John", lastName: "Doe"}

---

John

---

Doe

<http://api.icndb.com/jokes/random>

```
var data = {  
  "type": "success",  
  "value": {  
    "id": 243,  
    "joke": "Contrary to popular belief, the Titanic didn't  
hit an iceberg. The ship was off course and ran into  
Chuck Norris while he was doing the backstroke across  
the Atlantic.",  
    "categories": []  
  }  
}
```

