

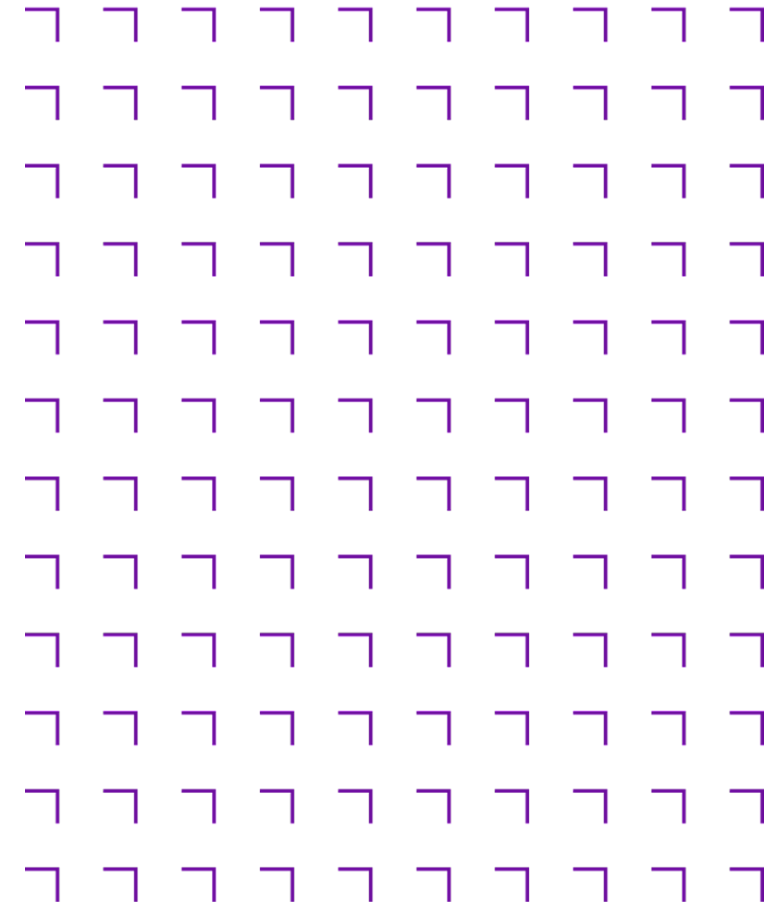
Front-End Web Development

Unit 15: Other Frameworks, SEO, Web Security, Performance

Course Outline



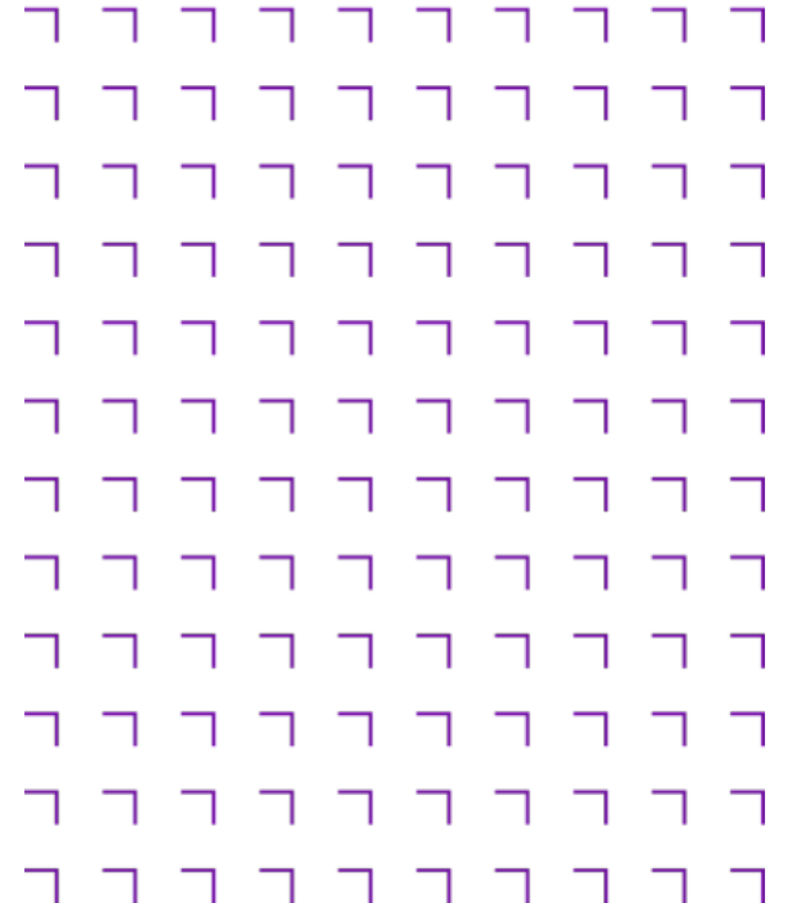
1. Getting Started
2. HTML - Structuring the Web
3. CSS - Styling the Web
4. JavaScript - Dynamic client-side scripting
5. CSS - Making Layouts
6. Introduction to Websites/Web Applications
7. CSS – Advanced
8. Reviewing Progress
9. JavaScript - Modifying the Document Object Model (DOM)
10. Dynamic HTML
11. Web Forms - Working with user data
12. JavaScript - Advanced
13. Building a Web Application with JavaScript
14. Introduction to CSS Frameworks – Bootstrap
- 15. Other Frameworks, SEO, Web security, Performance**
16. Walkthrough project



