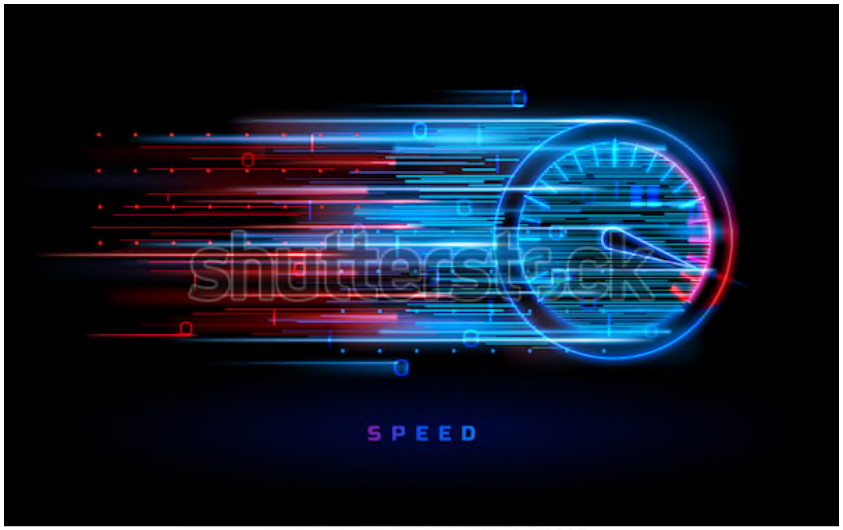
FIT5032 (Suzhou)

Internet Applications Development

Week 7: Web Optimisations & Evolution of ASP.NET CORE

Murray Mount / ABM Russel



www.shutterstock.com · 1334759021

Unit Topics

Week	Activities	Assessment
0		No formal assessment or activities are undertaken in week 0
1A	Intro to Web development and ASP.NET	Note: Studio classes commence in week 1
1B	The front end, user experience, accessibility and ASP.NET Scaffolding	
2	Introduction to C# & Version Control	
3	Entity Framework	
4	Fundamentals of Client side Javascript	Studio assessment task 1 due
5A	Validation	
5B	Security and Identity	
6	Sending Email, File Upload and Signal R	Studio assessment task 2 due
7	Web Optimisations & Evolution of ASP.NET CORE	
8A	Modern JavaScript Web Development Approaches	
8B	Testing and Deployment in Cloud	Final Portfolio and Learning Summary due
9	Review & Revision	
10	Examination period	
		LINK to Assessment Policy:http://policy.monash.edu.au/policy-bank/academic/education/assessment/assessment-in-coursework-

Today

- Recap: Sending Email, File Upload, SignalR
- Web Optimisations
  - Reducing HTTP Connections
  - Reducing File sizes
  - Caching
  - Perceived improvements
- Evolution of ASP.NET Core

Recap: Sending Email, File Upload, SignalR

Sending Email Example

```
using System.Net.Mail;


….
var body = "<p>Email From: {0} ({1})</p><p>Message:</p><p>{2}</p>";
var message = new MailMessage();
message.To.Add(new MailAddress("abm.russel@monash.edu"));
// replace with valid value
message.From = new MailAddress("abm.russel@monash.edu");
// replace with valid value
message.Subject = "Your email subject";
message.Body = string.Format(body, model.FromName, model.FromEmail,
model.Message);
message.IsBodyHtml = true;
```

Change examples to your email

Model for email example

```
using System.ComponentModel.DataAnnotations;
using System.Web;
namespace Week8Email.Models
{
    public class EmailFormModel
    {
        [Required, Display(Name = "Your name")]
        public string FromName { get; set; }
        [Required, Display(Name = "Your email"), EmailAddress]
        public string FromEmail { get; set; }
        [Required]
        public string Message { get; set; }
     }
}
```

SmtpClient send method

- *SendMailAsync* method of the SmptClient class, which takes argument, message which includes the:

From
To
Subject
Message Text

MailMessage Object

MailAddress constructor parameters:

email address and a display name

email will be sent to the users email address

firstname and surname will be displayed in the To field of the email client when the email is received.

Attachments

- Use the Attachment class
  a collection of the MailMessage object.

```
Attachment newAttach = new Attachment(Server.MapPath("~/
    MyFile.txt"));
 newMsg.Attachments.Add(newAttach);
```

Adds "MyFile.txt", located in the root directory of web application, as an attachment to the email.

.

