JS-biblioteket jQuery

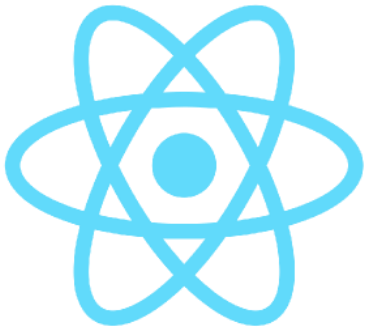
Dagens föreläsing

- 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


WORK
SMARTER
30%
70%
BUSINESS PLAN
time
money
quality
USER
CLOUD
cost
quality
price
EXPENSES
TAX
PROFIT
INCOME
20%
30%

Framworks & Libraries

Ramverk?



React

ANGULARJS

by Google

Vue.js

Bibliotek

DING
S
ML
aScript
hniques

SIGN
o Design
ponsive
ography
piration

OBILE
one & iPad
droid
sign Patterns

APHICS
toshop
works
lpapers
ebies

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.

To be clear, this isn't another rant about how it's all bad and everything is moving too fast — I love that technology is evolving so rapidly. Equally, I can appreciate how it can be

123 (1).jpg | 123.jpg | 123 | accessKeys.csv | Invoice_85682496 ....pdf | Invoice_89327338.pdf

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

jQuery

Erfarenheter hittills utav jQuery?

