



MVC med Javascript och Ajax

Filip Ekberg

A photograph of a workspace featuring a silver laptop on the left, two smartphones (one black, one silver) on the right, and a tablet in the foreground. The laptop screen shows a website with a 'NEW PHOTOS ADDED' section. A large white text overlay is centered on the image.

MVC med Javascript och Ajax

Lektion 3 & 4 - Javascript & Ajax

Vad ska vi lära oss?

I denna kurs lär sig den studerande hur man jobbar med den naturliga uppdelningen av **data, GUI och logik** i sina projekt.

Detta görs med **MVC** som är ett ramverk. MVC har mer eller mindre blivit en **standard** för avancerade .NET-webbsidor. Arbetssättet gör att koden blir mer **återanvändbar och kostnadseffektiv**.

Man delar upp projektet i de **logiska lagren** i **Modell, View och Controller**. Varje logiskt lager kan ersättas av en ny modul för att skapa en helt ny applikation.

Vad ska vi lära oss?

- Skapa projekt i ramverket ASP.NET MVC
- Projektstruktur i Visual Studio
- Model-View-Controller:
 - Lagerstruktur och concept
 - Designmönster & Arkitektur
- Integrering av databas och Entity Framework
- Lägga till och modifiera: Layout, Controllers, Razor Views
- JavaScript, Ajax
- Säkerhet & Infrastruktur

Bildgalleri?

- Ladda upp bilder
- Skapa album av uppladdade bilder
- Kommentera på bilder
- Inloggning & Säkerhet
- Administrationsgränssnitt

mail@filipekberg.se



Hur har det gått?

Vad är svårast att förstå?

Dagens mål

- Repetition
 - Vad är MVC?
 - Hur hänger det ihop med ASP.NET?
 - Hur bygger jag ett MVC-projekt?
- HTTP
 - Hur funkar egentligen GET, POST, PUT, DELETE?
 - När använder man vilken och hur funkar en webserver?
- Hur skiljer sig server-kod från kod som exekveras på klienten?
- Vad är JavaScript och jQuery – Hur kommer vi igång med klient-programmering?
- Vad är Asynkron Programmering?
- Vad är Ajax?
- GET, POST via Ajax
- Hur hanterar vi svar från servern via GET/POST med JavaScript?
- Hur manipulerar vi DOM?

MVC

Var ska min logik vara?

Undvik logik i vyer!

(Det finns undatag)

“Löst Kopplat”

**Var ska jag hantera var
bilden sparas?**

MVC

Controller
Action
View
View Model

Busines-lager

Integrations
Domain Validation
Domain Model

Data-lager

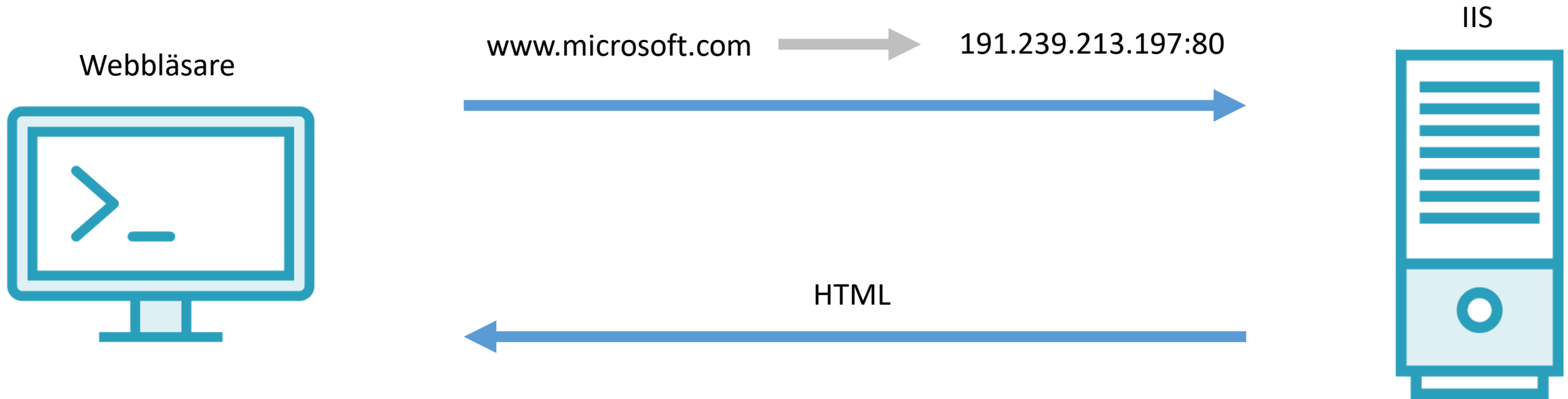
Repositories
Entity Framework
Data Model

**Hur lätt är de för mig
att byta ut ett lager nu?**

HTTP

Hur funkar en webserver?

Hur funkar en webserver?

