

CSC 591

Systems Attacks and Defenses

Web Security

Alexandros Kapravelos
akaprav@ncsu.edu

(Derived from slides by Giovanni Vigna and Adam Doupe)

The XMLHttpRequest Object

- Microsoft developers working on Outlook Web Access for Exchange 2000
- Scalability problems with traditional web application
- They created a DHTML version (circa) 1998 using an ActiveX control to fetch bits of data from the server using JavaScript
- OWA team got the MSXML team (MSXML is Microsoft's XML library, and it shipped with IE) to include their ActiveX control (hence the XML in the name)
 - Shipped in IE 5, March 1999
- Exchange 2000 finally released in November 2000, and OWA used the ActiveX Object
- Added by Netscape in December 2000 as XMLHttpRequest
- Find the full story here: <https://hackerfall.com/story/the-story-of-xmlhttp-2008>



The XMLHttpRequest Object

- Allows JavaScript code to (asynchronously) retrieve data from the server, then process the data and update the DOM
- Because of the origin (ActiveX control on Windows and included in Netscape's DOM), used to need two different ways to instantiate the control
 - Most browsers (including Microsoft Edge):
 - `http_request = new XMLHttpRequest();`
 - Internet Explorer
 - `http_request = new ActiveXObject("Microsoft.XMLHTTP");`

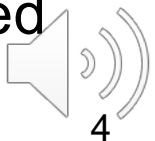


Creating an XMLHttpRequest

- Using the `onreadystatechange` property of an `XMLHttpRequest` object one can set the action to be performed when the result of a query is received

```
http_request.onreadystatechange = function(){  
    <JS code here>  
};
```

- Then, one can execute the request
- `http_request.open('GET' ,
 'http://example.com/show.php?keyword=foo' , true);`
- `http_request.send();`
- Note that the third parameter indicates that the request is asynchronous, that is, the execution of JavaScript will proceed while the requested document is being downloaded



XMLHttpRequest Lifecycle

- The function specified using the "onreadystatechange" property will be called at any change in the request status
 - 0 (uninitialized: Object is not initialized with data)
 - 1 (loading: Object is loading its data)
 - 2 (loaded: Object has finished loading its data)
 - 3 (interactive: User can interact with the object even though it is not fully loaded)
 - 4 (complete: Object is completely initialized)
- Usually wait until the status is “complete”
 - `if (http_request.readyState == 4) {
operates on data} else {
not ready, return}`



XMLHttpRequest Success

- After having received the document (and having checked for a successful return code – 200) the content of the request can be accessed:
 - As a string by calling:
`http_request.responseText`
 - As an XMLDocument object:
`http_request.responseXML`
 - In this case the object can be modified using the JavaScript DOM interface



XMLHttpRequest Example

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>AJAX Example</title>
  </head>
  <body>
    <h1>AJAX Example</h1>
    <div id='insert_here'>
    </div>
    <script>
      ...
    </script>
  </body>
</html>
```



XMLHttpRequest Example

```
if (typeof XMLHttpRequest != "undefined") {
    var http_request = new XMLHttpRequest();
}
else {
    var http_request = new ActiveXObject("Microsoft.XMLHTTP");
}
if (typeof console == "undefined") {
    console = { "log" : function (text) { alert(text); } };
}
http_request.onreadystatechange = function () {
    console.log(http_request.readyState);
    if (http_request.readyState === 4) {
        var text = http_request.responseText;
        var new_node = document.createTextNode(text);
        document.getElementById('insert_here').appendChild(new_node);
    }
};
console.log("Before Request");
http_request.open('GET', 'ajax_test.txt', true);
http_request.send();
console.log("After Request");
```



Browser window: AJAX Example | 192.168.84.165/code/ajax.html

AJAX Example

Paused in debugger

Elements Network Sources Timeline Profiles Resources Audits Console

Sources Content Snippets ajax.html x

```

21 }
22 http_request.onreadystatechange = function () {
23   console.log(http_request.readyState);
24   if (http_request.readyState === 4) {
25     var text = http_request.responseText;
26     var new_node = document.createTextNode(text);
27     document.getElementById('insert_here').appendChild(new_node);
28   }
29 };
30 console.log("Before Request");
31 http_request.open('GET', 'ajax_test.txt', true);
32 http_request.send();
33 console.log("After Request");
34 </script>
35 </body>
36 </html>
37

```

Line 30, Column 1

Watch Expressions +

Call Stack Async

(anonymous ajax.html:30 function)

Paused on a JavaScript breakpoint.

Scope Variables

Global Window

Breakpoints

- ajax.html:24 if (http_request.readyState...
- ajax.html:30 console.log("Before Requ...

Console Search Emulation Rendering

<top frame> Preserve log



Paused in debugger

AJAX Example

192.168.84.165/code/ajax.html

Elements Network Sources Timeline Profiles Resources Audits Console

Sources Content Snippets ajax.html x