jQuery Usage Statistics

Websites using jQuery

Download Lead List

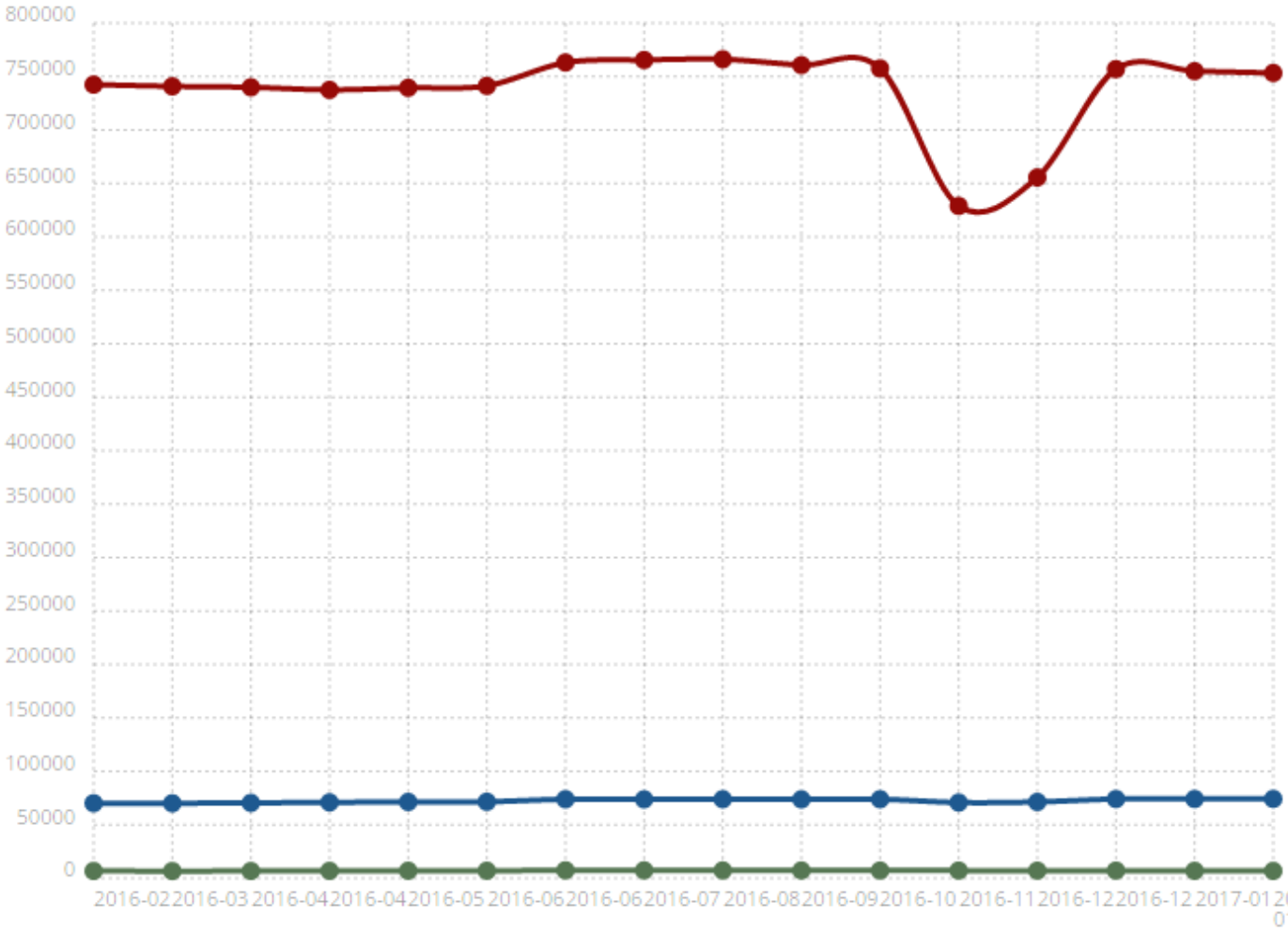


Chart Data

Source	Legend	Chart
Top 10k		☑
Top 100k		☑
Top 1m		☑
Internet		☐

Coverage Totals

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

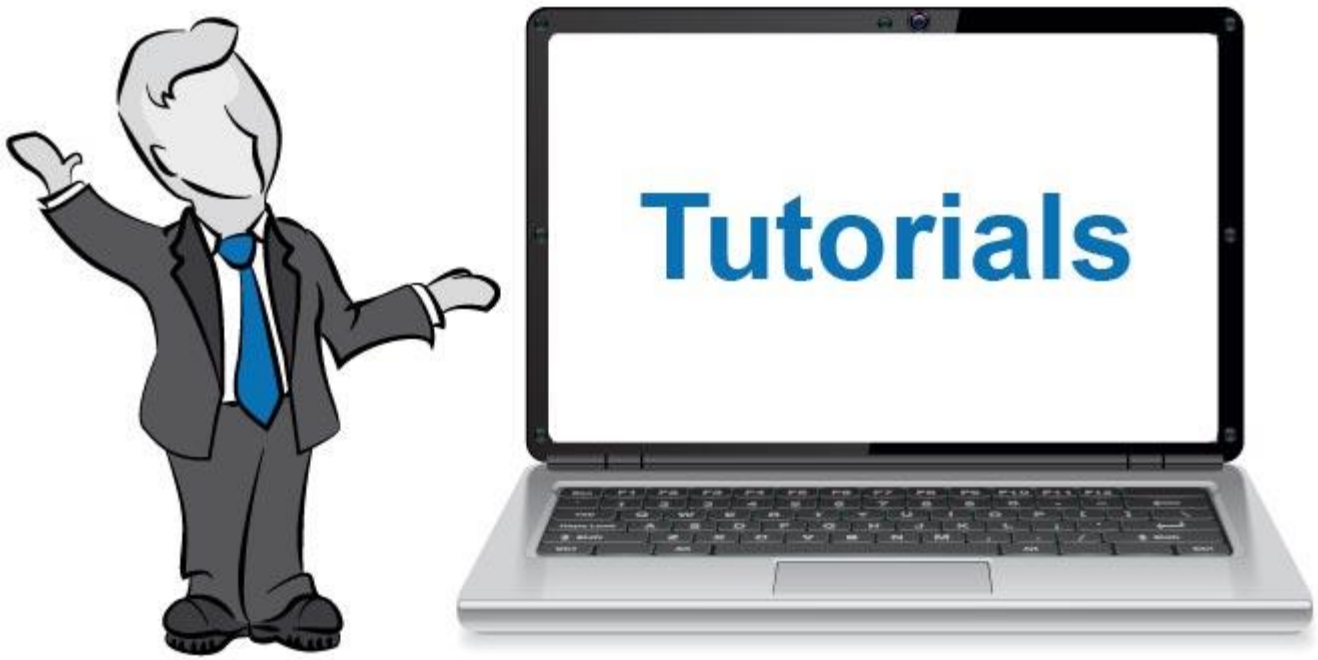
http://trends.builtwith.com/javascript/jQuery

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/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();
// Lägger till klassen "center" på start
start.addClass("start");
// 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

Att göra!