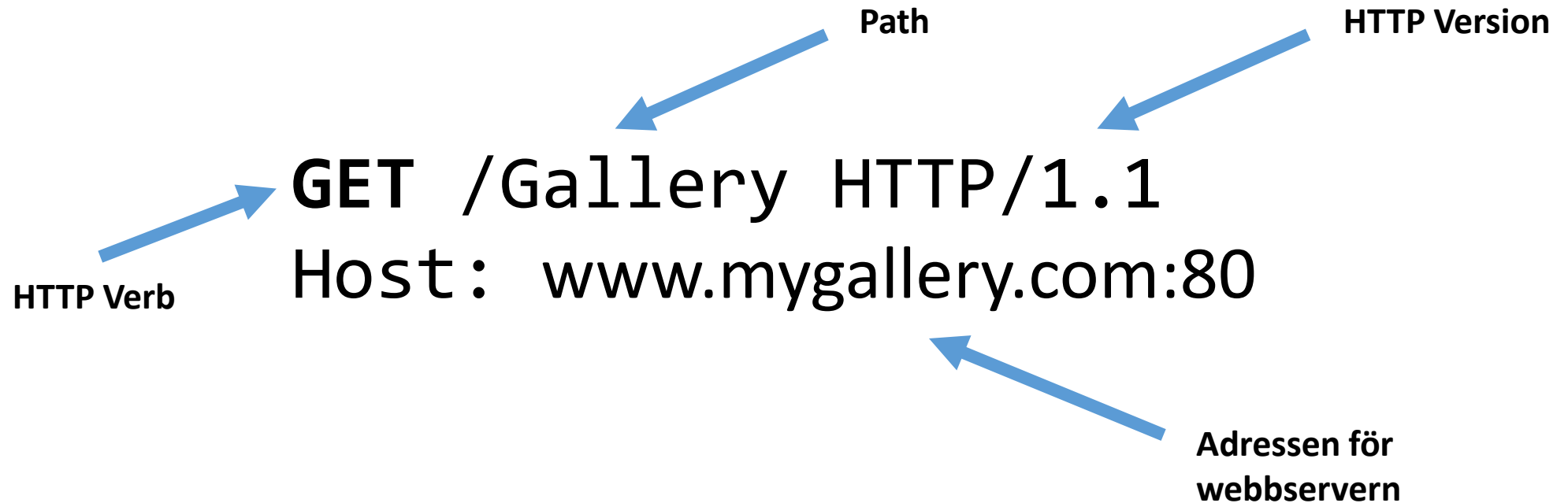


**Hur vet webbservern vad
som ska hända?**

HTTP

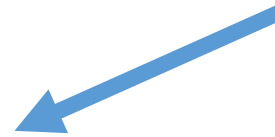
Ett protokoll för hur webben fungerar

www.mygallery.com/Gallery



www.mygallery.com/Gallery?id=1

Query String



GET /Gallery?id=1 HTTP/1.1
Host: www.mygallery.com:80

GET /Gallery HTTP/1.1

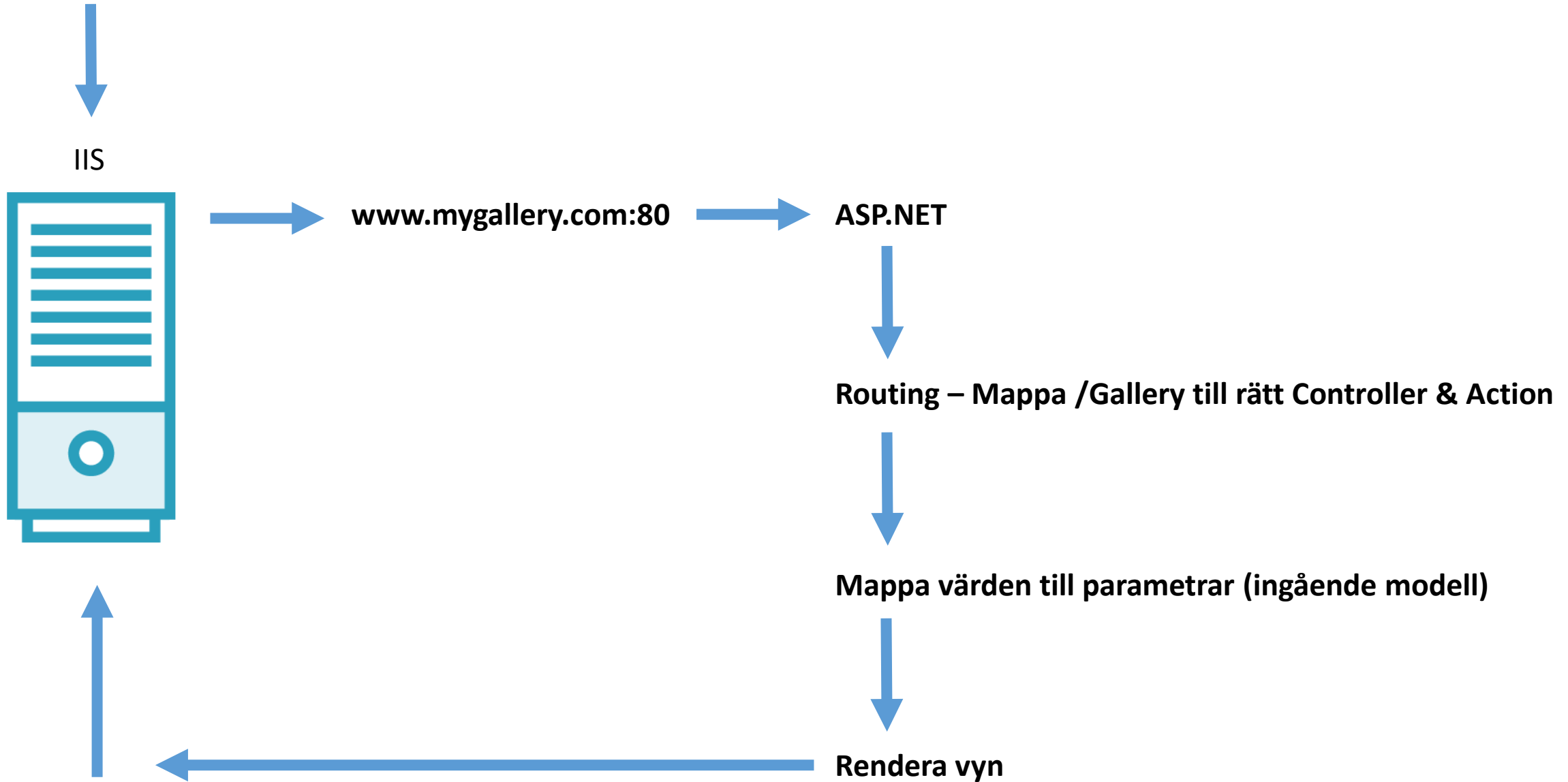
Host: www.mygallery.com



```
<!DOCTYPE html>
<html>
<head>
  <meta name="viewport" content="width=device-width" />
  <title>My Gallery</title>
</head>
<body>
  <div>
    <h2>This is my gallery page</h2>
  </div>
</body>
</html>
```


ASP.NET MVC


GET /Gallery?id=1 HTTP/1.1
Host: www.mygallery.com



GET /Gallery?id=1 HTTP/1.1

Host: www.mygallery.com

```
public class GalleryController : Controller
{
    public ActionResult Index(int id)
    {
        return View();
    }
}
```



GET /Gallery?id=1 HTTP/1.1

Host: www.mygallery.com

```
public class MyModel
```

```
{
```

```
    public int Id { get; set; }
```

```
}
```

```
public class GalleryController : Controller
```

```
{
```

```
    public ActionResult Index(MyModel model)
```

```
{
```

```
        var idFromModel = model.Id;
```

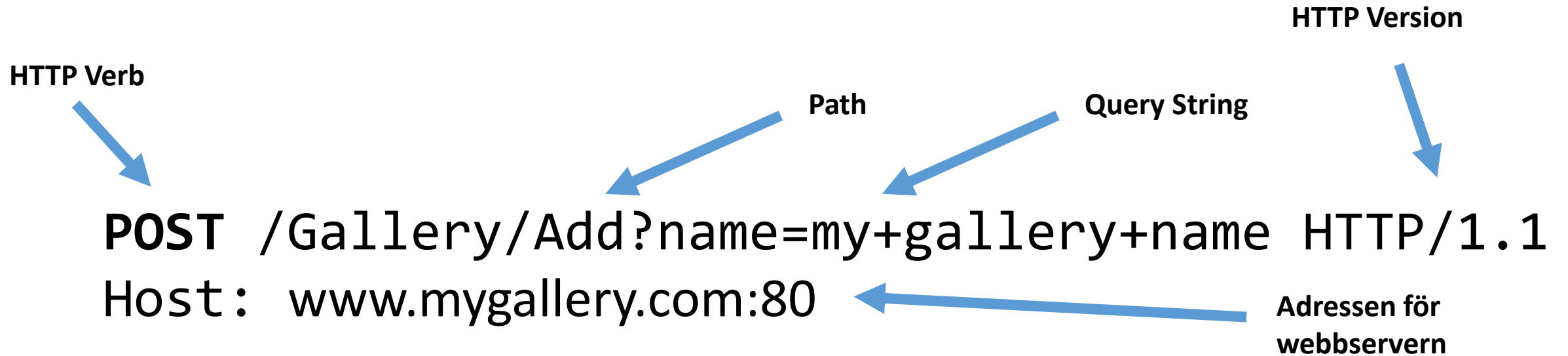
```
        return View();
```

```
}
```

```
}
```