```

22 http_request.onreadystatechange = function () {
23   console.log(http_request.readyState);
24   if (http_request.readyState === 4) {
25     var text = http_request.responseText;
26     var new_node = document.createTextNode(text);
27     document.getElementById('insert_here').appendChild(new_node);
28   }
29 };
30 console.log("Before Request");
31 http_request.open('GET', 'ajax_test.txt', true);
32 http_request.send();
33 console.log("After Request");
34 </script>
35 </body>
36 </html>
37

```

Line 31, Column 1

Watch Expressions +

Call Stack Async

(anonymous ajax.html:31 function)

Paused on a JavaScript breakpoint.

Scope Variables

Global Window

Breakpoints

- ajax.html:24 if (http_request.readyState...
- ajax.html:30 console.log("Before Requ...

Console Search Emulation Rendering

<top frame> Preserve log

Before Request ajax.html:30



Browser window: AJAX Example, 192.168.84.165/code/ajax.html

Paused in debugger

AJAX Example

Debugger interface showing the Sources panel with the following code:

```

21 }
22 http_request.onreadystatechange = function () {
23   console.log(http_request.readyState);
24   if (http_request.readyState === 4) {
25     var text = http_request.responseText;
26     var new_node = document.createTextNode(text);
27     document.getElementById('insert_here').appendChild(new_node);
28   }
29 };
30 console.log("Before Request");
31 http_request.open('GET', 'ajax_test.txt', true);
32 http_request.send();
33 console.log("After Request");
34 </script>
35 </body>
36 </html>
37

```

The Call Stack shows:

- ajax.html:24 http_request.onreadystatechange
- (anonymous function) ajax.html:31

Paused on a JavaScript breakpoint.

Scope Variables:

- Local
 - new_node: undefined
 - text: undefined

Console:

```

Before Request ajax.html:30
1 ajax.html:23

```



AJAX Example

Paused in debugger

192.168.84.165/code/ajax.html

Elements Network Sources Timeline Profiles Resources Audits Console

Sources Content Snippets ajax.html x

```

21 }
22 http_request.onreadystatechange = function () {
23   console.log(http_request.readyState);
24   if (http_request.readyState === 4) {
25     var text = http_request.responseText;
26     var new_node = document.createTextNode(text);
27     document.getElementById('insert_here').appendChild(new_node);
28   }
29 };
30 console.log("Before Request");
31 http_request.open('GET', 'ajax_test.txt', true);
32 http_request.send();
33 console.log("After Request");
34 </script>
35 </body>
36 </html>
37

```

Line 33, Column 1

Watch Expressions +

Call Stack Async

(anonymous ajax.html:33 function)

Paused on a JavaScript breakpoint.

Scope Variables

Global Window

Breakpoints

- ajax.html:24 if (http_request.readyState...
- ajax.html:30 console.log("Before Requ...

Console Search Emulation Rendering

<top frame> Preserve log

Before Request ajax.html:30

1 ajax.html:23



Paused in debugger

AJAX Example

192.168.84.165/code/ajax.html

Elements Network Sources Timeline Profiles Resources Audits Console

Sources Content Snippets ajax.html x

```

22 http_request.onreadystatechange = function () {
23   console.log(http_request.readyState);
24   if (http_request.readyState === 4) {
25     var text = http_request.responseText;
26     var new_node = document.createTextNode(text);
27     document.getElementById('insert_here').appendChild(new_node);
28   }
29 };
30 console.log("Before Request");
31 http_request.open('GET', 'ajax_test.txt', true);
32 http_request.send();
33 console.log("After Request");
34 </script>
35 </body>
36 </html>
37

```

Line 24, Column 1

Watch Expressions +

Call Stack Async

ajax.html:24
http_request.onreadystatechange

Paused on a JavaScript breakpoint.

Scope Variables

Local

- new_node: undefined
- text: undefined
- this: XMLHttpRequest

Global Window

Console Search Emulation Rendering

<top frame> Preserve log

Before Request ajax.html:30

1 ajax.html:23

After Request ajax.html:33

2 ajax.html:23



Paused in debugger

AJAX Example

192.168.84.165/code/ajax.html

Elements Network Sources Timeline Profiles Resources Audits Console

Sources Content Snippets ajax.html x

```

21 }
22 http_request.onreadystatechange = function () {
23   console.log(http_request.readyState);
24   if (http_request.readyState === 4) {
25     var text = http_request.responseText;
26     var new_node = document.createTextNode(text);
27     document.getElementById('insert_here').appendChild(new_node);
28   }
29 };
30 console.log("Before Request");
31 http_request.open('GET', 'ajax_test.txt', true);
32 http_request.send();
33 console.log("After Request");
34 </script>
35 </body>
36 </html>
37

```

Line 24, Column 1

Watch Expressions +

Call Stack Async

ajax.html:24
http_request.onreadystatechange
change

Paused on a JavaScript breakpoint.

Scope Variables

Local

- new_node: undefined
- text: undefined
- this: XMLHttpRequest

Global Window

Console Search Emulation Rendering

<top frame> Preserve log

Before Request ajax.html:30

1 ajax.html:23

After Request ajax.html:33

2 ajax.html:23

3 ajax.html:23



AJAX Example

Paused in debugger

192.168.84.165/code/ajax.html

Elements Network Sources Timeline Profiles Resources Audits Console

Sources Content ... Snippets ajax.html x