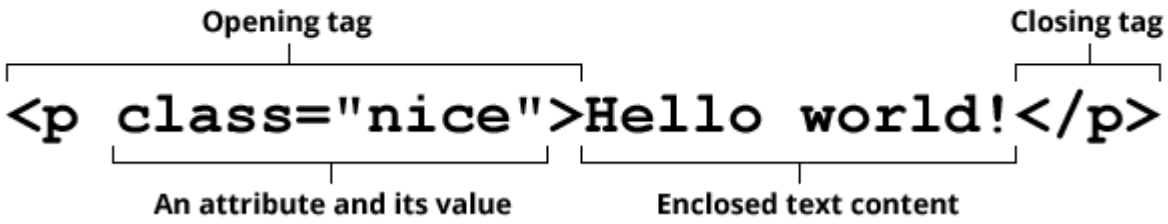
Lär känna din HTML-kod

Vad är syftet med HTML?

Varför har vi element i HTML?

Varför har vi attribut i HTML?

Anatomy of an HTML element



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.

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

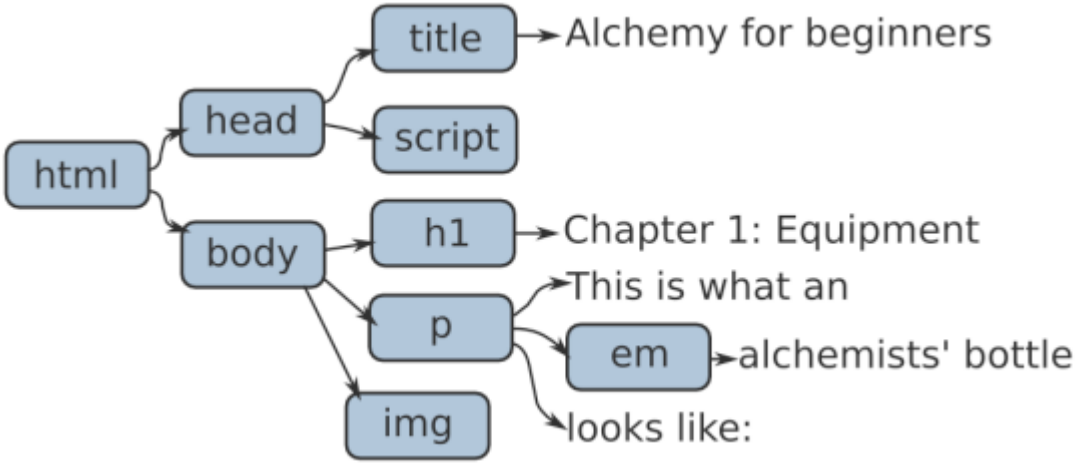
```
// Hämtar värdet för attributet "data-id"
$("#my-element").attr("data-id");
// Sätter värdet för attributet "data-id"
$("#my-element").attr("data-id", "new value");

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

Demo – *data-**

DOM

Behöver vi ha koll på nu!

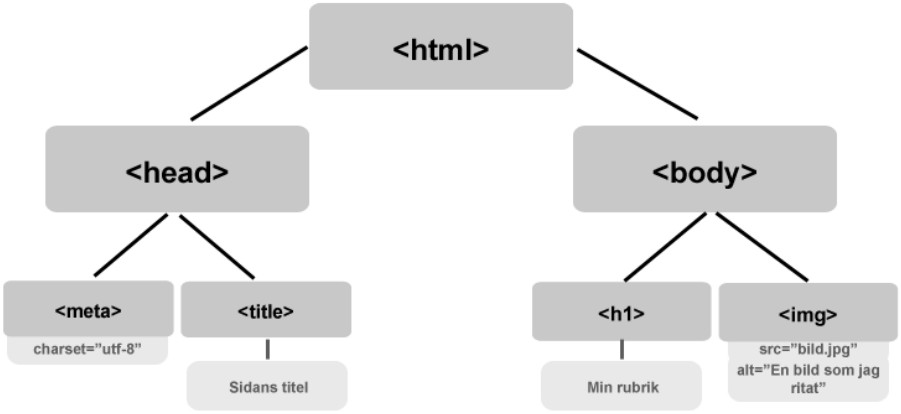
DOM



- 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

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

    <body>
        <h1>Min rubrik</h1>
        <img src="bild.jpg" alt="En bild som jag ritat">
    </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>)");
```