ASP.NET skapar en
instans av MyModel
och sätter Id

POST



POST /Gallery/Add?name=my+gallery+name HTTP/1.1

Host: www.mygallery.com:80

```
public class GalleryController : Controller
{
    [HttpPost]
    public ActionResult Add(MyModel model)
    {
        return RedirectToAction("Index");
    }

    public ActionResult Index()
    {
        return View();
    }
}
```

```
public class MyModel
{
    public string Name { get; set; }
}
```

Det skapas automatiskt upp en instans av **MyModel** där **Name** är satt till "my gallery name"

```
<form action="/Gallery/Add" method="post">
  <input type="text" name="name"/>
  <input type="submit"/>
</form>
```

```
public class GalleryController : Controller
{
    [HttpPost]
    public ActionResult Add(MyModel model)
    {
        return RedirectToAction("Index");
    }

    public ActionResult Index()
    {
        return View();
    }
}
```

```
public class MyModel
{
    public string Name { get; set; }
}
```

Det skapas automatiskt upp en instans av **MyModel** där **Name** är satt till värdet av input-fältet.

POST

POST /Gallery/Add HTTP/1.1

Host: www.mygallery.com:80

Content-Type: **application/json**

Hur ska servern tolka
det vi skickar in?

JSON Data som
servern ska ta emot

```
{  
  "name" : "my gallery name"  
}
```

Body

POST /Gallery/Add HTTP/1.1
Host: www.mygallery.com:80
Content-Type: **application/json**

```
{  
  "name": "my gallery name"  
}
```

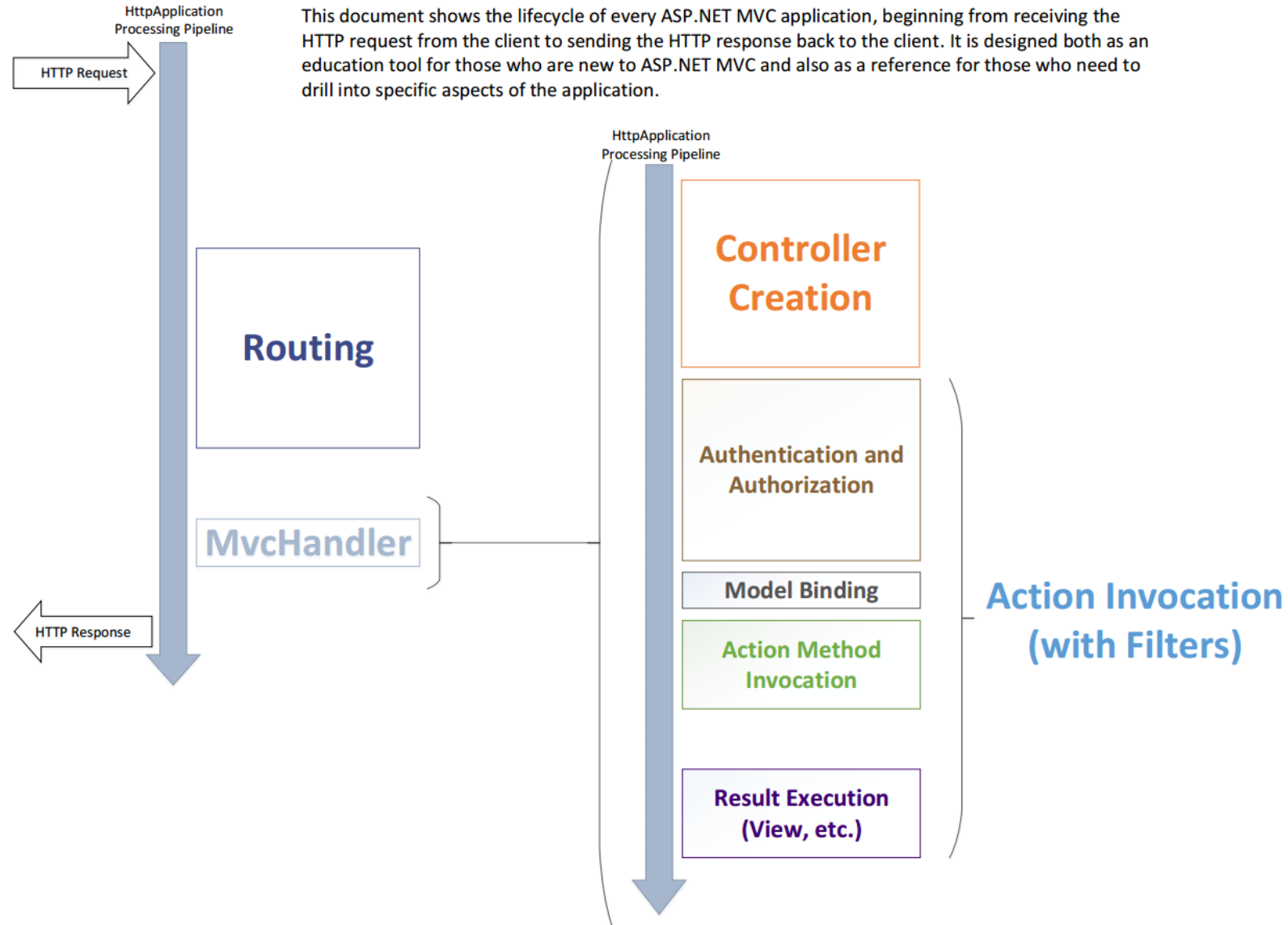
```
public class GalleryController : Controller  
{  
    [HttpPost]  
    public ActionResult Add(MyModel model)  
    {  
        return RedirectToAction("Index");  
    }  
}
```

```
public class MyModel  
{  
    public string Name { get; set; }  
}
```

```
public ActionResult Index()  
{  
    return View();  
}
```

Det skapas automatiskt upp en instans av **MyModel** där **Name** är satt till **"my gallery name"**

ASP.NET MVC 5 APPLICATION LIFECYCLE – HIGH-LEVEL VIEW



**GET är för att läsa –
Inga sidoeffekter**

POST är för lägg till

DELETE?

Delete

DELETE /Gallery/Delete/1 HTTP/1.1

Host: www.mygallery.com:80

[HttpDelete]

```
public ActionResult Delete(int id)
{
}
}
```

PUT?