Attachments

- Can add an attachment sent by the user from a form

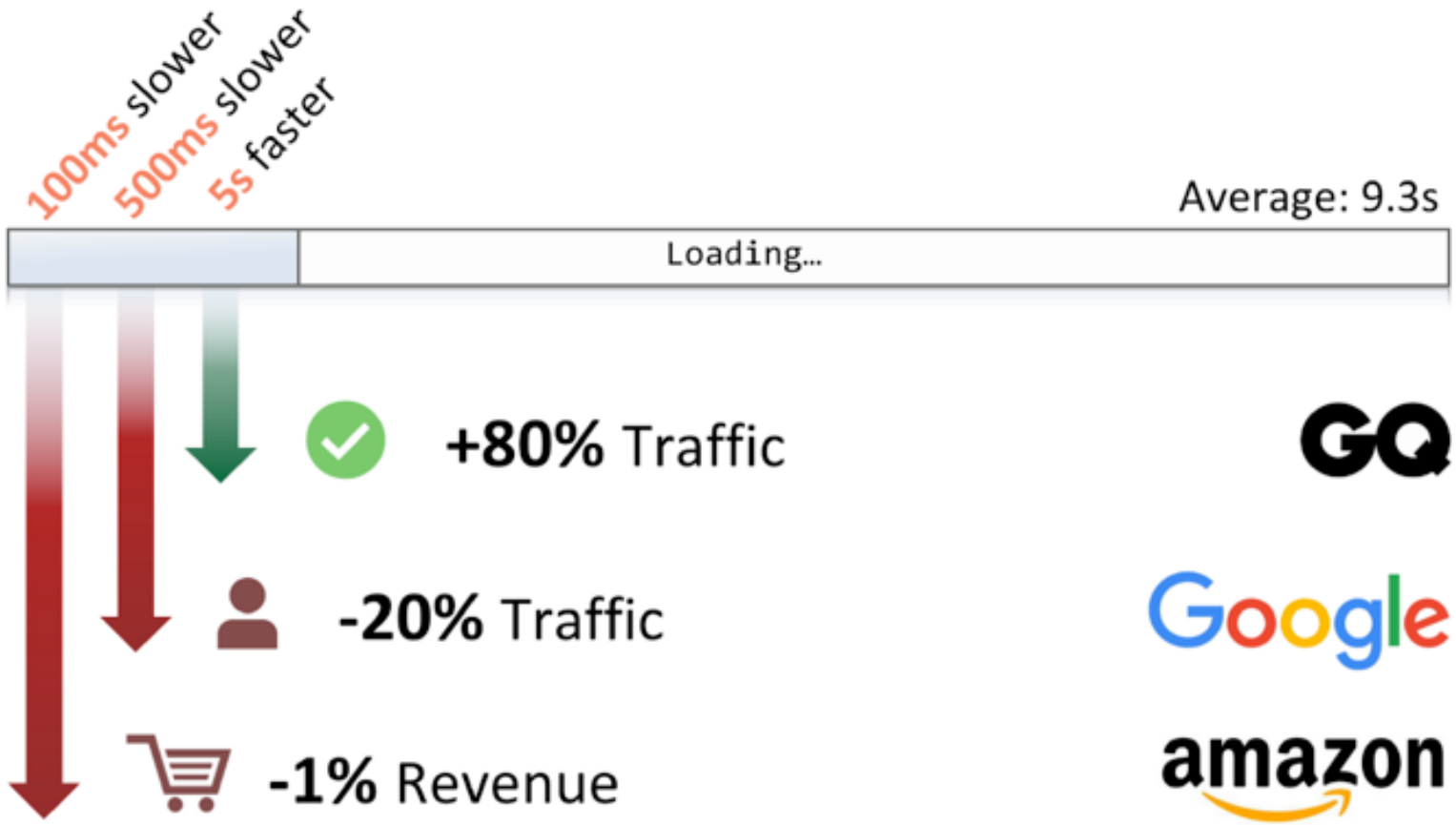
```
if (model.Upload != null && model.Upload.ContentLength > 0)
            {
                message.Attachments.Add(new
Attachment(model.Upload.InputStream,
System.IO.Path.GetFileName(model.Upload.FileName)));
            }
```

Update model with

- public HttpPostedFileBase Upload { get; set; }

Web Optimisations

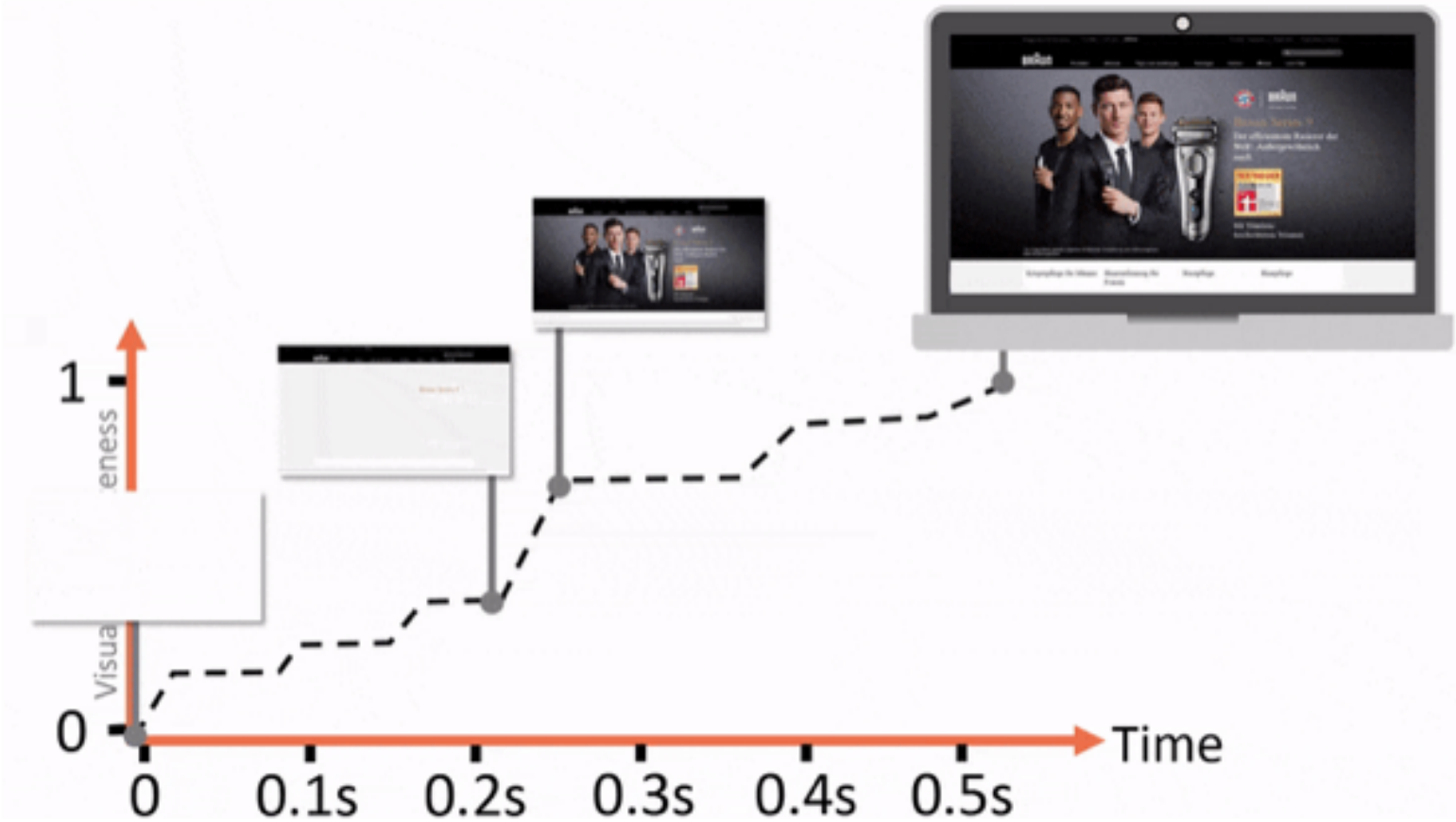
Why is page speed important?