Demo - Navigera i DOM

Ajax

Asynchronous JavaScript and XML

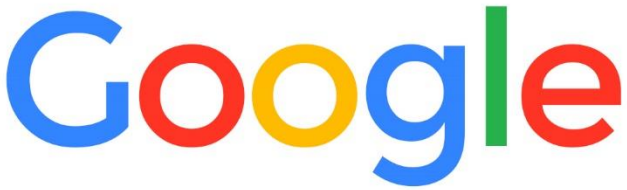
Dagens agenda

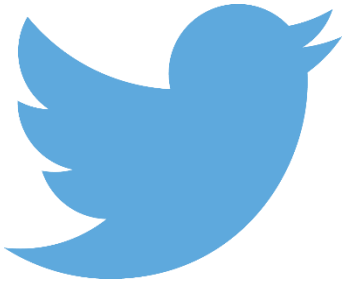
- Vad innebär ajax?
- Hur använder man ajax?
- Exempel på ajax
- Att bygga en enkel tjänst med ajax

*”**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.”*


Web 2.0
Aggregators
Folksonomy
Wikis
Blogs
User Centered
Joy of Use
Participation
Six Degrees
Usability
Pagerank
XFN
Widgets
Recommendation
Social Software
FOAF
Browser
Simplicity
Sharing
Collaboration
Perpetual Beta
Videocasting
Podcasting
AJAX
Audio
IM
Video
Design
Convergence
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
Microformats
Syndication
Modularity
SOAP


AJAX
Asynchronous Javascript And XML


Google




YouTube

Exempel: Google

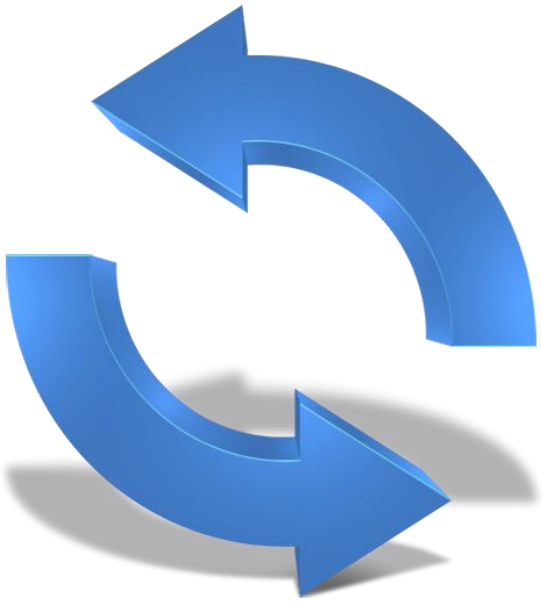
https://www.google.se

”…för flera olika tekniker”

- XMLHttpRequest-objektet, som tillåter JavaScript på en webbsida att göra anrop till en webbserver 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.