JavaScript

**Vad är skillnad på kod
som körs i webbläsaren
och på webbservern?**

**ASP.NET körs på servern –
kommer aldrig åt något
på användarens dator**

JavaScript körs i webbläsaren

“Isolerad” miljö – Begränsat vad du kommer åt

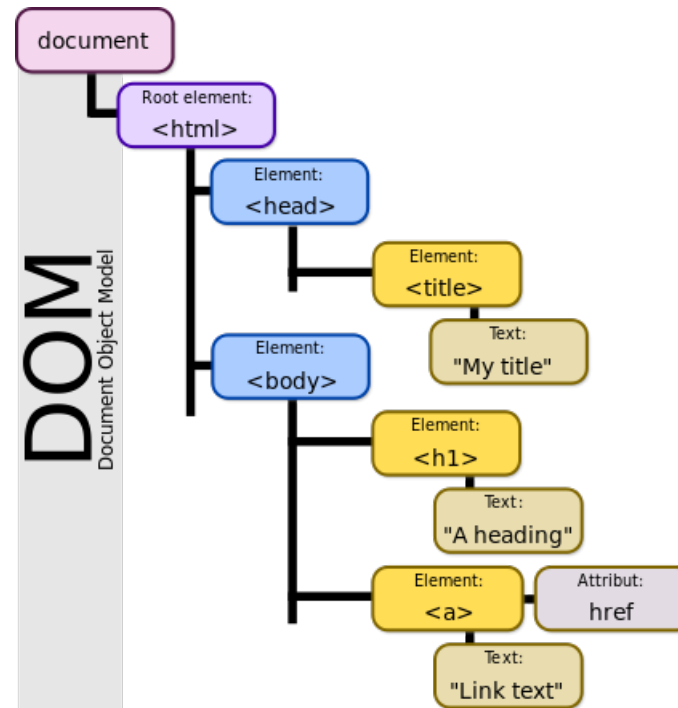
JavaScript är inte statiskt typat

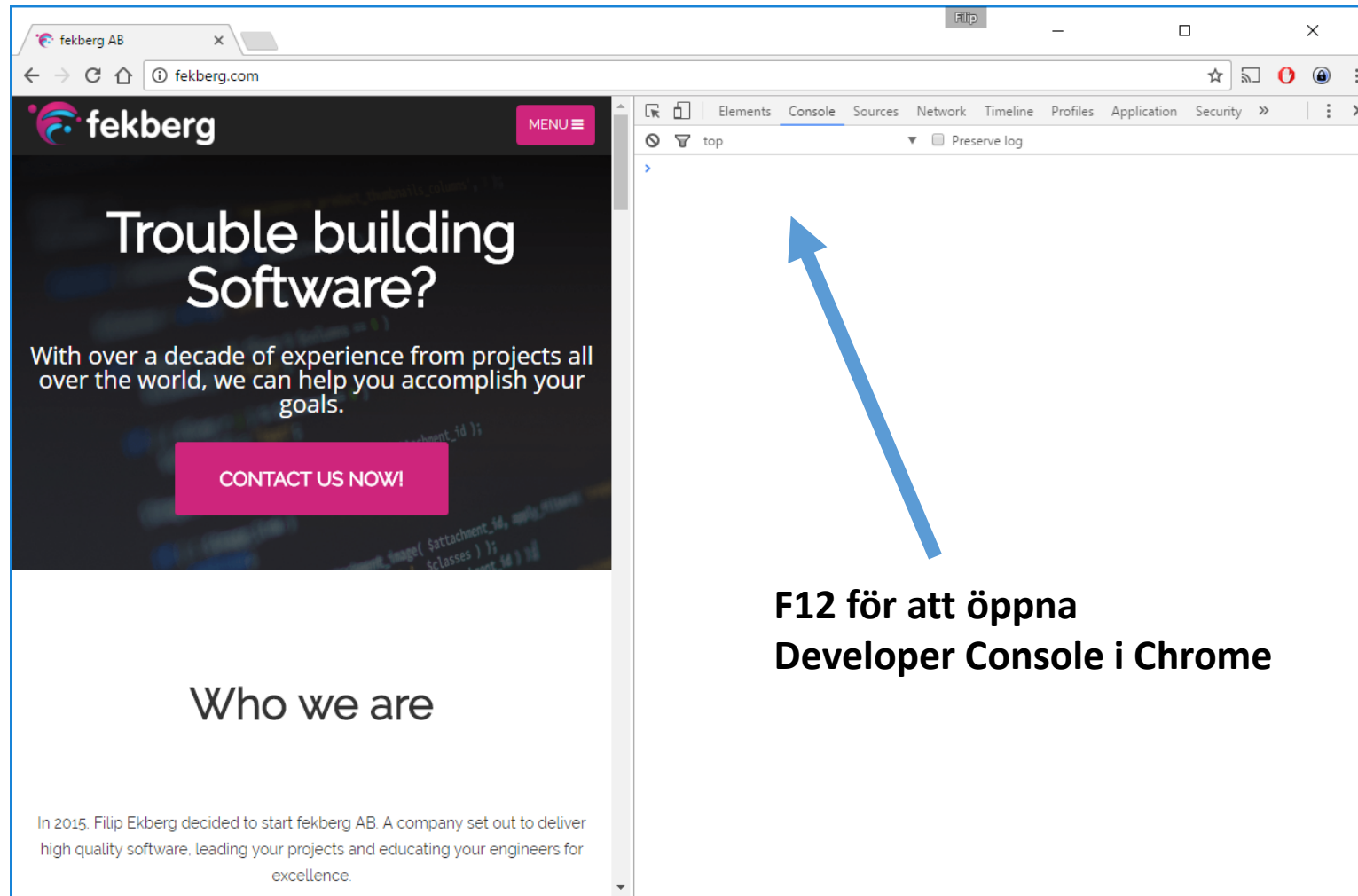
Dynamiskt programmeringsspråk

Bygg smarta klienter med hjälp av JavaScript

Jobba med DOM

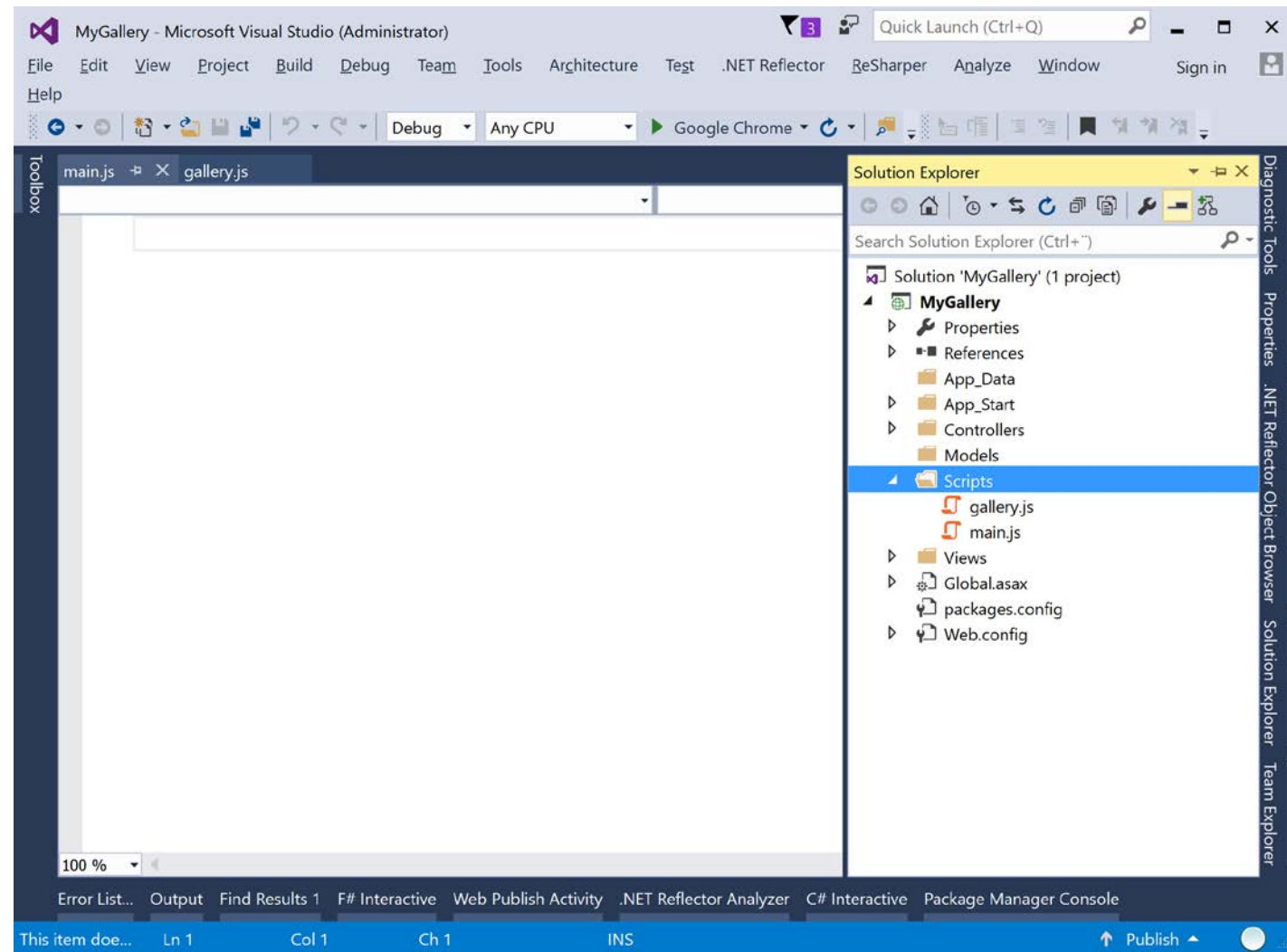
Document Object Model





**F12 för att öppna
Developer Console i Chrome**

Dela upp logik för olika sidor i olika JavaScript filer



```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
    <meta name="viewport" content="width=device-width" />
```

```
    <title>@ViewBag.Title</title>
```

```
</head>
```

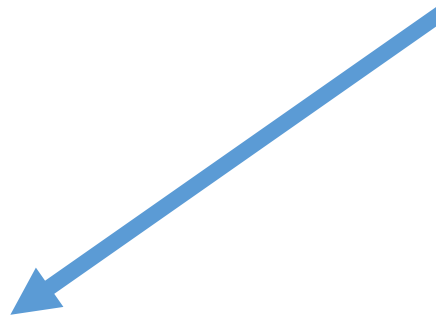
```
<body>
```

```
    <div>
```

```
        @RenderBody()
```

```
    </div>
```

Lägg till JavaScript sist!



```
    <script src="/Scripts/gallery.js"></script>
```

```
    <script src="/Scripts/main.js"></script>
```

```
</body>
```

```
</html>
```

```
<head>
  <meta name="viewport" content="width=device-width" />
  <title>@ViewBag.Title</title>

  <script type="text/javascript">

    if (window.location.href == "http://fekberg.com/") {
      alert("Hello there!");
    }

  </script>
</head>
```

Vad bör vi använda JavaScript till?

Validating

Redirects

```
location.href = "http://www.google.com";
```

Notifieringar

Growl

Smarta klienter

SPA – Single Page App

jQuery Growl

ksylvest.github.io/jquery-growl/

Warning!
The kitten is ugly!

Notice!
The kitten is cute!

Error!
The kitten is attacking!

jQuery Growl

Growl is a jQuery plugin designed to provide informative messages in the browser.

Example

- [Error!](#)
- [Notice!](#)
- [Warning!](#)

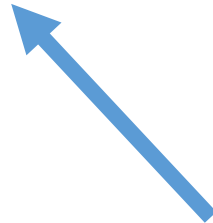
Installation

To install download one of these packages and include the jquery.growl.js and jquery.growl.css files in your head with the following:

ZIP **TAR**

Vänta på att DOM laddats?

```
(function() {  
  
})();
```



Kör funktionen

Lägg på Id för att komma åt
via JavaScript lättare

```
<form id="form" action="/Gallery/Add" method="post">  
  <input type="text" name="name" id="name"/>  
  <input type="submit" />  
</form>
```

Lägg på Id för att komma åt
via JavaScript lättare

```
<form id="form" action="/Gallery/Add" method="post">  
  <input type="text" name="name" id="name"/>  
  <input type="submit" />  
</form>
```

```
(function() {
```

```
  var form = document.getElementById("form");
```

```
})();
```

```
var form = document.getElementById("form");
```

```
form.addEventListener("submit", function (e) {  
});
```



Registrera en event handler



För eventet "submit"



Inline-funktion som
körs när eventet sker


```
function (e) {
```

Information om eventet, och
hjälpmedel för att påverka vad som
sker härnäst!

```
e.preventDefault();
```

Kör inte det inbyggda flödet för vad som
sker när jag submittar ett formulär –
posta alltså inte det till servern!

```
var galleryName =  
    document.getElementById("name");
```

```
alert(galleryName.value);
```

```
<input type="text" name="name" id="name"/>
```

Hämta värdet på vår input

```
}
```

@model Photo

```
@{  
    ViewBag.Title = $"Du tittar på {Model.Name}";  
}
```

```
<a class="delete"  
    href="/delete/"  
    data-imageId="@Model.Id">Delete @Model.Name</a>
```

`document.getElement`

- 📦 `getElementById` (in lib.d.ts)
- 📦 `getElementsByClassName` (in lib.d.ts)
- 📦 `getElementsByName` (in lib.d.ts)
- 📦 `getElementsByTagName` (in lib.d.ts)
- 📦 `getElementsByTagNameNS` (in lib.d.ts)
- 📦 `getAnonymousElementByAttribute` (in DHtml.d.ts)
- 📦 `getTransformToElement`

(elementId: string): Element in 'DomCore.d.ts' ▼

(elementId: string): HTMLElement in 'lib.d.ts'

```
var deleteLink =  
    document.getElementsByClassName("delete")[0];
```

Leta efter ett element med
CSS-klassen "delete".
OBS: Det kan finnas fler!