Source: https://medium.baqend.com/the-technology-behind-fast-websites-2638196fa60a

Mobile users

When does a website feel fast?



First Meaningful Paint and **Speed Index** are performance measures that represent user-perceived page speed.

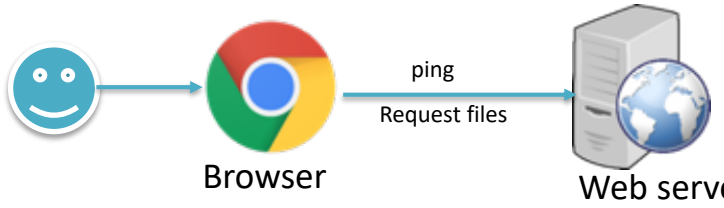
Optimization Summary

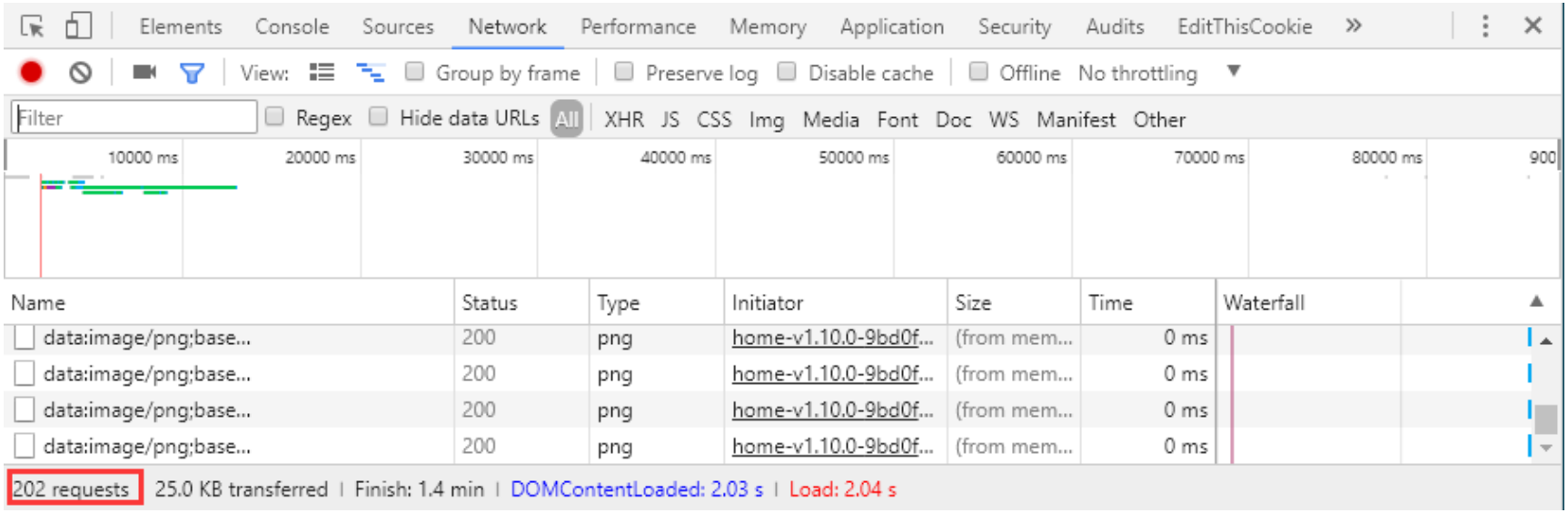
- There are 3+1 main optimisation techniques
- **Reducing HTTP Connections**
    - Each connection and request is time consuming and takes resources
- **Reducing File sizes**
    - Larger file sizes take more bandwidth (and time) to down load
- **Caching**
    - Can reduce the need to request a resource that has already been down loaded
- **Perceived Improvements**
    - Some changes can be made that does not improve absolute performance, but makes the user believe there is an improvement (due to lower latencies and quicker perceived response time)

Optimization Summary

1. Reduce HTTP Requests
2. Image Optimization
3. Minify CSS and JavaScript
4. Render Blocking Resources (CSS + JS + Web font)
5. Reduce Latency with a CDN + TTFB
6. Caching
7. Prefetch and Preconnect
8. Hotlink protection
9. Fix 404 Errors
10. Database Optimization