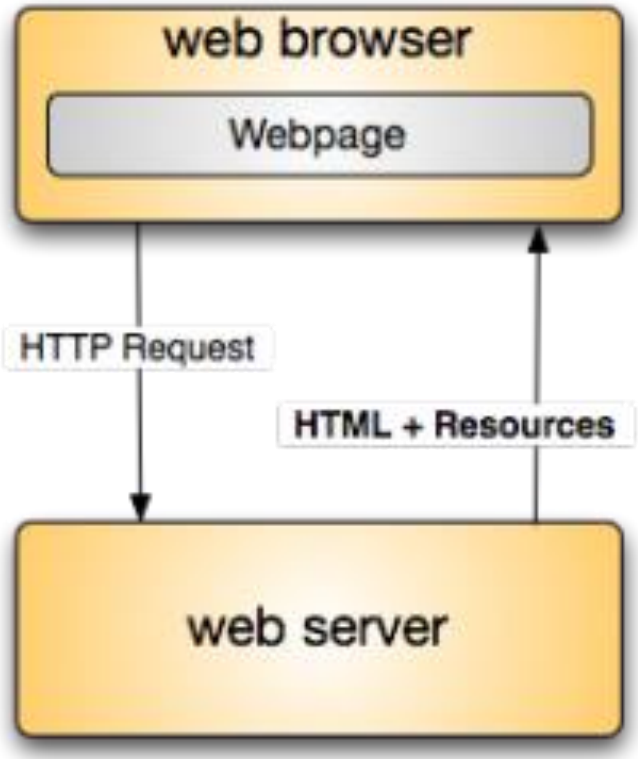
Fördelar med Ajax

- 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

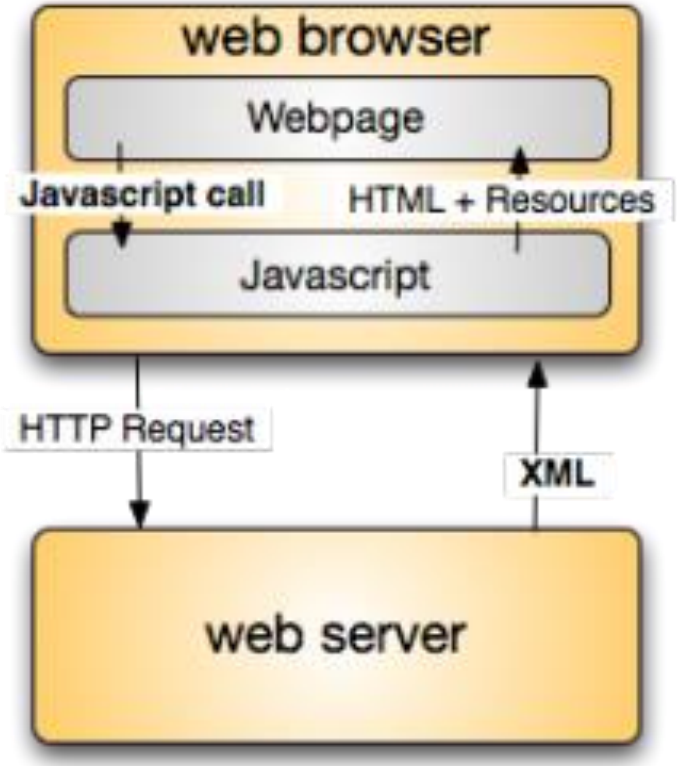
Nackdelar med Ajax

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

Traditional web model



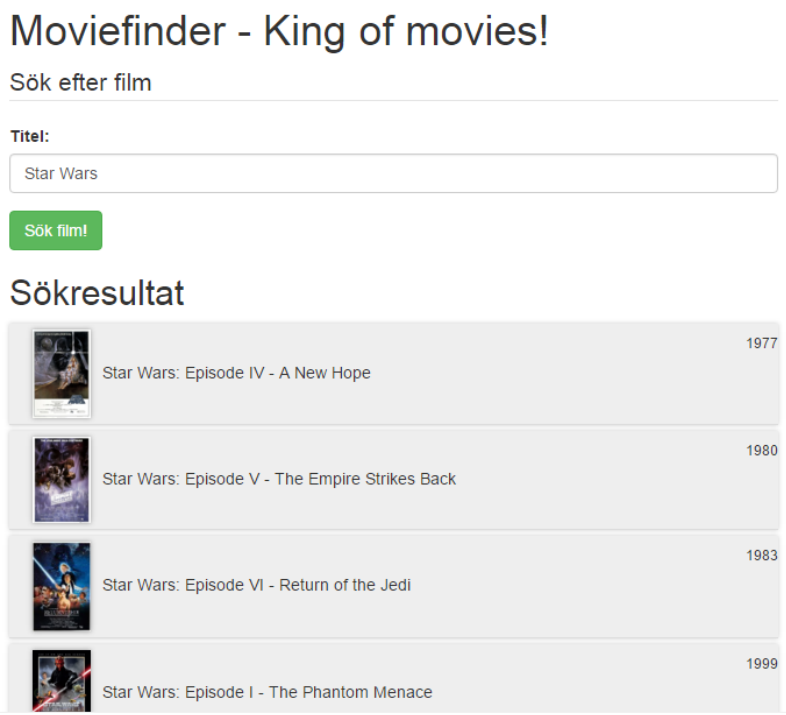