Course Learning Outcomes



- Competently write HTML and CSS code
- Create web page layouts according to requirements using styles
- Add interactivity to a web page with JavaScript
- Access and display third-party data on the web page
- Leverage Bootstrap and Static Site Generator



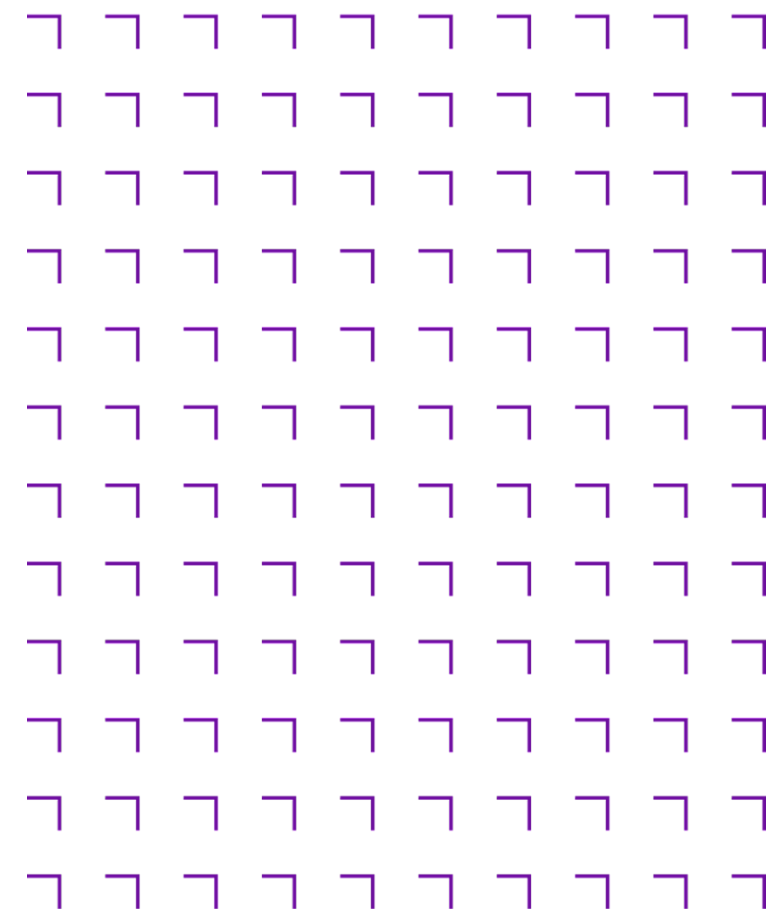
- Final Project - 100% of the grade

- Design and Build functioning Website using HTML5, CSS (including Bootstrap), JavaScript (browser only)

- ✓ Code will be managed in GitHub
 - ✓ Website will be deployed to GitHub Pages
 - ✓ All code to follow best practice and be documented

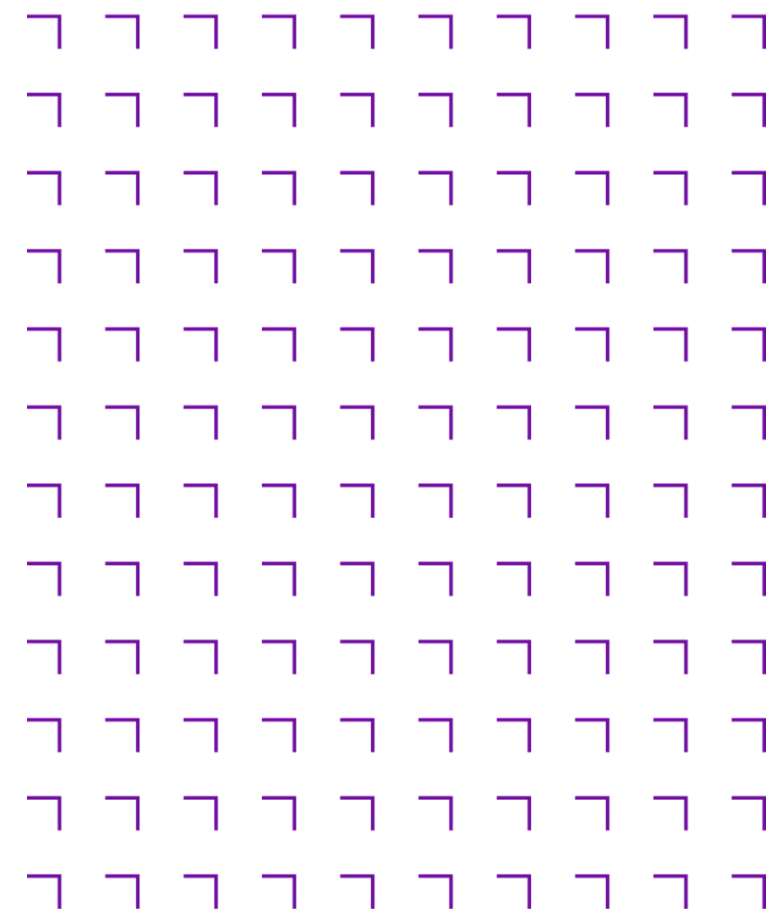
- Details and How-To-Guide are available on the course page under the section called Assessments