1. Reduce HTTP request





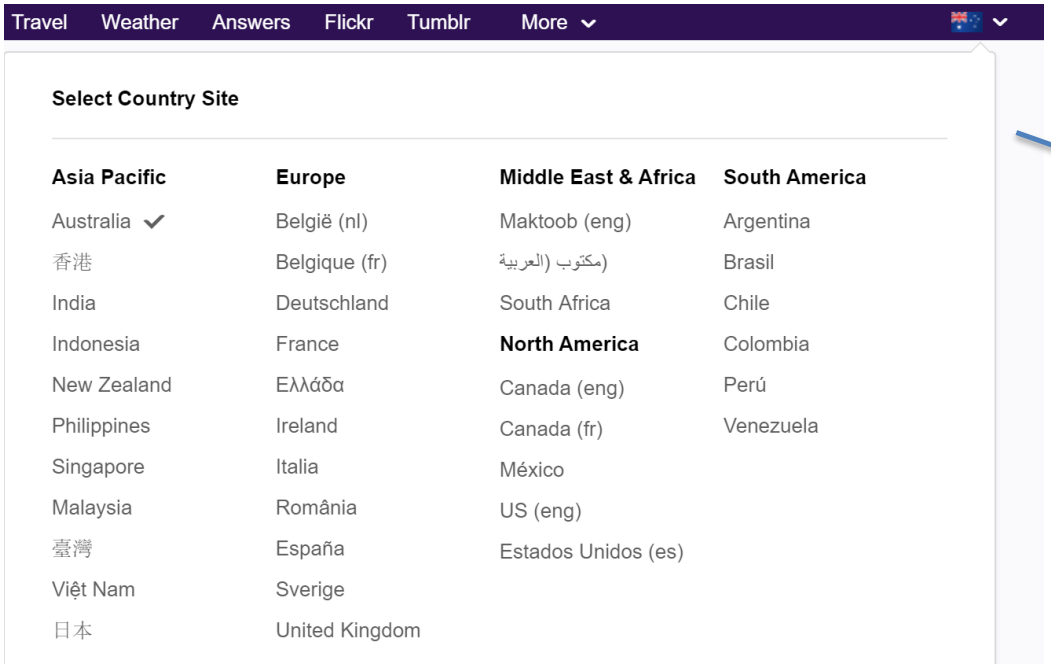
Optimize HTTP request by:

- Combing images
- Reduce the size of files
- Making JavaScript asynchronous

Optimization Summary

1. Reduce HTTP Requests
2. Image Optimization
3. Minify CSS and JavaScript
4. Render Blocking Resources (CSS + JS + Web font)
5. Reduce Latency with a CDN + TTFB
6. Caching
7. Prefetch and Preconnect
8. Hotlink protection
9. Fix 404 Errors
10. Database Optimization

2.1 Image Optimization



235px X 60px

```
#IntlSelector .flag-au {                (index):115
   background-position:▶ -148px -15px;
}
```

- Make use of CSS to replace icons instead of small sized images
- If small icons is unavoidable, merge the icons in one picture, access each picture by limiting the display boundary of the picture through CSS
- Reason: retrieving each picture needs to initialise a HTTP request each time
- Picture obtained Yahoo representing each country

2.2 Image compression – Types

Right picture shows the image after deep compression ↑

- There are various of image types, most common images compression types are:
- PNG – Lossless compression
  - (The original data is perfectly reconstructed from the compressed data)
  - Suitable for icons, pictures with no background
- JPEG – Lossy compression
  - (Ideals for balance between the storages size and image quality)

2.3 Image compression – Responsive Images

Srcset & Size attributes

How you load the images normally:

```
<img src=“monash.jpg" alt="responsive images car" width=“800" height=“600">
```

With “Srcset” attribute: (Browser will decide which image it choose, w means the width of the viewport)

```
<img src=“monash.jpg" alt="responsive images car" srcset=“monash-160.jpg 160w, monash-320.jpg 320w,
monash-640.jpg 640w, monash-1280.jpg 1280w">
```

Usually used with conjunction of the “size” attribute if want to force browser to use the image under certain width

`(max-width: 480px) 100vw` means if the browser is lower than 480px, then show 100% of its size

```
<img src="responsive-images-car.jpg" alt="responsive images car" srcset="responsive-images-car-
160.jpg 160w, responsive-images-car-320.jpg 320w, responsive-images-car-640.jpg 640w,
responsive-images-car-1280.jpg 1280w" sizes="(max-width: 480px) 100vw, (max-width: 900px) 33vw,
254px">
```

Optimization Summary

1. Reduce HTTP Requests
2. Image Optimization
3. Minify CSS and JavaScript
4. Render Blocking Resources (CSS + JS + Web font)
5. Reduce Latency with a CDN + TTFB
6. Caching
7. Prefetch and Preconnect
8. Hotlink protection
9. Fix 404 Errors
10. Database Optimization

3.1 Minify CSS and JavaScript

https://javascript-minifier.com/

C#

```
if (productExists)
    GetOrders();
```

Result:

C#

```
if (productExists) {
    GetOrders();
}
```

Input JavaScript

```
function myFunction() {
    var x = document.getElementById("demo");
    x.style.fontSize = "25px";
    x.style.color = "red";
}
```

Minified Output

```
function myFunction(){var e=document.getElementById("demo");e.style.fontSize="25px",e.style.color="red"}
```

Removing unnecessary characters from your HTML

- White space characters
- New line characters
- Comments
- Block delimiters

3.2 Uglify CSS and JavaScript

Changing variables and functions with long names into shortnames

- Can act as an obfuscator, but is just a simple replacement

3.3 Bundle CSS and JavaScript

Bundler combines multiple fines into a single file

- Often used with minify and uglify

Optimization Summary

1. Reduce HTTP Requests
2. Image Optimization
3. Minify CSS and JavaScript
4. Render Blocking Resources (CSS + JS + Web font)
5. Reduce Latency with a CDN + TTFB
6. Caching
7. Prefetch and Preconnect
8. Hotlink protection
9. Fix 404 Errors
10. Database Optimization

4.1 Render Blocking Resources (CSS + JS + Web font)

Render Blocking Resources
- HTML
- CSS
- JavaScript
- Web fonts

Images aren't render blocking

CSS
Ultimate Goal is get it to the client as soon and as quickly as possible to optimize the time to first render. So it is loaded in <head> section
But,
Some of the css style is not that critical, such as when we only use the css when printing

CSS "media types" and "media queries" allow us to address these use cases:

```
<link href="style.css" rel="stylesheet">
<link href="print.css" rel="stylesheet" media="print">
<link href="other.css" rel="stylesheet" media="(min-width: 40em)">
<link href="portrait.css" rel="stylesheet" media="orientation:portrait">
```

4.2 Render Blocking Resources (CSS + JS + Web font)

Additional way to optimize your CSS

- Lessen the amount of CSS files (concatenate your CSS files into one file)
- Minify Your CSS (remove extra spaces, characters, comments, etc)
- Use less CSS overall

JavaScript

Move your scripts to the bottom of the page right before your </body> tag.