AJAX web model



moviefinder.com/index.html

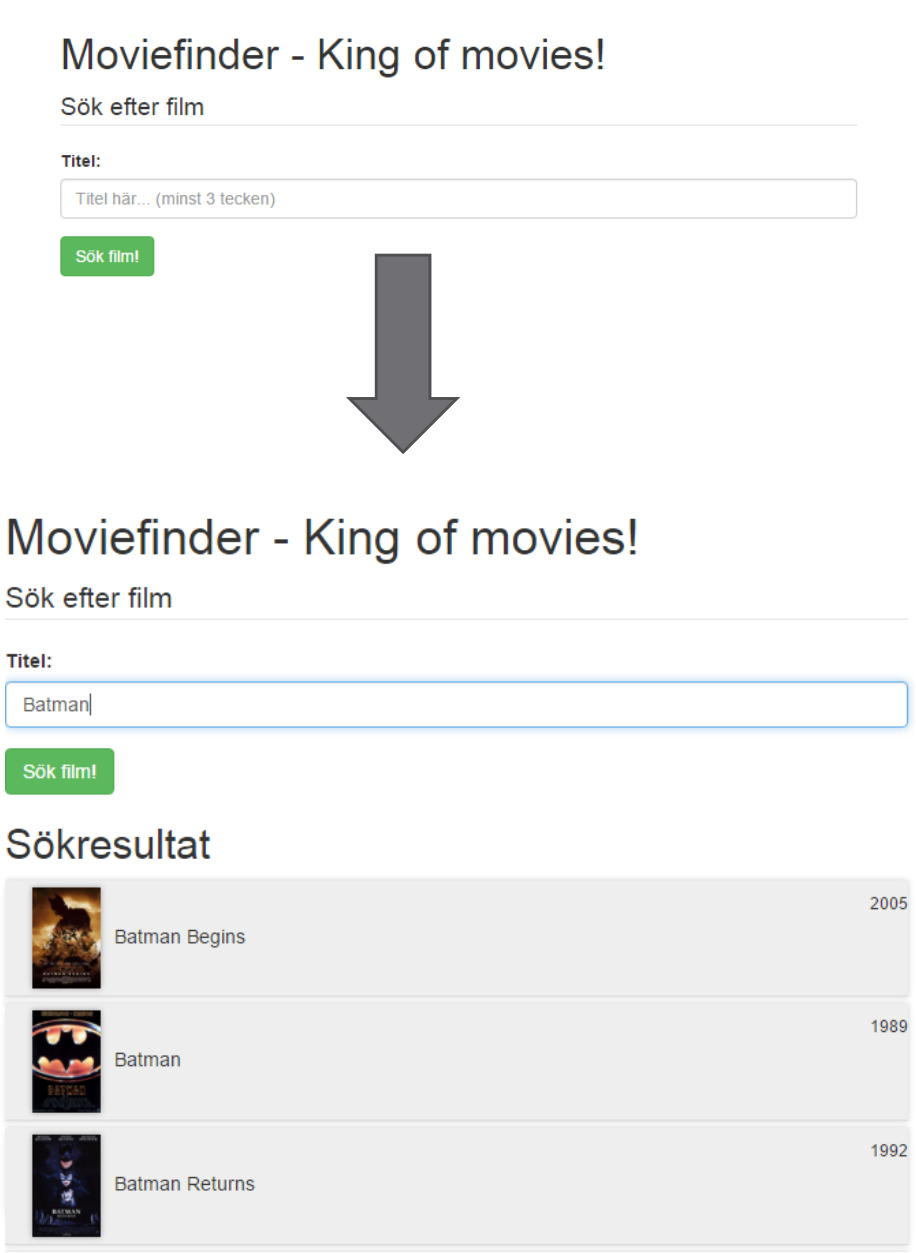
Sidomladdning

moviefinder.com/result.html



Klassisk webbsida

moviefinder.com/index.html



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.

jQuery.getJSON()

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()