Assessment





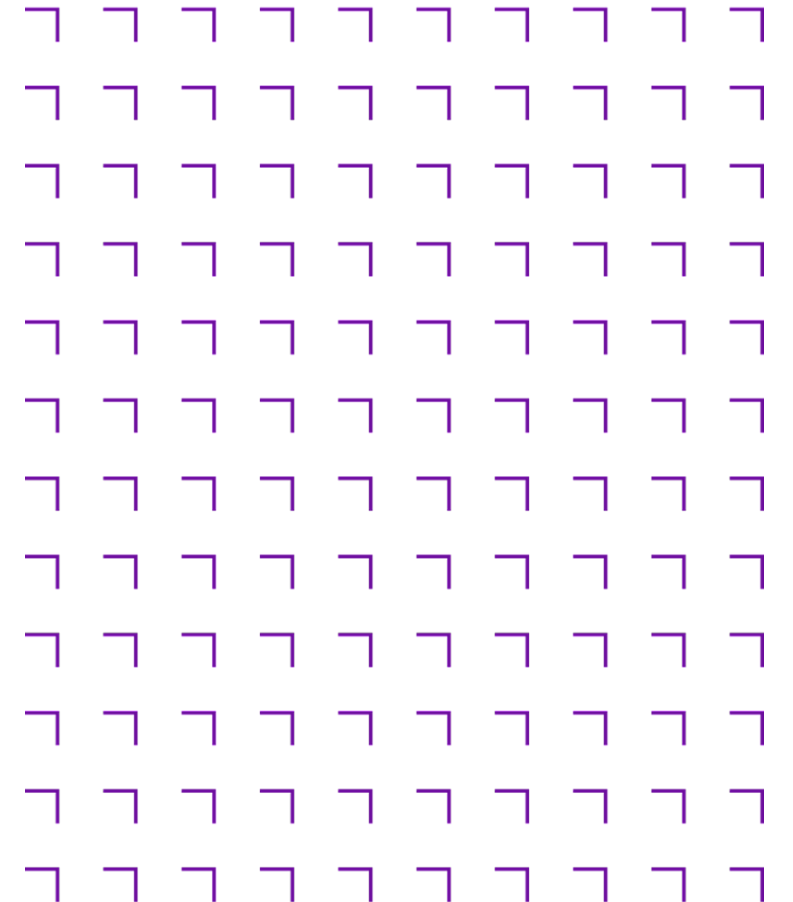
In This Unit



15. Other Frameworks, SEO, Web Security, Performance

Title
Other Web Frameworks
SEO and Web Security
Web Performance

Other Web Frameworks



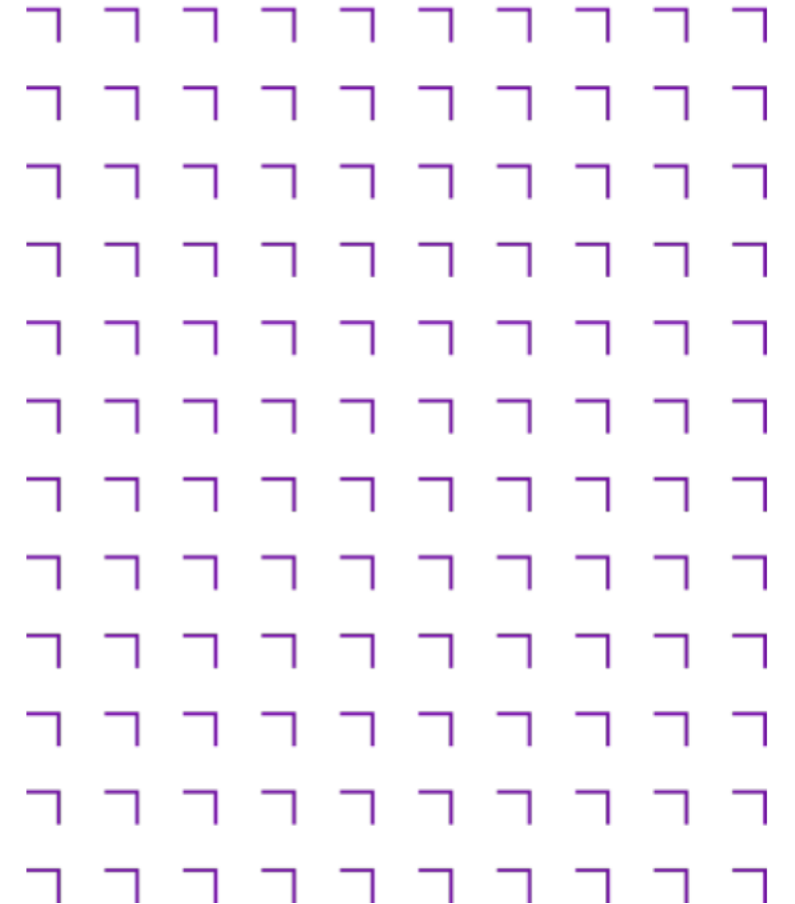
Demo

Activity

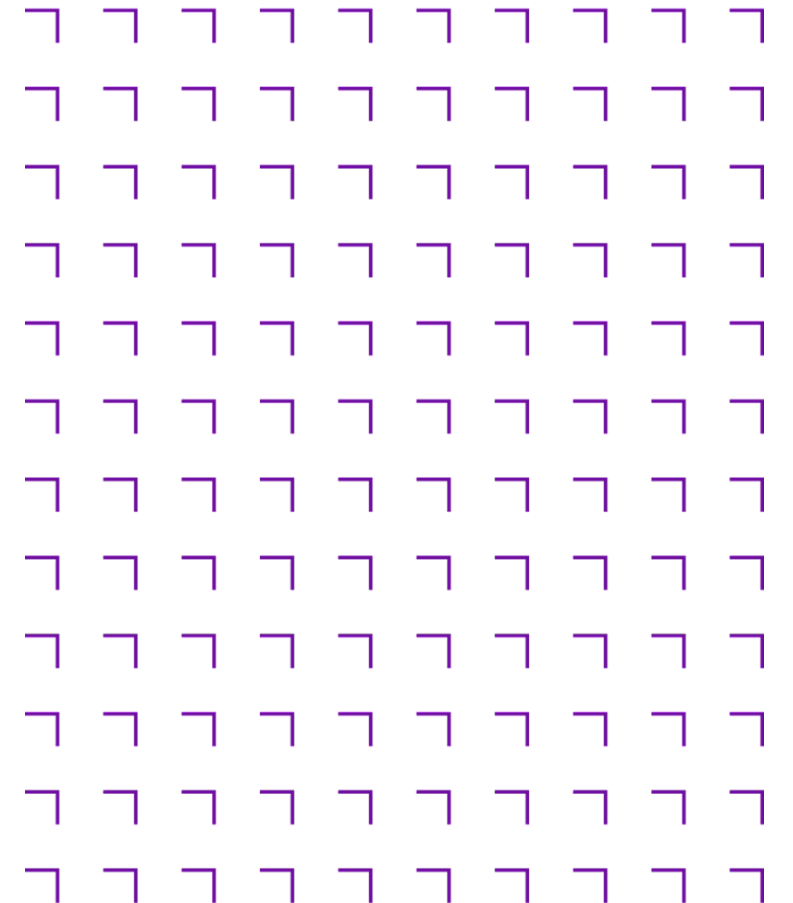


Breakout

- Join a breakout room
- Download the unit 15 exercises code from Moodle
- Follow the instructions and complete the exercises
- You have 35 minutes
- Lecturer will visit each room in turn, etc...
- Will start next topic on the hour