Reason: Putting JavaScript in the top/ middle of your body will stop the browser rendering the rest of the content before JavaScript is loaded

- Use the async or defer JavaScript to avoid render blocking.

```
<script async src="foobar.js"></script>
```

*Does not mean the webpage is fully loaded before JavaScript is executed

4.3 Render Blocking Resources (CSS + JS + Web font)

Deferring JavaScript
- Ultimate goal is loading or parsing JavaScript only after page content has loaded

```
<script type="text/javascript">
    function downloadJSAtOnload() {
        var element = document.createElement("script");
        element.src = "filesYouWishToLoad.js";
        document.body.appendChild(element);
    }
    if (window.addEventListener)
        window.addEventListener("load", downloadJSAtOnload, false);
    else if (window.attachEvent)
        window.attachEvent("onload", downloadJSAtOnload);
    else window.onload = downloadJSAtOnload;
</script>
```

Best to separate JS into two Groups
- Essential JavaScript files need to load (jQuery Library, etc)
- Non essential JavaScript files (Click event etc)

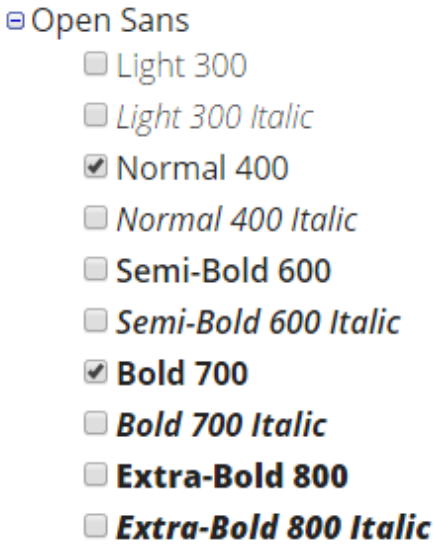
4.4 Render Blocking Resources (CSS + JS + Web font)

Web Font

- The disadvantages of web fonts are that they add extra HTTP requests to external resources.

Recommendations:

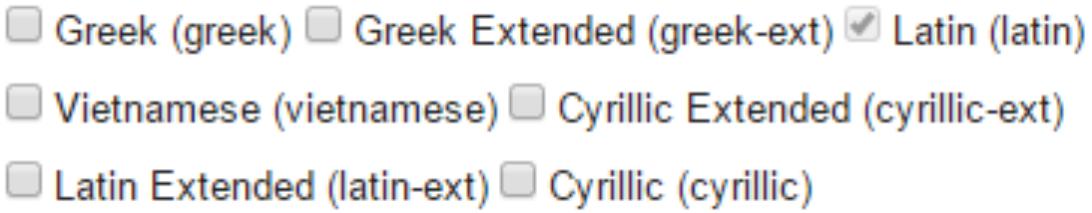
- Prioritize font based on browser support
  - Serve WOFF 2.0 variant to browsers that support it.
  - **Serve WOFF variant to the majority of browsers.**
  - Serve TTF variant to old Android (below 4.4) browsers.
  - Serve EOT variant to old IE (below IE9) browsers.
- Choose only the styles you need



4.5 Render Blocking Resources (CSS + JS + Web font)

Recommendations (continues):

- Keep character sets down to a minimum
  - Unless you are dealing with multiple languages



- Host fonts locally or prefetch
  - Reduce HTTP Request

*Check whether it is open source license

Optimization Summary

1. Reduce HTTP Requests
2. Image Optimization
3. Minify CSS and JavaScript
4. Render Blocking Resources (CSS + JS + Web font)
5. Reduce Latency with a CDN + TTFB
6. Caching
7. Prefetch and Preconnect
8. Hotlink protection
9. Fix 404 Errors
10. Database Optimization

5.1 Reduce Latency with a CDN + TTFB

POP = points of presence

Imagine your hosting server is in Dallas.

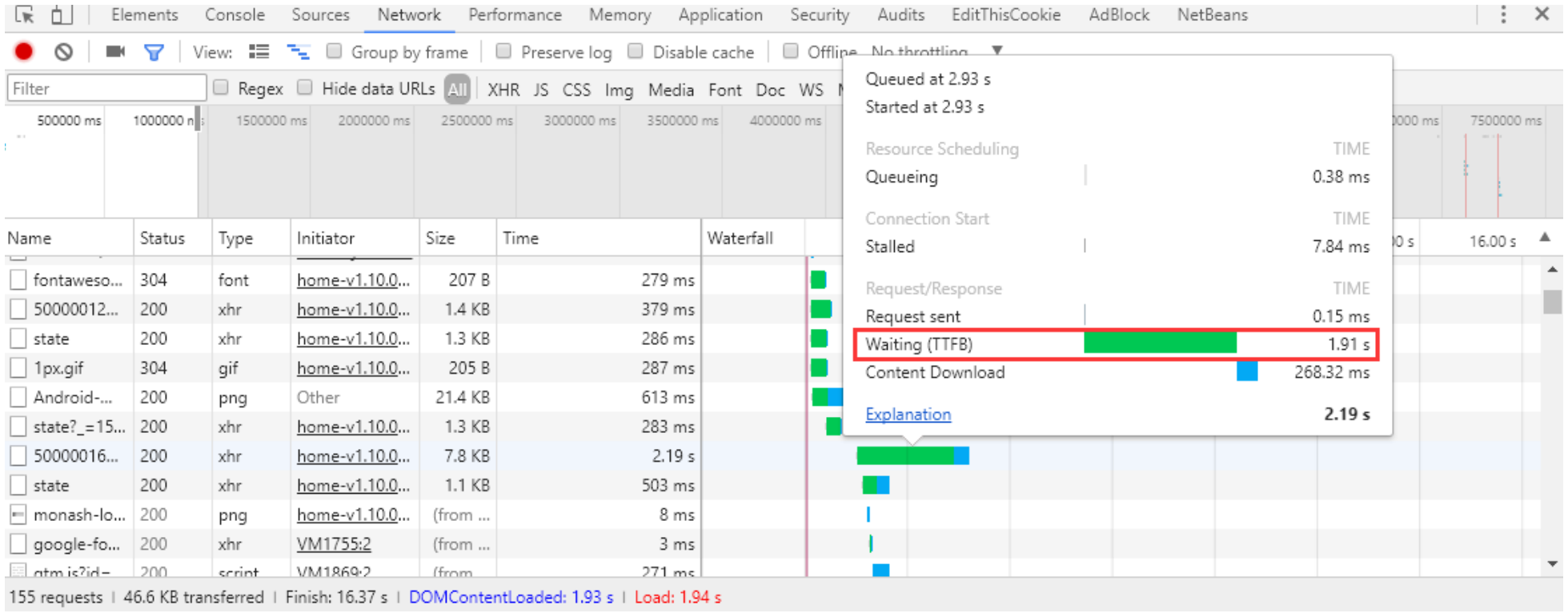
CDN
A **content delivery network** (**CDN**) is a system of distributed servers (**network**) that deliver pages and other Web **content** to a user, based on the geographic locations of the user, the origin of the webpage and the **content delivery** server.