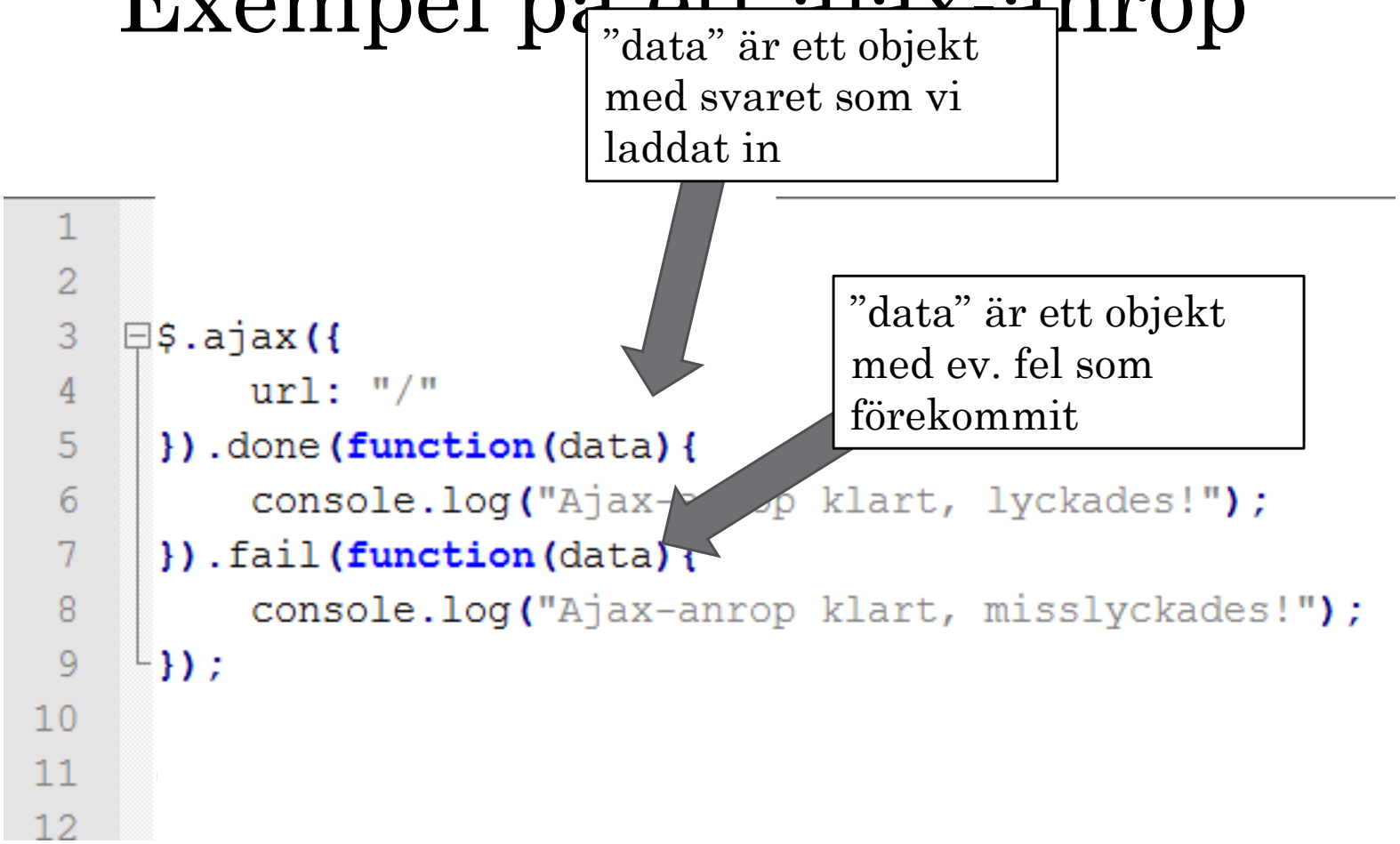
Set default values for future Ajax requests. Its use is not recommended.

jQuery.ajaxTransport()

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

jQuerys Ajax-metoder

Exempel på ett ajax-anrop



Asynkront

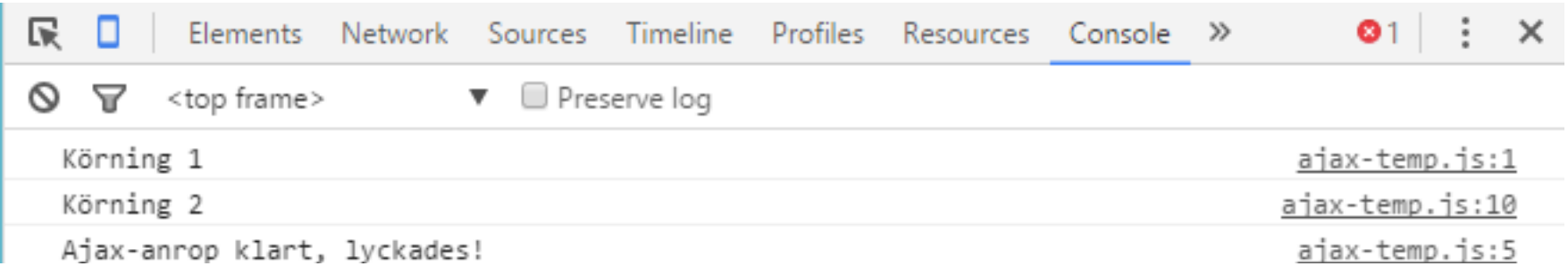
Bra att veta!

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 raggningsrepliker 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

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":[]
    }
}
```