Registrera vad som händer
vid eventet "click"

Inline-funktion som
körs när eventet sker

```
deleteLink.addEventListener("click", function(e) {  
});
```

```
function (e) {
```

Information om eventet, och
hjälpmedel för att påverka vad som
sker härnäst!

```
e.preventDefault();
```

Vi vill inte navigera någonstans när vi
klickar på länken!

```
var imageId = this.getAttribute("data-imageId");
```

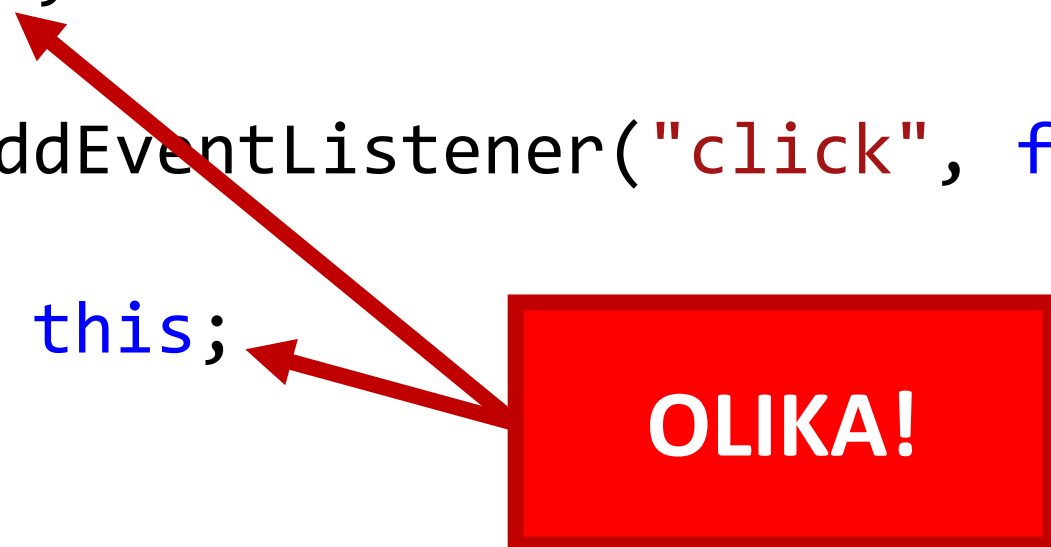
```
alert(imageId);
```

```
}
```

this pekar på "anchor", alltså
länken!

`data-imageId="@Model.Id"`

```
(function () {  
    var t1 = this;  
    deleteLink.addEventListener("click", function (e) {  
        var t2 = this;  
    });  
})();
```



OLIKA!

```
<a class="delete"  
  href="/delete/"  
  data-imageId="1">Delete Swimming with Sharks</a>
```

```
<a class="delete"  
  href="/delete/"  
  data-imageId="2">Delete Sunset at the Beach</a>
```

```
var links = document.getElementsByClassName("delete");  
for (var i = 0; i < links.length; i++) {  
    links[i].addEventListener("click", function (event) {  
        event.preventDefault();  
        alert(this.getAttribute("data-imageId"));  
    });  
}
```

Alla länkarna med klassen delete

Lägg på en event-listener på samtliga länkar


```
var number = 0;  
var numberString = "0";  
  
alert(number == numberString);  
alert(number === numberString);
```

```
var number = 0;  
var numberString = "";  
  
alert(number == numberString);  
alert(number === numberString);
```

Du kommer jobba mycket med

- `if / else`
- `for`
- `switch`
- `document`
 - `getElement*`
 - Olika events

Vad händer om jag stänger av JavaScript?

JavaScript Cheat Sheet

<https://www.cheatography.com/davechild/cheat-sheets/javascript/>

A photograph of a workspace featuring a silver laptop on the left, two smartphones (one black, one silver) on the right, and a tablet in the foreground. The laptop screen shows a website with a 'NEW PHOTOS ADDED' section. A large white text overlay is centered on the image.

MVC med Javascript och Ajax

Lektion 4 - Javascript & Ajax Fortsätter

Dagens mål

- Vad är JavaScript och jQuery – Hur kommer vi igång med klient-programmering?
- Vad är Asynkron Programmering?
- Vad är Ajax?
- GET, POST via Ajax
- Hur hanterar vi svar från servern via GET/POST med JavaScript?
- Hur manipulerar vi DOM?
- Validering - Automatisk & Manuell validering på klientsidan
- Validering - Server- eller Klient-validering?
- Ramverk för att underlätta smarta klienter
- SPA (Single Page Apps)
- Hantera uppdatering av segment i DOM
- Hantera fel
- Jobba med JSON

jQuery

Ett bibliotek för att göra det enklare att jobba med JavaScript och DOM

<https://jquery.com/>

**Allt går att göra
UTAN jQuery!**

**Extremt mycket
tredjeparts-komponenter!**

Kör via CDN

Install-Package jQuery

```
/// <reference path="jquery-3.1.1.intellisense.js" />
```

\$

Prefix för att använda jQuery

Selector – Vad är det vi använder jQuery mot?



`$(document)`

Kan skriva jQuery istället för \$

Selector – Vad är det vi använder jQuery mot?



jQuery(document)

Använd jQuery för att hantera elementet med id myId




```
$("#myId").hide();
```



Action – Vad är det vi vill göra?

Använd jQuery för att hantera elementet med klassen myId



```
$( ".myId" ).show();
```



Action – Vad är det vi vill göra?

Använd jQuery för att hantera ALLA div-element



```
$(".div").hide();
```



Action – Vad är det vi vill göra?

Använd jQuery för att hantera alla länkar

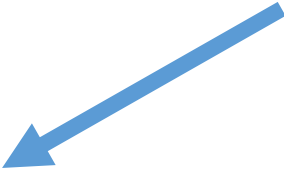
Inline-funktion som tar vilket index
samt element i listan av länkar

Iterera över alla länkar

```
$("a").each(function(index, element) {  
    alert(element);  
});
```

Visa en alert-box med länken

Använd jQuery för att hantera document




```
$(document).ready(function() {  
});
```



Action – Vad är det vi vill göra?

```
(function () {  
    var deleteLink = document.getElementsByClassName("delete")[0];  
    deleteLink.addEventListener("click", function (e) {  
    });  
})();
```

**CSS-selector för att hämta första
elementet med klassen delete**



```
$(document).ready(function() {  
    $(".delete:first").click(function(event) {  
        event.preventDefault();  
  
        alert($(this).attr("data-imageId"));  
    });  
});
```

**Använd jQuery för elementet vi får ett
click-event för – Detta för att vi kan
använda t.ex. attr, hide, show, etc**

Manipulera DOM

```
$( 'div' ).each( function (index, element) {  
    $(this).html( '<strong>' + $(this).html() + '</strong>' );  
});
```