5.2 Reduce Latency with a CDN + TTFB

Server (POP) Location	No CDN RTT (ms)	KeyCDN RTT (ms)	Difference %
New York, US	36.908	18.096	– 50.97%
Dallas, US	0.751	1.138	+ 51.53%
San Francisco, US	39.645	18.900	– 52.33%
Frankfurt, DE	123.072	3.734	– 96.97%
London, UK	127.555	4.548	– 96.43%
Paris, FR	112.417	1.689	– 98.5%
Amsterdam, NL	118.418	10.364	– 91.25%
Singapore, SG	202.682	2.002	– 99.01%
Sydney, AU	191.848	0.705	– 99.63%
Tokyo, JP	130.804	3.379	– 97.42%

*RTT = Round-trip time

5.3 Reduce Latency with a CDN + TTFB



TTFB

- Time to first byte (TTFB) is the measurement of the responsiveness of a web server

HTTP request time + Process request time + HTTP response time

5.4 Reduce Latency with a CDN + TTFB

Factors influence TTFB

- Database design
- Webserver Configuration
- Hardware limit of the original server

Mostly done by web server administrator

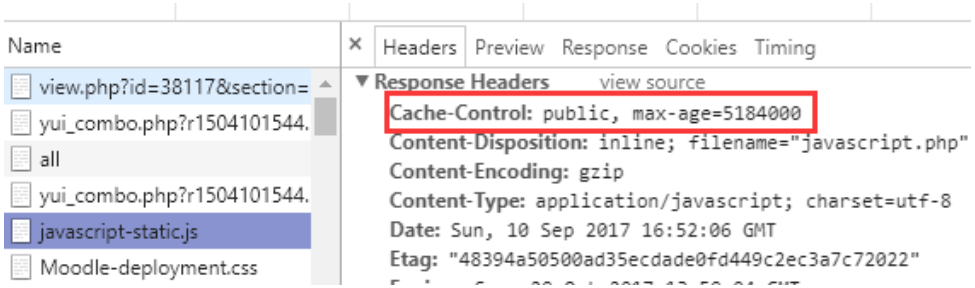
Optimization Summary

1. Reduce HTTP Requests
2. Image Optimization
3. Minify CSS and JavaScript
4. Render Blocking Resources (CSS + JS + Web font)
5. Reduce Latency with a CDN + TTFB
6. Caching
7. Prefetch and Preconnect
8. Hotlink protection
9. Fix 404 Errors
10. Database Optimization

6.1 Caching

Making use of the browser cache for assets that are rarely changing

- Catch control
- ETag

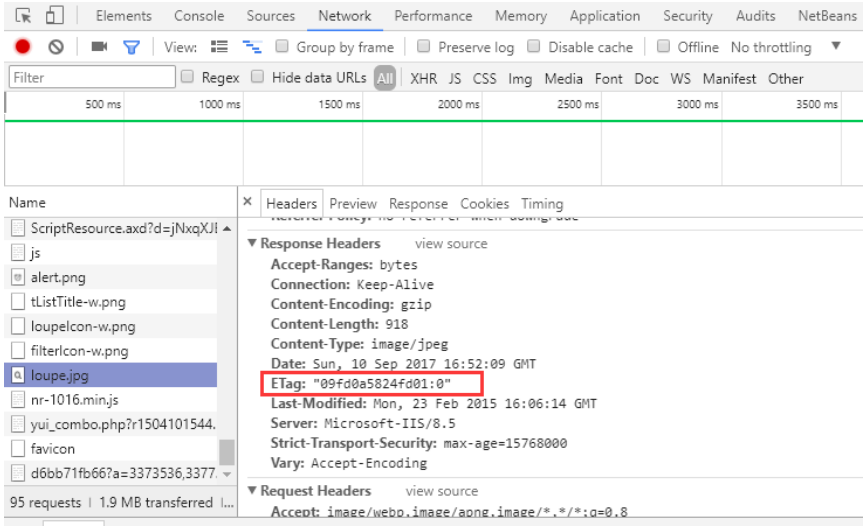


Catch control

Each resource can define its caching policy via the Cache-Control HTTP header.
Cache-Control directives control who can cache the response, under which conditions, and for how long.
Catch control using header in asp.net
Under web.config

```
<configuration>
   <system.webServer>
      <staticContent>
         <clientCache cacheControlCustom="public"
                  cacheControlMaxAge="12:00:00" cacheControlMode="UseMaxAge" />
      </staticContent>
   </system.webServer>
</configuration>
```

6.2 Caching



ETag

- The server uses the ETag HTTP header to communicate a validation token.
- The validation token enables efficient resource update checks: no data is transferred if the resource has not changed.