SEO and Web Security



What is Search Engine Optimisation (SEO)

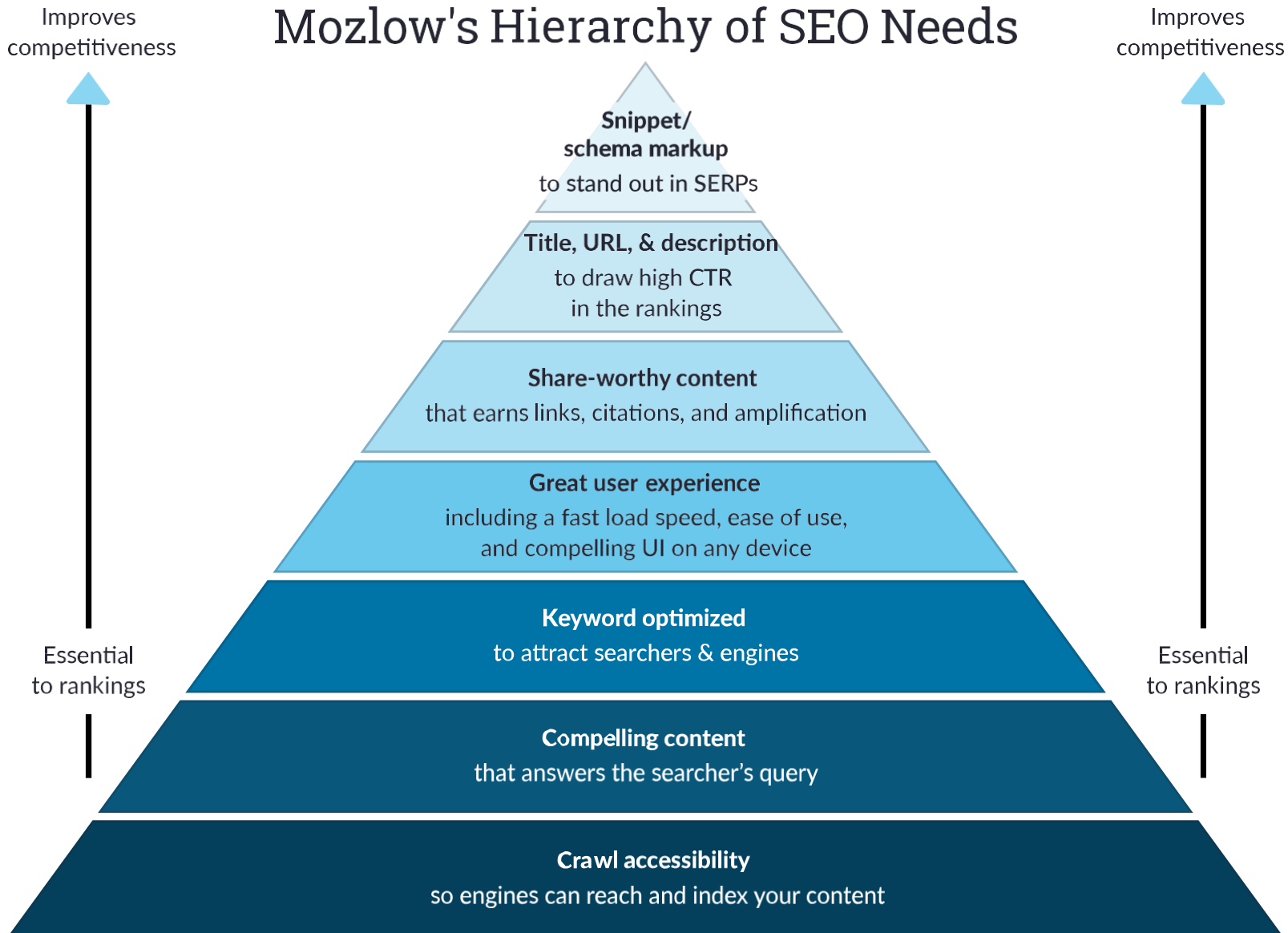
- ♦ Search engine optimization (SEO) is the process of improving the quality and quantity of website traffic to a website or a web page from search engines
- ♦ SEO targets unpaid traffic (known as "natural" or "organic" results) rather than direct traffic or paid traffic
- ♦ Unpaid traffic may originate from different kinds of searches, including image search, video search, academic search, news search, and industry-specific vertical search engines