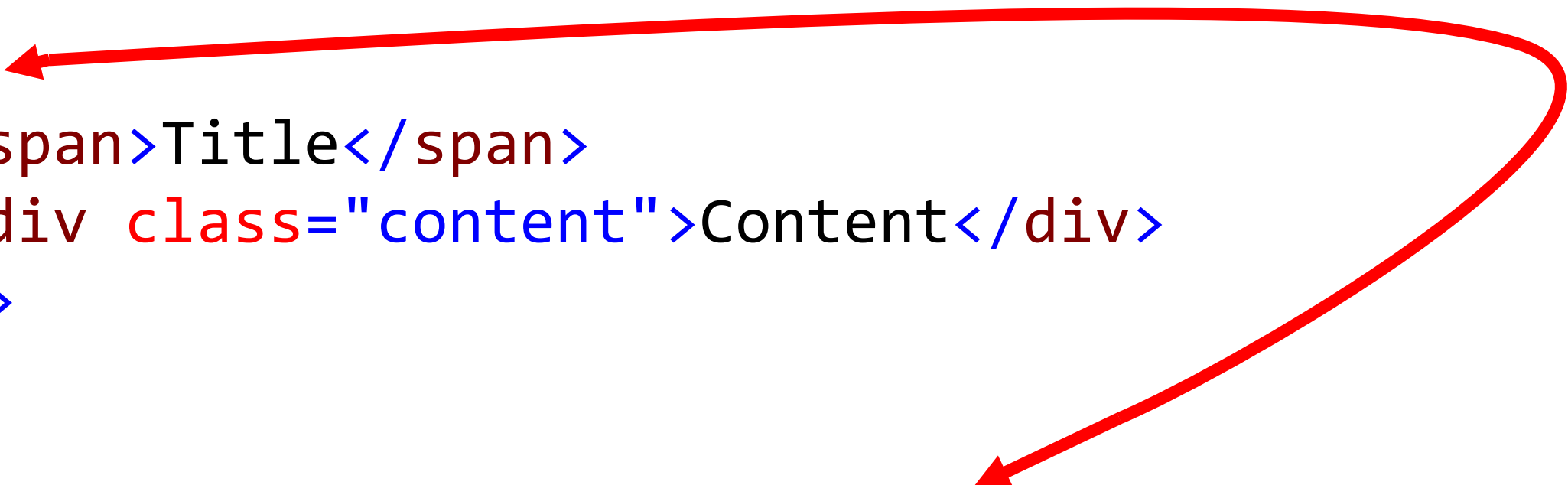


```
<div id="test">Alright!</div>
```

```
$("#div#test").html();
```

```
$("#div#test").html("<strong> " + $("#div#test").html() + "</strong>");
```

```
<div>  
  <span>Title</span>  
  <div class="content">Content</div>  
</div>
```



```
var parent = $('div.content').parent();  
var span = parent.children("span:first");  
span.html("<h1>" + span.html() + "</h1>");
```

jQuery Validation

```
install-package jquery.validation
```

```
install-package Microsoft.jQuery.Unobtrusive.Validation
```

test

Submit

```
<input class="valid"
      data-val="true"
      data-val-required="The Name field is required."
      id="Name"
      name="Name"
      type="text"
      value=""
      data-cip-id="Name"
      aria-required="true"
      aria-describedby="Name-error"
      aria-invalid="false">
```

Submit

```
<input class="input-validation-error"  
  data-val="true"  
  data-val-required="The Name field is required."  
  id="Name"  
  name="Name"  
  type="text"  
  value=""  
  data-cip-id="Name"  
  aria-required="true"  
  aria-describedby="Name-error"  
  aria-invalid="true">
```



```
$("form").submit(function (e) {  
    var form = $(this);  
  
    form.validate();  
  
    if (!form.valid()) {  
        e.preventDefault();  
    }  
});
```

```
public class Photo
{
    public Guid Id { get; set; }

    [Required]
    public string Name { get; set; }
}
```

Server-validering

```
[HttpPost]
public ActionResult Index(Photo model)
{
    if (ModelState.IsValid)
    {
        // Add photo
    }

    return View(model);
}
```

Använd båda!

Tips på vad du kan göra

- Hitta närliggande element
- Sök efter element i hela DOM:en
- Ta bort, Lägg till och Ändra innehåll
- Lägg till och Ta bort klasser
- Fade In/Out
- Timer – Kör något regelbundet
- AJAX

jQuery Cheat Sheet

<https://oscarotero.com/jquery/>

Mer JavaScript


```
function Photo(name) {  
    this.name = name;  
    this.id = null;  
}
```

```
var photo = new Photo("test");  
console.log(photo.name);
```

NodeJS

Inget vi kommer använda i denna kursen

MVC/MVVM på klienten

**Olika ramverk för att göra
det enklare att bygga
smarta klienter**

Client-side - JavaScript

MVC / MVVM

View Model
Bindings
Controllers
Actions



Serverside - ASP.NET

MVC

Controller
Action
View
View Model

Busines-lager

Integrations
Domain Validation
Domain Model

Data-lager

Repositories
Entity Framework
Data Model

MVC eller MVVM?

Model – View – ViewModel

Automatiska bindningar i JavaScript

Ta hjälp av ramverk

Angular

Kraftfult, Vedertaget och bra – Men svårt att komma igång

Vue.js

Components istället för Controllers

Samma koncept – olika namn

```
<script src="https://unpkg.com/vue/dist/vue.js"></script>
```

```
<div id="app">  
    {{ message }}  
</div>
```

```
var app = new Vue({  
  el: '#app',  
  data: {  
    message: 'Hello Vue!'  
  }  
});
```

```
var app = new Vue({  
  el: '#app',  
  data: {  
    message: 'Hello Vue!',  
    someMessage: Detta laddades in ' + new Date()  
  }  
});
```



```
<span v-bind:title="someMessage">
```

Hover your mouse over me for a few seconds
to see my dynamically bound title!

```
</span>
```

```
<ol>
  <li v-for="todo in todos">
    {{ todo.text }}
  </li>
</ol>
```

```
var app = new Vue({  
  el: '#app',  
  data: {  
    message: 'Hello Vue!',  
    todos: [  
      { text: 'Lära oss MVC' },  
      { text: 'Lära oss JavaScript' },  
      { text: 'Lära oss Säkerhet' }  
    ]  
  }  
});
```

```
<div id="app">  
  {{ message }}
```

```
  <button v-on:click="reverseMessage">
```

```
    Reverse Message
```

```
  </button>
```

```
</div>
```

```
var app = new Vue({  
  el: '#app',  
  data: {  
    message: 'Hello Vue!'  
  },  
  methods: {  
    reverseMessage: function() {  
      this.message = this.message.split('').reverse().join('')  
    }  
  }  
});
```

```
<div id="app">
  {{ message }}
  <input v-model="message">
  <button v-on:click="reverseMessage">
    Reverse Message
  </button>
</div>
```

<https://vuejs.org>



Vue.js

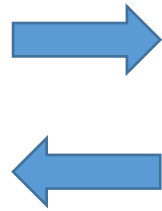
Enklare att få en bra arkitektur för klient-kod

Inte ett krav för labbarna!