Optimization Summary

1. Reduce HTTP Requests
2. Image Optimization
3. Minify CSS and JavaScript
4. Render Blocking Resources (CSS + JS + Web font)
5. Reduce Latency with a CDN + TTFB
6. Caching
7. Prefetch and Preconnect
8. Hotlink protection
9. Fix 404 Errors
10. Database Optimization

7. Prefetch and Preconnect

DNS Prefetching

- DNS prefetching is an attempt to resolve domain names before a user tries to follow a link

```
<link rel="dns-prefetch" href="//www.example.com">
```

Preconnect

- **Allows the browser to set up early connections** before an HTTP request is actually sent to the server.
  Connections such as DNS Lookup, TCP Handshake, and TLS negotiation can be initiated beforehand, **eliminating roundtrip latency** for those connections and saving time for users.

```
<link href='https://example.com' rel='preconnect' crossorigin>
```

Optimization Summary

1. Reduce HTTP Requests
2. Image Optimization
3. Minify CSS and JavaScript
4. Render Blocking Resources (CSS + JS + Web font)
5. Reduce Latency with a CDN + TTFB
6. Caching
7. Prefetch and Preconnect
8. Hotlink protection
9. Fix 404 Errors
10. Database Optimization

8. Hotlink protection

Restricting HTTP referrers in order to prevent others from embedding your assets on other websites. (For example: cannot load your website image under other website)

Hotlink protection will save bandwidth by **prohibiting other sites from displaying your images/ contents**.

Optimization Summary

1. Reduce HTTP Requests
2. Image Optimization
3. Minify CSS and JavaScript
4. Render Blocking Resources (CSS + JS + Web font)
5. Reduce Latency with a CDN + TTFB
6. Caching
7. Prefetch and Preconnect
8. Hotlink protection
9. Fix 404 Errors
10. Database Optimization

9. Fix 404 Errors

Depends on the platforms, some of the 404 errors uses a large amount of memory

- For example, Drupal uses 60 – 100 mb memory to deliver a 404

- Recommendation:

  Make use of external tools to run through the website to see whether it has broken links:
  Online Broken Link Checker
  http://www.brokenlinkcheck.com/

  Or
  Screaming Frog
  https://www.screamingfrog.co.uk/

Optimization Summary

1. Reduce HTTP Requests
2. Image Optimization
3. Minify CSS and JavaScript
4. Render Blocking Resources (CSS + JS + Web font)
5. Reduce Latency with a CDN + TTFB
6. Caching
7. Prefetch and Preconnect
8. Hotlink protection
9. Fix 404 Errors
10. Database Optimization

10. Database Optimization

- Cleaning out old unused tables or creating indexes for faster access there are always things that can be optimized.
  - E.g. Decide which join give best performance Hash join / Merge join / Nested Loops
  - Whether to create a view
  - Some of the reporting functions can be performed overnight
- If website intention is built on reporting and business intelligence, make use of star schema instead of traditional normalised tables.

What is web optimisation?

What is web optimisation?

A. Ways to increase the browser speed

B. Ways to increase the server speed

C. Ways to increase the web application speed

D. All the answers (except none)

E. None of the answers

Which of the following will NOT reduce the number of http requests?

Which of the following will NOT reduce the number of http requests?

A. Combining icons into one large image

B. Combining JavaScript files into one large file

C. Combining CSS files into one large file

D. Minifying JavaScript files

E. Combining a banner animation into one file

Which of the following will NOT reduce the size (in bytes) of an image file?

Which of the following will NOT reduce the size (in bytes) of an image file?

A. Reducing the horizontal and vertical size

B. Reducing the number of colours

C. Applying Lossy (e.g. jpeg) compression

D. Applying Lossless (e.g. png) compression

E. Zipping up a fully compressed png file

What does minify do?

What does minify do?

A. Change large variable and function names to smaller names

B. Combines multiple files into a single file

C. Removes while space characters (tabs and line breaks)

D. All the answers (except none)

E. None of the answers

What does uglify do?

What does uglify do?

A. Change large variable and function names to smaller names

B. Combines multiple files into a single file

C. Removes while space characters (tabs and line breaks)

D. All the answers (except none)

E. None of the answers

Which resources are NOT render blocking?

Which resources are NOT render blocking?