Internet marketing strategy

- ♦ SEO considers how search engines work, the computer-programmed algorithms that dictate search engine behavior, what people search for, the actual search terms or keywords typed into search engines, and which search engines are preferred by their targeted audience
- ♦ SEO is performed because a website will receive more visitors from a search engine when websites rank higher on the search engine results page (SERP)
- ♦ These visitors can then potentially be converted into customers

<https://moz.com/beginners-guide-to-seo>

Mozlow's Hierarchy of SEO Needs



Seven steps to successful SEO:

- Crawl accessibility so engines can read your website
- Compelling content that answers the searcher's query
- Keyword optimized to attract searchers & engines
- Great user experience including a fast load speed and compelling UX
- Share-worthy content that earns links, citations, and amplification
- Title, URL, & description to draw high click-through-rate (CTR) in the rankings
- Snippet/schema markup to stand out in SERPs

Getting started

Install Analytics

Web analytics gives you information about who visits your site, how much traffic you receive and from where, what pages people visit, how they engage with your content, and more

- [Google Analytics](#)
- [Matomo](#)
- [Statcounter](#)

Register With Search Engines

Search engines like Bing and Google offer a wealth of information about your website that you simply can't get anywhere else. This data often includes the keywords you rank for in search, how each search engine crawls your site and more

Registering your site with search engines doesn't necessarily get you more traffic, but it can give you access to special tools to help you get there

- [Google Search Console](#)
- [Bing Webmaster Tools](#)