```

21 }
22 http_request.onreadystatechange = function () {
23   console.log(http_request.readyState);
24   if (http_request.readyState === 4) {
25     var text = http_request.responseText;
26     var new_node = document.createTextNode(text);
27     document.getElementById('insert_here').appendChild(new_node);
28   }
29 };
30 console.log("Before Request");
31 http_request.open('GET', 'ajax_test.txt', true);
32 http_request.send();
33 console.log("After Request");
34 </script>
35 </body>
36 </html>
37

```

Line 24, Column 1

Watch Expressions +

Call Stack Async

ajax.html:24
http_request.onreadystatechange

Paused on a JavaScript breakpoint.

Scope Variables

Local

- new_node: undefined
- text: undefined
- this: XMLHttpRequest

Global Window

Console Search Emulation Rendering

<top frame> Preserve log

Before Request [ajax.html:30](#)

1 [ajax.html:23](#)

After Request [ajax.html:33](#)

2 [ajax.html:23](#)

3 [ajax.html:23](#)

4 [ajax.html:23](#)



The screenshot shows a web browser window with the title "AJAX Example" and the URL "192.168.84.165/code/ajax.html". The page content includes the heading "AJAX Example" and the text "TEST AJAX".

The Chrome DevTools interface is open, showing the "Sources" panel with the file "ajax.html" selected. The code in the editor is as follows:

```

21 }
22 http_request.onreadystatechange = function () {
23     console.log(http_request.readyState);
24     if (http_request.readyState === 4) {
25         var text = http_request.responseText;
26         var new_node = document.createTextNode(text);
27         document.getElementById('insert_here').appendChild(new_node);
28     }
29 };
30 console.log("Before Request");
31 http_request.open('GET', 'ajax_test.txt', true);
32 http_request.send();
33 console.log("After Request");
34 </script>
35 </body>
36 </html>
37

```

The "Console" panel at the bottom shows the following log entries:

Message	File	Line
Before Request	ajax.html	30
1	ajax.html	23
After Request	ajax.html	33
2	ajax.html	23
3	ajax.html	23
4	ajax.html	23



AJAX Example

192.168.84.165/code/ajax.html

AJAX Example

TEST AJAX

Elements Network Sources Timeline Profiles Resources Audits Console

Preserve log Disable cache

Name Path	Method	Status Text	Type	Initiator	Size Content	Time Latency	Timeline
ajax.html /code	GET	200 OK	text/ht...	Other	809 B 983 B	6 ms 4 ms	
ajax_test.txt /code	GET	304 Not M...	text/pl...	ajax.html:32 Script	177 B 10 B	4 ms 3 ms	

2 requests | 986 B transferred | 2.94 s (load: 4.52 s, DOMContentLoaded: 4.52 s)

Console Search Emulation Rendering

<top frame> Preserve log

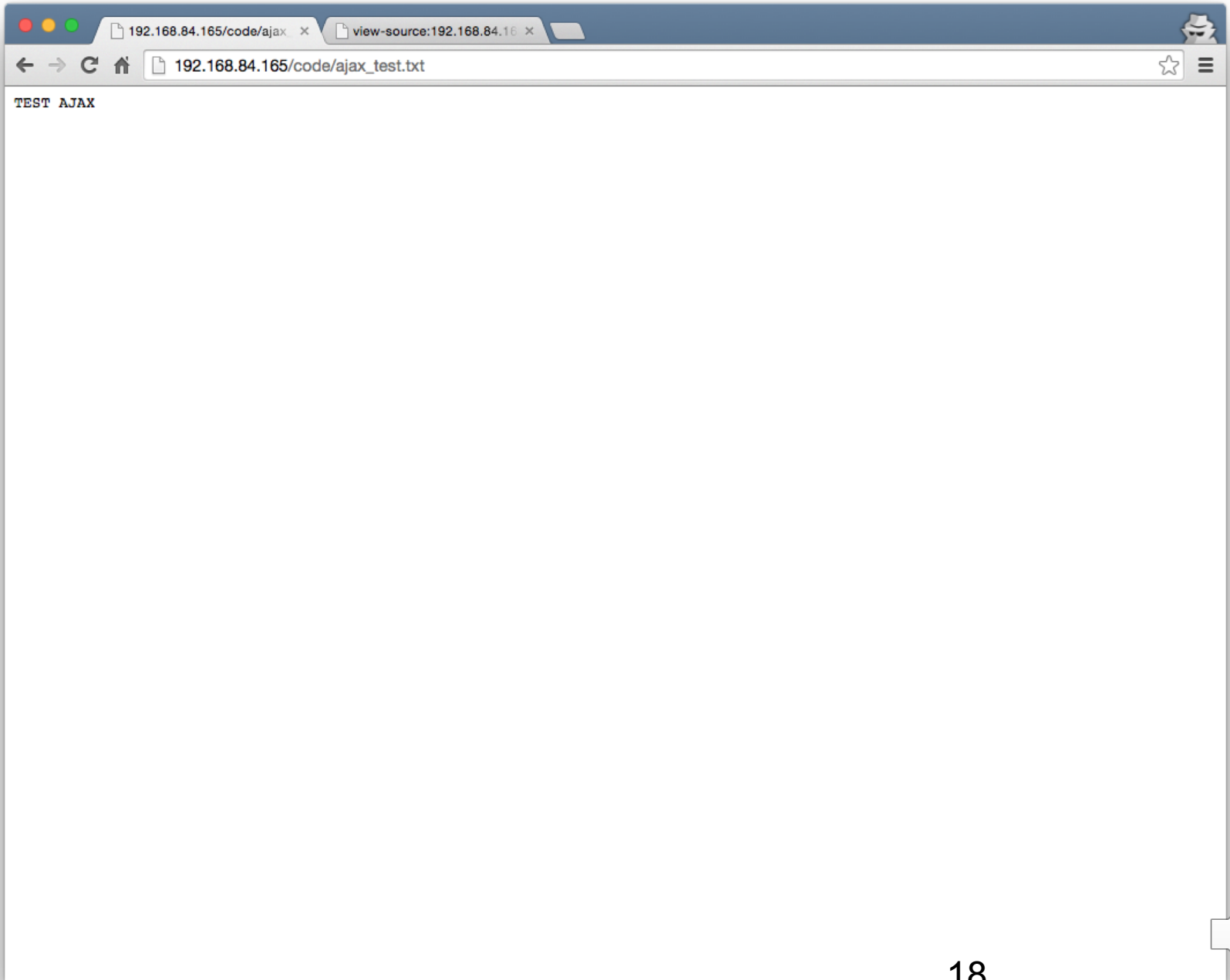
```