A. JavaScript

B. CSS

C. HTML

D. Images

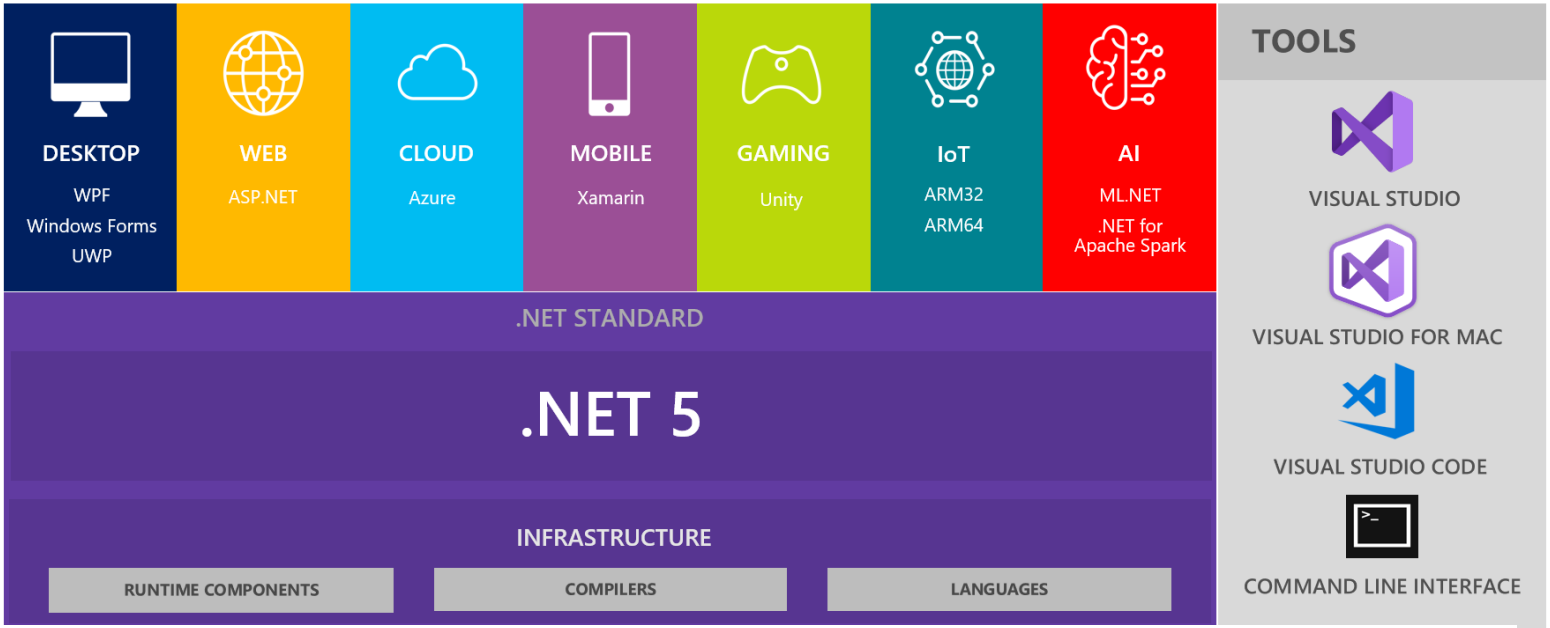
E. Web Fonts

Evolution of ASP.NET Core

What is .NET CORE

- A cross-platform .NET optimised for open source development that runs on Windows, Mac, and Linux.
- Unlike Mono (a 3rd party implementation of .NET framework on Linux/Android/iOs)
- Kestrel

Evolution



.NET Schedule

https://devblogs.microsoft.com/dotnet/introducing-net-5/

Some .NET Core Characteristics

- Flexible deployment
- Cross-platform
- Command-line tools
- Compatible
- Open source
- Supported by Microsoft
- Modular
- Can be distributed in several ways
- Doesn’t support all frameworks
- Performance as priority
- Any IDE

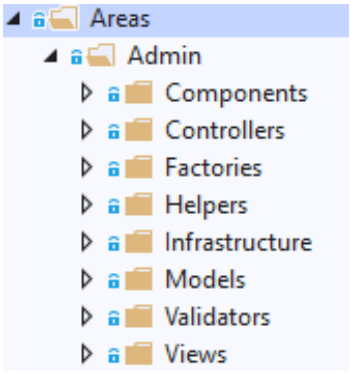
What is ASP.NET CORE

- ASP.NET is a web development framework for .NET
- ASP.NET CORE is a brand new cross-platform web framework built with .NET Core framework.
- A complete rewrite of the ASP.NET framework and works with both .NET Core and .NET Framework.

Some new features of ASP.NET CORE

- The new format of csproj file in .NET Core
- Namespace update (System.Web with Microsoft.AspNetCore)
- Startup.cs file instead of using global.asax
- IoC (Inversion of Control) containers to inject dependencies and represented by the IServiceProvider interface. Services are installed in the app in the Startup.ConfigureServices() method.
- Key-value pairs set by configuration providers in appsettings.json file. Use NuGet package System.Configuration.ConfigurationManager

Some new features of ASP.NET CORE



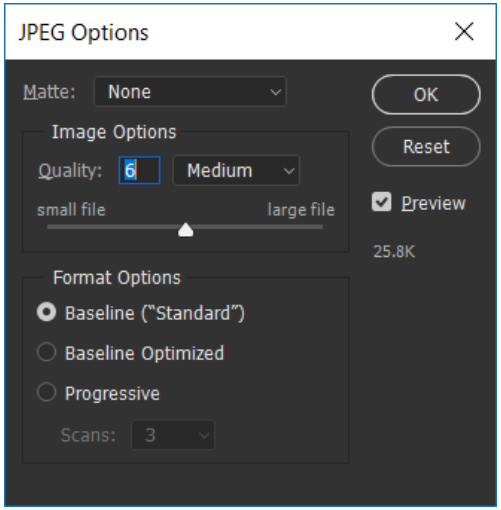
- Static content in wwwroot
- EntityFramework to EF Core: System.Data.Entity namespace is replaced by Microsoft.EntityFrameworkCore;
- Built-in data protection feature.

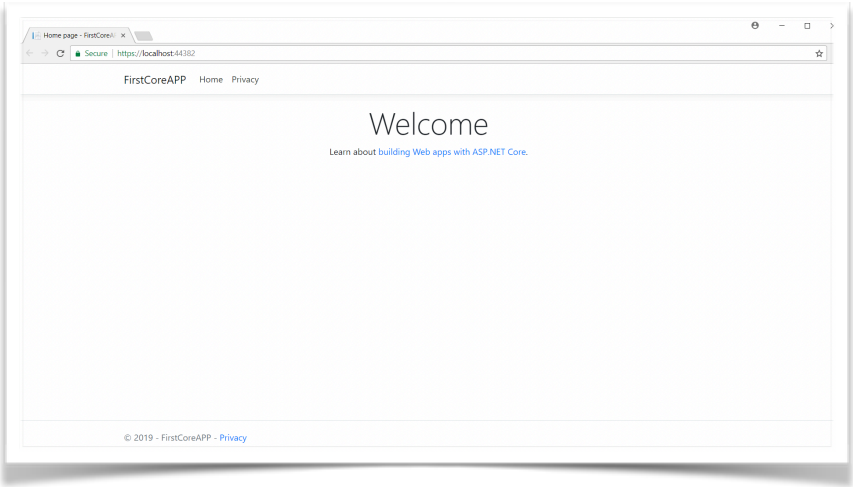
Lecture Summary

- Recap: Sending Email, File Upload, SignalR
- Web Optimisations
  - Reducing HTTP Connections
  - Reducing File sizes
  - Caching
  - Perceived improvements
- Evolution of ASP.NET Core

Week 9 Studio Overview

- Web Optimisation
- ASP.NET CORE





Next week: Modern JavaScript Web Development Approaches

- Modern JavaScript Web Development Approaches