Getting Started

Quick Website Assessment

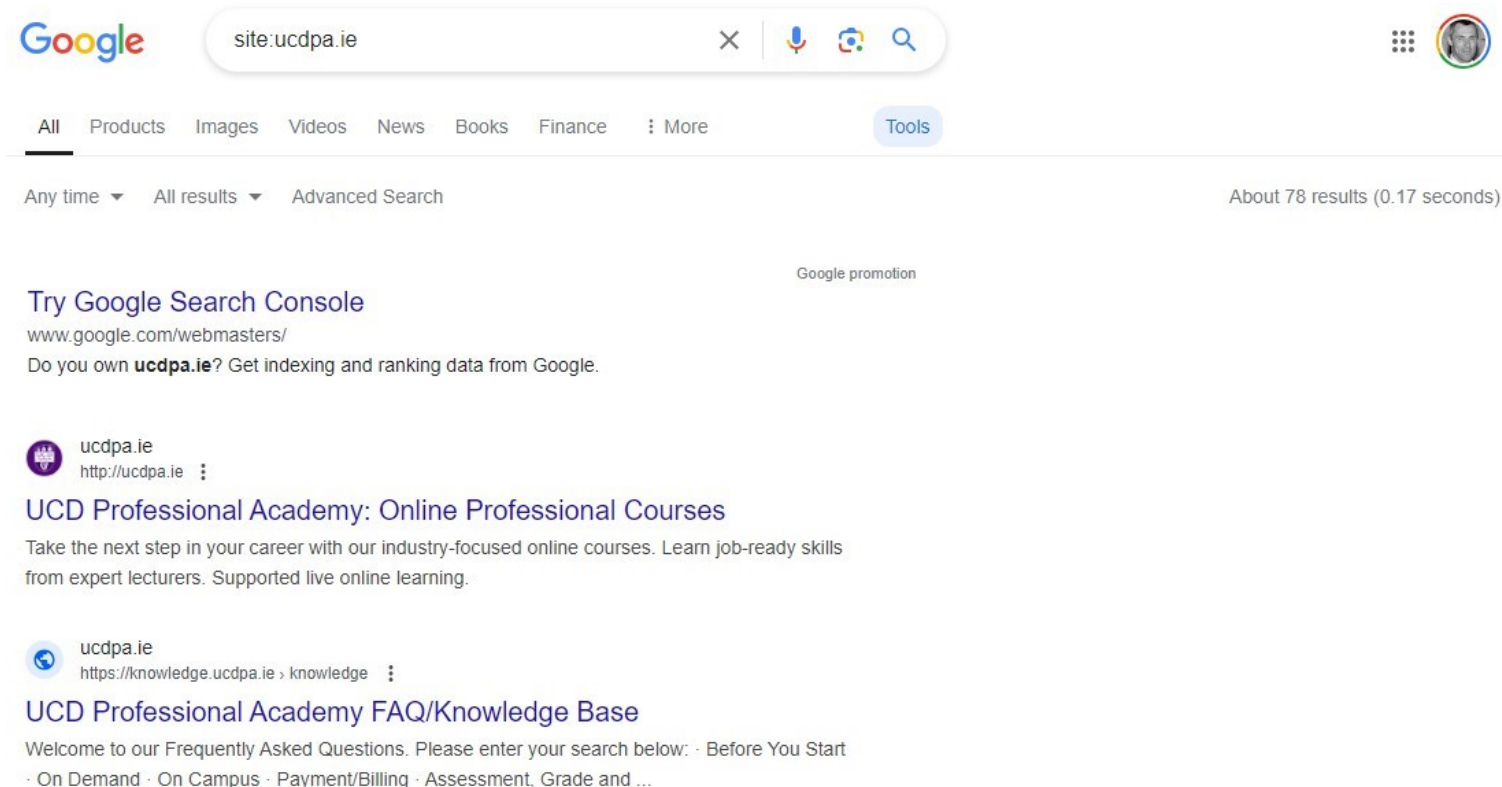
Gather some metrics together that will give you a bird's-eye-view of your overall performance Possessing these baseline metrics will give you a firm grasp on where you stand - your strengths, weaknesses, authority, and health

- ♦ [Moz Domain Authority Checker](#)

Getting Started

Check How Your Site Is Indexed

To be eligible to show up in search results, the pages of your site first must be “indexed” by search engines
Indexing basically means a search engine has crawled a page and made a record of its content



Webmaster Guidelines

Optimizing your site will help deliver better information to search engines so that your content can be properly indexed and displayed within search results

- ◆ Make pages primarily for users, not search engines
- ◆ Don't deceive your users
- ◆ Avoid tricks intended to improve search engine rankings. A good rule of thumb is whether you'd feel comfortable explaining what you've done to a website to a Google employee. Another useful test is to ask, "Does this help my users? Would I do this if search engines didn't exist?"
- ◆ Think about what makes your website unique, valuable, or engaging

Things to avoid:

- ◆ Automatically generated content
- ◆ Participating in link schemes
- ◆ Creating pages with little or no original content (i.e. copied from somewhere else)
- ◆ Cloaking — the practice of showing search engine crawlers different content than visitors
- ◆ Hidden text and links
- ◆ Doorway pages — pages created to rank well for specific searches to funnel traffic to your website

How do search engines work?

Search engines work through three primary functions:

- **Crawling:** Scour the Internet for content, looking over the code/content for each URL they find
- **Indexing:** Store and organize the content found during the crawling process. Once a page is in the index, it's in the running to be displayed as a result to relevant queries
- **Ranking:** Provide the pieces of content that will best answer a searcher's query, which means that results are ordered by most relevant to least relevant

Tell search engines how to crawl your site

- **Robots.txt** files are located in the root directory of websites (ex. `yourdomain.com/robots.txt`) and suggest which parts of your site search engines should and shouldn't crawl, as well as the speed at which they crawl your site, via [specific robots.txt directives](https://www.irishtimes.com/robots.txt), e.g. <https://www.irishtimes.com/robots.txt>
- **sitemap.xml**: an xml file in the root directory containing a list of URLs on your site that crawlers can use to discover and index your content. While submitting a sitemap doesn't replace the need for good site navigation, it can certainly help crawlers follow a path to all of your important pages; <https://support.google.com/webmasters/answer/183668?hl=en>, e.g. <https://www.irishtimes.com/arc/outboundfeeds/sitemap-news-index/latest/>