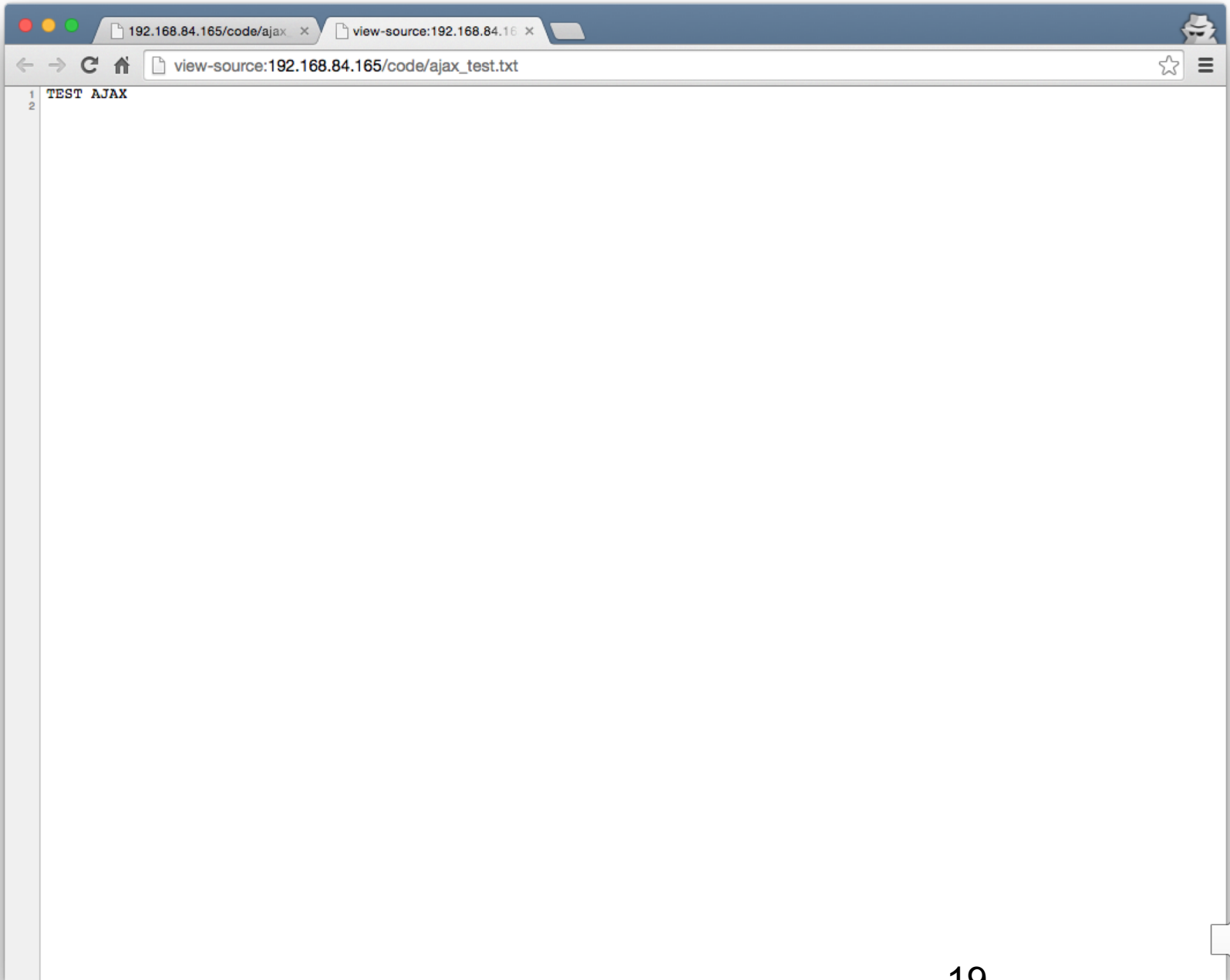
Before Request
1
After Request
2
3
4

```

ajax.html:30
 ajax.html:23
 ajax.html:33
 ajax.html:23
 ajax.html:23
 ajax.html:23







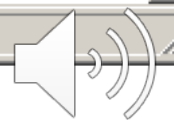
The image shows a web browser window with two tabs. The active tab is titled 'view-source:192.168.84.165/code/ajax_test.txt'. The address bar contains the same URL. The main content area displays the source code for the file 'ajax_test.txt', which consists of two lines: '1 TEST' and '2 AJAX'. The browser interface includes standard navigation buttons (back, forward, refresh, home) and a search icon. A speaker icon is visible in the bottom right corner of the browser window.