Client-side - JavaScript

MVC / MVVM

View Model
Bindings
Controllers
Actions



Serverside - ASP.NET

MVC

Controller
Action
View
View Model

Busines-lager

Integrations
Domain Validation
Domain Model

Data-lager

Repositories
Entity Framework
Data Model

Asynkron Programmierung

**Något som sker samtidigt
som nuvarande exekvering –
utan att låsa applikationen!**

Undvik att låsa applikationen

Synkront är blockerande
Asynkront är icke-blockerande

“Gör X åt mig – när du är klar säg till”

Exempel

Användaren **Laddar upp** en **bild**

Webbsidan **visar** en **"spinner"**

Användaren **får reda på** att bilden **sparats**

Utan att sidan laddades om!

Vad händer om de tar för lång tid?

AJAX

Asynchronous JavaScript and XML

XMLHttpRequest

```
var request = new XMLHttpRequest();
```

← Förbered för en ny AJAX request

```
request.addEventListener("load", function(e) {  
    alert(this.responseText);  
});
```

← Vad händer när vi får ett svar?

```
request.open("GET", "/Gallery/Info");
```

← Vad ska vi anropa?

```
request.send();
```

← Skicka iväg vår request

```
public class Photo
{
    public Guid Id { get; set; }
    public string Name { get; set; }
}

public class GalleryController : Controller
{
    [HttpPost]
    public ActionResult Add(Photo model)
    {
        return Content($"Added {model.Name}");
    }
}
```

```
var model = {  
    id : "bcf54577-600a-4813-b026-838ff385d3af",  
    name : "Sunset at the beach"  
};  
  
var request = new XMLHttpRequest();  
  
request.addEventListener("load", function(e) {  
    alert(this.responseText);  
});  
  
request.open("POST", "/Gallery/Add");  
  
request.setRequestHeader("Content-Type",  
    "application/json;charset=UTF-8");  
  
request.send(JSON.stringify(model));
```




Användaren klickar på "Delete"



**Visa en laddnings-indikation
för användaren**



Skicka iväg ett anrop i bakgrunden mot servern



**Du har fått svar!
Ta bort laddnings-indikatorn
och hantera svaret**



Webbservern hanterar anropet

jQuery + AJAX

```
$.ajax({  
    type: "GET",  
    url: url,  
    data: data,  
    success: success,  
    dataType: dataType  
});
```

```
$.ajax({  
    type: "GET",  
    url: "/Gallery/Info",  
    success: function(data) {  
        alert(data);  
    }  
});
```

```
$.get("/Gallery/Info", function(data) {  
    alert(data);  
});
```

```
$.ajax({  
    type: "POST",  
    url: url,  
    data: data,  
    success: success,  
    dataType: dataType  
});
```

```
$.ajax({  
  type: "POST",  
  url: "/Gallery/Add",  
  data: JSON.stringify(model),  
  success: function(data) {  
    alert(data);  
  },  
  error: function(data) {  
  
  },  
  contentType: "application/json; charset=utf-8",  
  dataType: "text"  
});
```

Skicka modellen som JSON

Hantera ett svar

Det är JSON vi skickar

Vad för data får vi tillbaka?

JSON


```
[HttpPost]
public ActionResult Add(Photo model)
{
    model.Name = "Test";
    return Json(model);
}
```

```
$.ajax({  
  type: "POST",  
  url: "/Gallery/Add",  
  data: JSON.stringify(model),  
  success: function(data) {  
    model.id = data.id;  
    model.name = data.name;  
  },  
  contentType: "application/json; charset=utf-8",  
  dataType: "json"  
});
```

Svaret parsas automatiskt från JSON

Hantera svaret som JSON

```
$.post("/Gallery/Add", model, function (data) {  
    alert(data.Name);  
});
```



Case-sensitive!



Svaret har deserialiserats från
JSON automatiskt

```
$.post("/Gallery/Add",  
      $("form").serialize(),  
      function (data) {}  
);
```



Serialisera formuläret och
skicka med AJAX-anropet

```
function Photo(name) {  
    this.name = name;  
    this.id = null;  
}
```

```
var photo = new Photo("test");  
console.log(photo.name);
```

```
console.log(JSON.stringify(photo))
```

Ladda upp fler filer

```
<input type="file" id="files" name="files" multiple="multiple" />
```

```
[HttpPost]
public ActionResult Index(HttpPostedFileBase[] files)
{
    foreach (var file in files)
    {
        file.SaveAs(Server.MapPath($"~/Uploads/{file.FileName}"));
    }
}
```



```
var form = document.getElementById("form");  
  
var xhr = new XMLHttpRequest();  
  
xhr.open("post", form.action);  
  
xhr.send(new FormData(form));
```

```
var form = document.getElementById("form");  
$.ajax({  
    url: form.action,  
    data: new FormData(form),  
    type: 'POST',  
  
    // Nedan måste sättas för att det ska fungera!  
    contentType: false,  
    processData: false  
});
```

Byta ut DOM-element

```
[HttpPost]
public ActionResult Index(HttpPostedFileBase[] files)
{
    foreach (var file in files)
    {
        file.SaveAs(Server.MapPath($"~/Uploads/{file.FileName}"));
    }

    return PartialView("Success", files.Select(x => x.FileName));
}
```

```
var xhr = new XMLHttpRequest();

xhr.onloadend = function (e) {
    $("#result").html(this.responseText);
}

xhr.open("post", formElement.action);
xhr.send(new FormData(formElement));
```

Labb 2

Fortsätt med Bildgalleriet

- Controllers & Actions för att kommentera på individuella bilder samt album
- Controller & Action för att skapa ett album där man kan välja en samling av de bilderna som laddats upp
- Validering vid samtliga POST-anrop samt använda AntiForgeryTokens(Attribut) för samtliga POST-anrop
- Alla POST-anrop ska ske via AJAX – tänk på att inte låta användaren trycka på samma knapp flera gånger, undvik att låta användaren dubbelposta
- Alla AJAX-anrop ska vara korrekt felhanterade och felmeddelanden ska presenteras till användaren snyggt
- Du ska kunna bläddra bland bilder i ett fotoalbum utan att göra en "postback", d.v.s. utnyttja AJAX för att ladda in nästa bild och presentera den
- Vid varje AJAX-anrop ska du visa en laddnings-indikation för användaren
- Du ska automatiskt hämta nya kommenterar på bilder var 10:e sekund – Om du öppnar två webbläsare och kommenterar på samma bild, ska båda klienterna få bådas kommentarer utan att behöva ladda om webbsidan

**Skickas in på
Torsdag!**

mail@filipekberg.se



Håll följande i åtanke

- Håll din Controller liten
- Håll dina Views rena
- Dela upp i Partial Views
- Använd dig av Layout
- Undvik ViewBag om du inte behöver den
- Validera formulärdata
- Visa laddnings-indikation för klienten
- Sköt alla POST via AJAX

Skicka Github länk för review!

- Skicka en Github-länk till labben
- Jag återkommer med feedback och förbättringsförslag