SEO Checklist

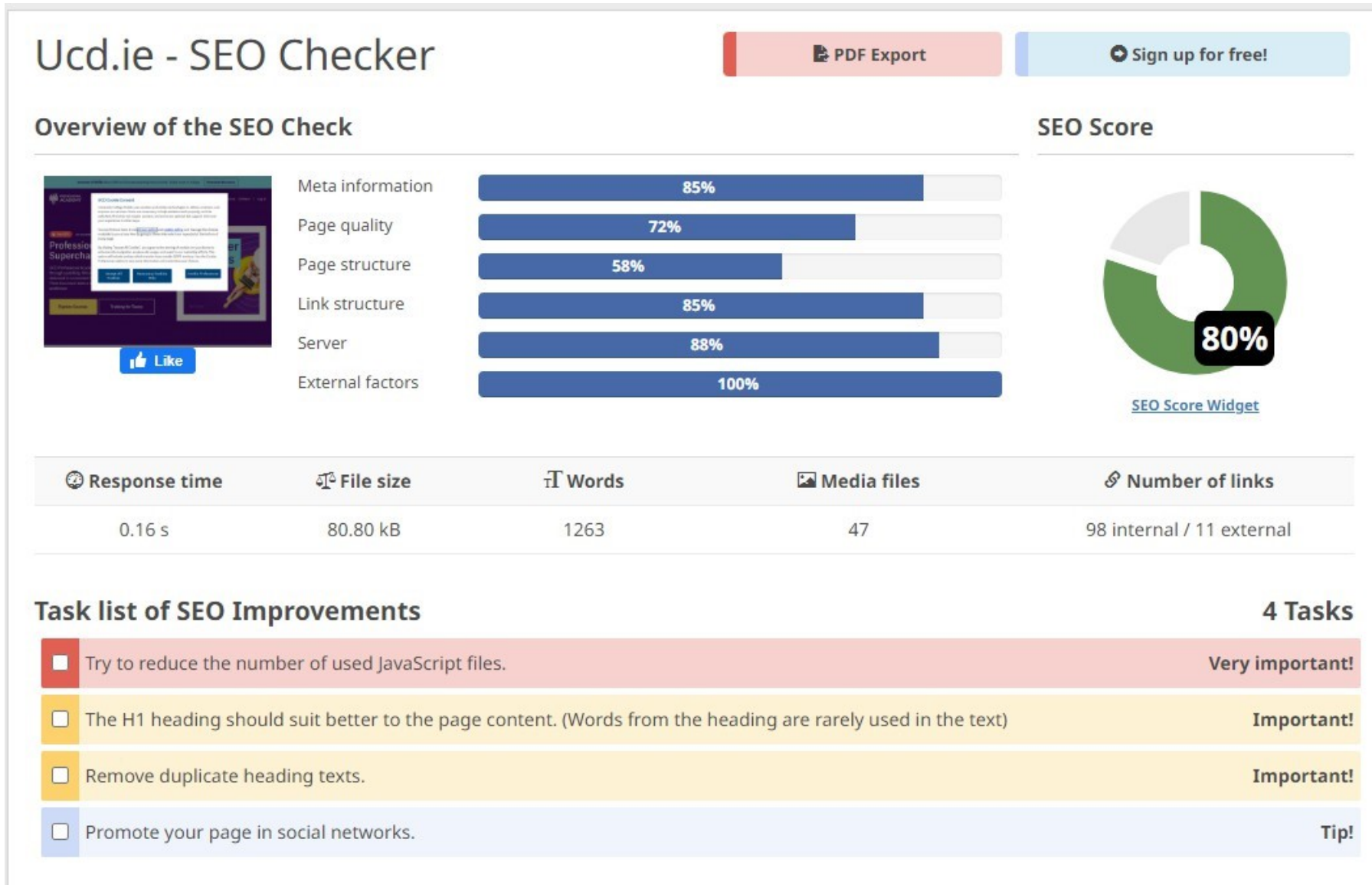
See <https://backlinko.com/seo-checklist> (demo/walk-thru)

- ♦ SEO Basics Checklist
- ♦ Keyword Research Checklist
- ♦ On-Page SEO Checklist
- ♦ Technical SEO Checklist
- ♦ Content Checklist
- ♦ Link Building Checklist

Tools

- ♦ [Semrush Keyword](#), [BacklinkO Keyword Tool](#), [WordStream](#)
- ♦ [Google Keyword Planner](#) - requires Google Ads a/c

Demo - <https://www.seobility.net/en/seocheck/>



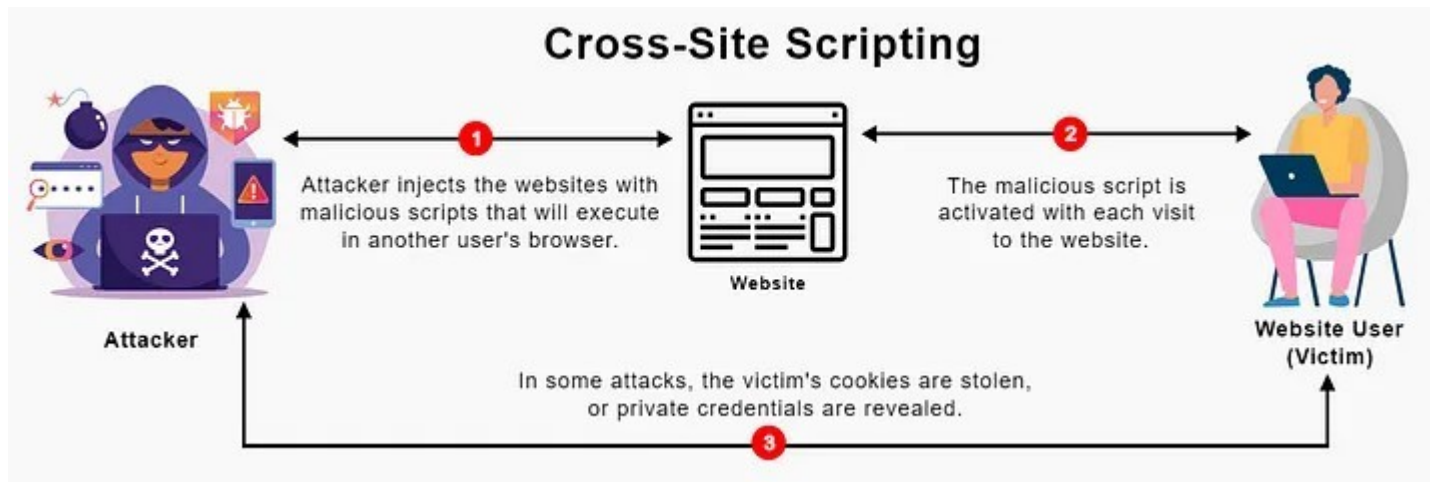
How to Secure Your Frontend: Top Security Practices