```
1 TEST  
2 AJAX
```

The image shows a screenshot of a Microsoft Internet Explorer browser window. The title bar reads "AJAX Example - Microsoft Internet Explorer". The address bar contains the URL "http://192.168.84.165/code/ajax.html". The main content area displays the text "AJAX Example" in a large, bold, serif font. A small dialog box titled "Microsoft Internet Explorer" is overlaid on the page, featuring a yellow warning triangle icon and the text "Before Request" above an "OK" button. The status bar at the bottom shows "Opening page http://192.168.84.165/code/ajax.html..." and includes a speaker icon for audio output.

The screenshot shows a Microsoft Internet Explorer browser window titled "AJAX Example - Microsoft Internet Explorer". The address bar contains the URL "http://192.168.84.165/code/ajax.html". The main content area displays the text "AJAX Example" in a large, bold, serif font. A small error dialog box is overlaid on the page, titled "Microsoft Internet Explorer" and containing a yellow warning triangle icon, the number "1", and an "OK" button. The status bar at the bottom shows "Opening page http://192.168.84.165/code/ajax.html..." and the "Internet" icon.

The screenshot shows a Microsoft Internet Explorer browser window titled "AJAX Example - Microsoft Internet Explorer". The address bar contains the URL "http://192.168.84.165/code/ajax.html". The main content area displays the text "AJAX Example" in a large, bold, serif font. A modal dialog box is overlaid on the page, titled "Microsoft Internet Explorer", featuring a yellow warning triangle icon, the number "2", and an "OK" button. The browser's status bar at the bottom shows "Opening page http://192.168.84.165/code/ajax.html..." and the "Internet" icon.



The screenshot shows a Microsoft Internet Explorer browser window titled "AJAX Example - Microsoft Internet Explorer". The address bar contains the URL "http://192.168.84.165/code/ajax.html". The main content area displays the text "AJAX Example" in a large, bold, serif font. A modal dialog box is overlaid on the page, titled "Microsoft Internet Explorer", featuring a yellow warning triangle icon, the number "3", and an "OK" button. The status bar at the bottom indicates "Opening page http://192.168.84.165/code/ajax.html..." and includes a volume icon.

The screenshot shows a Microsoft Internet Explorer browser window titled "AJAX Example - Microsoft Internet Explorer". The address bar contains the URL "http://192.168.84.165/code/ajax.html". The main content area displays the text "AJAX Example" in a large, bold, serif font. A small error dialog box is overlaid on the page, titled "Microsoft Internet Explorer" and containing a yellow warning triangle icon, the number "4", and an "OK" button. The status bar at the bottom shows "Opening page http://192.168.84.165/code/ajax.html..." and a system tray with a speaker icon.

The image shows a screenshot of a Microsoft Internet Explorer browser window. The title bar reads "AJAX Example - Microsoft Internet Explorer". The menu bar includes "File", "Edit", "View", "Favorites", "Tools", and "Help". The toolbar contains icons for Back, Forward, Stop, Refresh, Home, Search, Favorites, and other utility icons. The address bar shows the URL "http://192.168.84.165/code/ajax.html" with a "Go" button and a "Links" dropdown. The main content area displays the text "AJAX Example" in a large, bold, serif font, followed by "TEST AJAX" in a smaller, monospace font. The status bar at the bottom shows "Done" on the left, "Internet" on the right, and a speaker icon for audio output.

The screenshot shows a Microsoft Internet Explorer browser window titled "AJAX Example - Microsoft Internet Explorer". The address bar contains the URL "http://192.168.84.165/code/ajax.html". The main content area displays the text "AJAX Example" in a large, bold, serif font, followed by "TEST" in a smaller, plain font. A modal dialog box is overlaid on the page, titled "Microsoft Internet Explorer" and containing a yellow warning triangle icon, the text "After Request", and an "OK" button. The status bar at the bottom shows "Opening page http://192.168.84.165/code/ajax.html..." and the "Internet" icon.

The image shows a screenshot of a Microsoft Internet Explorer browser window. The title bar reads "AJAX Example - Microsoft Internet Explorer". The menu bar includes "File", "Edit", "View", "Favorites", "Tools", and "Help". The toolbar contains icons for Back, Forward, Stop, Refresh, Home, Search, Favorites, and other utility icons. The address bar shows the URL "http://192.168.84.165/code/ajax.html" with a "Go" button and a "Links" dropdown. The main content area displays the text "AJAX Example" in a large, bold, serif font, followed by "TEST AJAX" in a smaller, monospace font. The status bar at the bottom shows "Done" on the left and "Internet" on the right, along with a speaker icon.

The screenshot shows a Microsoft Internet Explorer browser window. The title bar reads "AJAX Example - Microsoft Internet Explorer". The address bar contains the URL "http://192.168.84.165/code/ajax.html". The main content area displays "AJAX Example" in a large, bold, serif font, with "TEST AJAX" in a smaller, monospace font below it. An "About Internet Explorer" dialog box is open in the foreground, displaying the Internet Explorer logo and the following text: "Version: 6.0.2900.5512.xpsp_sp3_qfe.130503-0418", "Cipher Strength: 128-bit", "Product ID:76487-031-5242976-22589", "Update Versions:; SP3;", "Based on NCSA Mosaic. NCSA Mosaic(TM); was developed at the National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign.", "Copyright ©1995-2004 Microsoft Corp.", and an "OK" button. The browser's status bar at the bottom shows "Done" on the left and "Internet" on the right, with a speaker icon in the bottom right corner.

XMLHttpRequest with jQuery

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>AJAX jQuery Example</title>
  </head>

  <body>
    <h1>AJAX jQuery Example</h1>
    <div id='insert_here'>
    </div>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js">
    </script>
    <script>
      $.get( "ajax_test.txt", function( data ) {
        $( "#insert_here" ).html( data );
      });
    </script>
  </body>
</html>
```



Browser window showing an AJAX jQuery Example page. The page content includes the title "AJAX jQuery Example" and the text "TEST AJAX".

The Network tab is active, displaying the following request details:

Name Path	Method	Status Text	Type	Initiator	Size Content	Time Latency	Timeline
ajax_jquery.html /code	GET	200 OK	text/ht...	Other	613 B 414 B	4 ms 2 ms	
jquery.min.js ajax.googleapis.com/aja...	GET	304 Not M...	text/ja...	ajax_jquery.h... Parser	33 B 93.7 KB	71 ms 68 ms	
ajax_test.txt /code	GET	304 Not M...	text/pl...	jquery.min.js:4 Script	177 B 10 B	4 ms 2 ms	

Summary: 3 requests | 823 B transferred | 125 ms (load: 123 ms, DOMContentLoaded: 122 ms)

Console: Search Emulation Rendering

Summary: <top frame> | Preserve log



The image shows a screenshot of a Microsoft Internet Explorer browser window. The title bar reads "AJAX jQuery Example - Microsoft Internet Explorer". The menu bar includes "File", "Edit", "View", "Favorites", "Tools", and "Help". The address bar shows the URL "http://192.168.84.165/code/ajax_jquery.html". The main content area displays the text "AJAX jQuery Example" in a large, bold, serif font, followed by "TEST AJAX" in a smaller, monospace font. The status bar at the bottom shows "Done" and "Internet". A speaker icon is visible in the bottom right corner of the browser window.

Asynchronous JavaScript and XML

– AJAX

- Can now make web applications that asynchronously fetch only the required data from the server
 - Can also respond to user input (clicks, form), and potentially load data
- First reference to the term AJAX
 - <https://web.archive.org/web/20050223021343/http://adaptivepath.com/publications/essays/archives/000385.php>



How to Design a Web Application

- Depends on the framework you use
- CGI applications
 - One single file that responds to multiple path infos
 - Multiple files that each respond to their own path
- PHP applications
 - Typically many files that correspond 1-1 with a URL
- ASP applications
 - Classic ASP is the same as PHP



"Natural" PHP code

```
<?php
session_start();
$_SESSION['username'] = 'admin';

$username_param = $_GET['username'];
if ($username_param != $_SESSION['username'])
{
    if ($_SESSION['username'] != 'admin')
    {
        echo "<h1>Sorry, you can only view your own comments.</h1>";
        exit(0);
    }
}

$username = $_SESSION['username'];

?>
```



"Natural" PHP code

```
<h1>CSC 591 Comments</h1>
<h2>Welcome <?php echo $username; ?>
<p>for debugging purposes you are: <span id='userinfo'><?php echo $_SESSION['loggedin2'];
?></span></p>
<h2>Here are the comments</h2>
    <?php
$db = sqlite_open("comments.sqlite");
$query = "select * from comments where username = '" . sqlite_escape_string($username_param) .
"';";
$res = sqlite_query($query, $db);
if ($res)
{
    while ($entry = sqlite_fetch_array($res, SQLITE_ASSOC))
    {
        ?>
        <p><?php echo $entry['comment']; ?>
        <br />- <?php htmlspecialchars($username); ?>
        </p>
        <?php
    }
?>
```



"Natural" PHP code

```
<h2>Make your voice heard!</h2>
<form action="add_comment.php?username=<?php echo urlencode($username); ?>"
method="POST">
<textarea name="comment"></textarea> <br>
<input type="submit" value="Submit" />
</form>
<p>
  <a href="logout.php">Logout</a>
</p>
<?php
}
else {
?>
<h1>Error</h1><p> <?php echo
htmlspecialchars(sqlite_error_string(sqlite_last_error($db))); ?> </p>
<?php
}
?>
```



Spaghetti Code

- How maintainable is this code?
 - Imagine all the files are like this
 - You want to change how comments are stored, giving them extra metadata
 - You must change every single SQL query in every PHP files that touches the comments, as well as all the outputs
- HTML output intermixed with SQL queries intermixed with PHP code



Tight Coupling of URLs to Scripts

- The natural way to design a web application is to map every (valid) URL to a specific script that gets executed
- URLs look like:
 - `http://example.com/add_comment.php`
 - `http://example.com/view_comments.php`
 - `http://example.com/users/view_users.php`
 - `http://example.com/admin/secret.php`
- And map directly to the following file structure
 - `add_comment.php`
 - `view_comments.php`
 - `users/view_users.php`
 - `admin/secret.php`
- Is this necessary?





Who's behind it?

Rails has been conceived, coded, and evangelized by [David Heinemeier Hansson](#) with the kind help of a lot of [contributors](#).

How did it start?

[Basecamp](#), a project-management tool by [37signals/Next Angle](#), was the original Rails application.

Dave Thomas:

"I think Rails may well be the framework to break Ruby into the mainstream"

Real-life apps

[Basecamp](#), [43 Things](#), [Ta-da List](#), [Hieraki](#), [S5 Presents](#), [Snowdevil](#)

What's Ruby?

Ruby is an object-oriented, highly dynamic "scripting" language created by Yukihiro Matsumoto with the intent to maximize the joy of programming [»](#).

Austin Moody:

"I'd rather write a video game in Fortran than have to write another web-based application without Rails."

What databases?

MySQL, PostgreSQL, SQLite, SQL Server, DB2, and Oracle are supported out of the box.

Web servers?

Apache, [lighttpd](#), and Ruby's own WEBrick are the primary targets using [FastCGI](#), [mod_ruby](#), and CGI.

Michael Koziarski:

"Rails is perhaps the most productive web development environment I've ever used."

Where to host?

[TextDrive](#) is the official Ruby on Rails host and offers [fantastic and cheap plans](#) where 50% of the proceeds go to Rails development!

Rails is a full-stack, open-source web framework in Ruby for writing real-world applications **with joy and less code** than most frameworks spend doing XML sit-ups

Being a full-stack framework means that all layers are built to work seamlessly together. That way you [Don't Repeat Yourself](#) (DRY) and you can use a single language from top to bottom. Everything from templates to control flow to business logic is written in Ruby—the language of love for [industry heavy-weights](#).

In striving for DRY compliance, Rails shuns configuration files and annotations in favor of reflection and run-time extensions.

This means the end of XML files telling a story that has already been told in code. It means no compilation phase: Make a change, see it work. Meta-data is an implementation detail left for the framework to handle.

- [Ruby on Rails](#) | [Screencasts](#) | [Download](#) | [Documentation](#) | [Weblog](#) | [Community](#) | [Source](#)

Get started with Ruby on Rails

```
class WeblogController < ActionController::Base
  include WeblogHelper

  def index
    @render_text = "hello world!"
  end

  def forside
    @render_text = "adding new a"
  end
end
```

Show, don't tell!
[10m setup video](#) (22MB)
[More in Rails Academy](#)



Hype and philosophy
[RUC video](#) (2h/160MB)
[RubyConf '04](#) (1h/56MB)



New to Ruby on Rails?
[Starting with Ruby](#)
[Beginning with Rails](#)



Download Rails
[Using RubyGems](#)



Speed up your app
[Production Envs](#)

The frameworks of Rails

Rails is composed of three sub-frameworks in addition to all the ties that makes them run so well together. The three frameworks are...

Active Record

Connects business objects and database tables to create a persistable domain model where logic and data is presented in one wrapping.

Action Pack

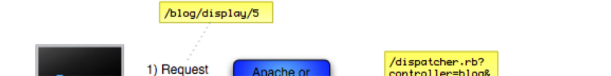
Routes incoming requests through controllers with one method per action and lets view rendering happen using Ruby templates.

Action Mailer

Consolidates code for sending out forgotten passwords and invoices for billing in easy-to-test email service layers on top of smtp or sendmail.

Flowing on the Rails

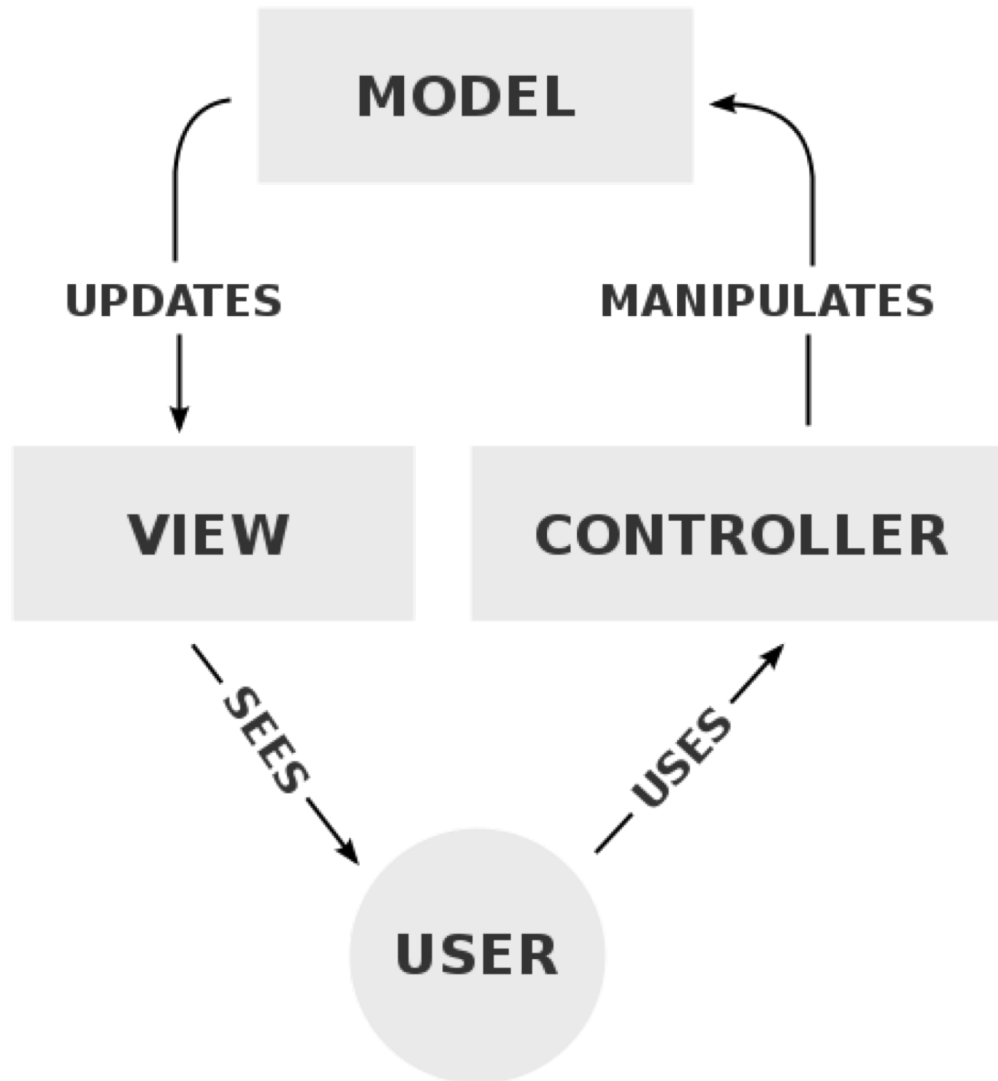
Most of the time, all the frameworks of Rails are invoked on each request in order to produce a response. The flow is as follows...



Model-View-Controller

- User Interface design framework
 - A way to separate the concerns of a GUI
 - Originally created in the early '90s
- Popularized by Ruby on Rails to structure the server-side code of web applications





Separation of Concerns

- Model
 - Handles all the "business logic" of the application
 - Stores the application state
- View
 - Responsible for generating a view for the user of the data from the model
 - Usually a simple templating system to display the data from the model
- Controller
 - Responsible for taking input from the user, fetching the correct data from the model, then calling the correct view to display the data
 - Should be very simple



Object Relational Mapping

- As a programmer, you don't need to worry about the database or "SQL" language
- Rails (ActiveRecord)
 - `user = User.create(name: "David", occupation: "Code Artist")`
 - `david = User.find_by(name: 'David')`
 - `david.destroy()`
 - `Article.where('id > 10').limit(20).order('id asc')`



Routing

- Define a mapping between URLs and server-side functions
- Also define parameters that get passed to the function from the URL
- Rails example:

```
class BooksController < ApplicationController
  def update
    @book = Book.find(params[:id])
    if @book.update(book_params)
      redirect_to(@book)
    else
      render "edit"
    end
  end
end
```



Routing

```
class BooksController < ApplicationController
  def index
    @books = Book.all
  end
end
```



Templating

- Define the view as a simplified language
 - Input: well-defined variables or dictionaries
 - Output: HTML (or JSON or XML, ...)
- Ruby on Rails uses ERB:

```
<h1>Listing Books</h1>
```

```
...
```

```
<% @books.each do |book| %>
```

```
  <tr>
```

```
    <td><%= book.title %></td>
```

```
    <td><%= book.content %></td>
```

```
    <td><%= link_to "Show", book %></td>
```

```
    <td><%= link_to "Edit", edit_book_path(book) %></td>
```

```
    <td><%= link_to "Remove", book, method: :delete, data: { confirm: "Are you  
sure?" } %></td>
```

```
  </tr>
```

```
<% end %>
```

```
...
```

```
<%= link_to "New book", new_book_path %>
```



Flask & Jekyll

- Similar to Ruby on Rails, but in Python
- Very nice tutorial if you want to build your own (complicated) site
 - <https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world>
- Plain text -> static website
 - Jekyll: <https://jekyllrb.com/>
 - What I use for kapravelos.com
 - Originally developed for Github Pages
 - Easy to host
- Write your own website
 - Google App Engine with Flask ([link](#))
 - Github Pages ([link](#))