- Frontend security is a critical component of web development
- Even though the frontend primarily deals with user interfaces and interactions, it plays a significant role in protecting your application and user data
- Some security practices to secure your frontend

Cross-Site Scripting (XSS)

Cross-site scripting or XSS occurs when attackers inject malicious scripts into web pages viewed by other users

This can happen through input fields, URL parameters, or even user-generated content. Once injected, these scripts can steal session tokens, manipulate page content, or redirect users to malicious sites

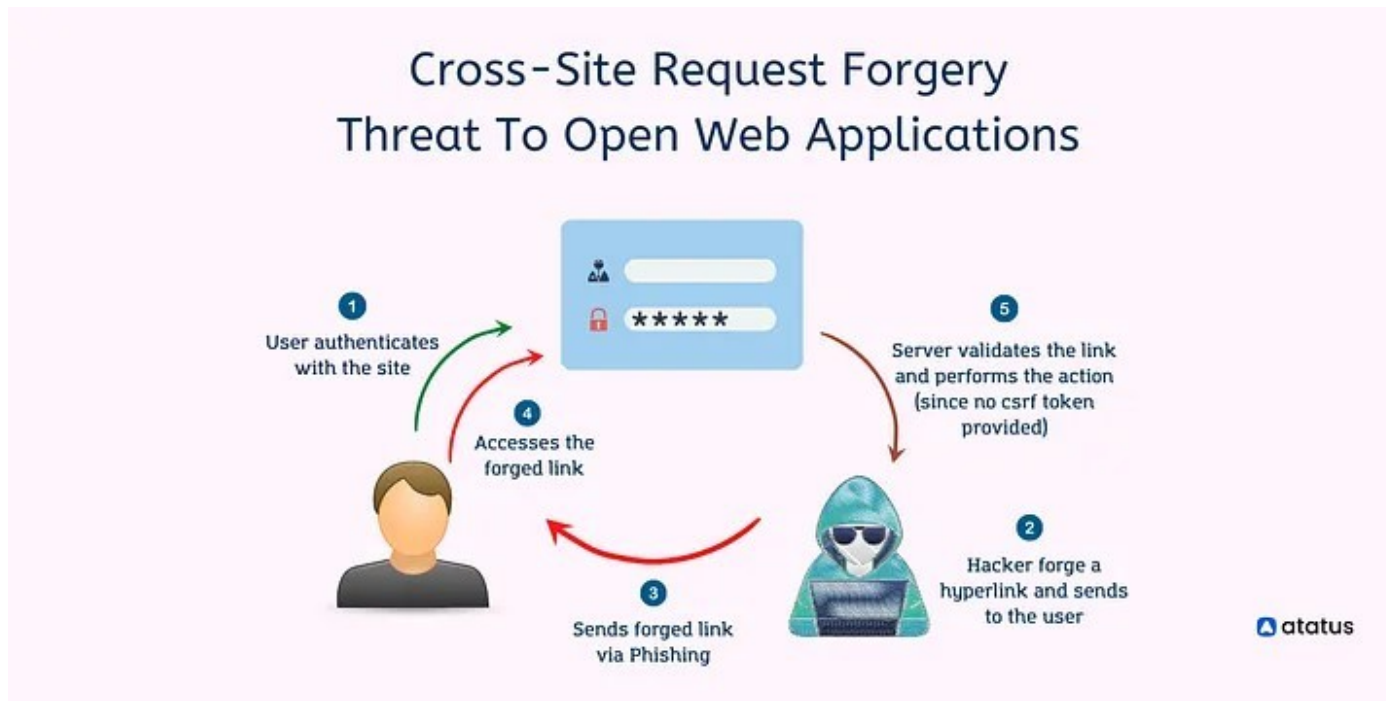


- XSS can be prevented by properly validating and sanitizing user input
- Use frameworks or libraries that automatically escape output
- Implement **Content Security Policy (CSP)** headers to restrict the sources from which content can be loaded

Cross-Site Request Forgery (CSRF)

CSRF attacks trick authenticated users into unknowingly executing actions on a web application without their consent

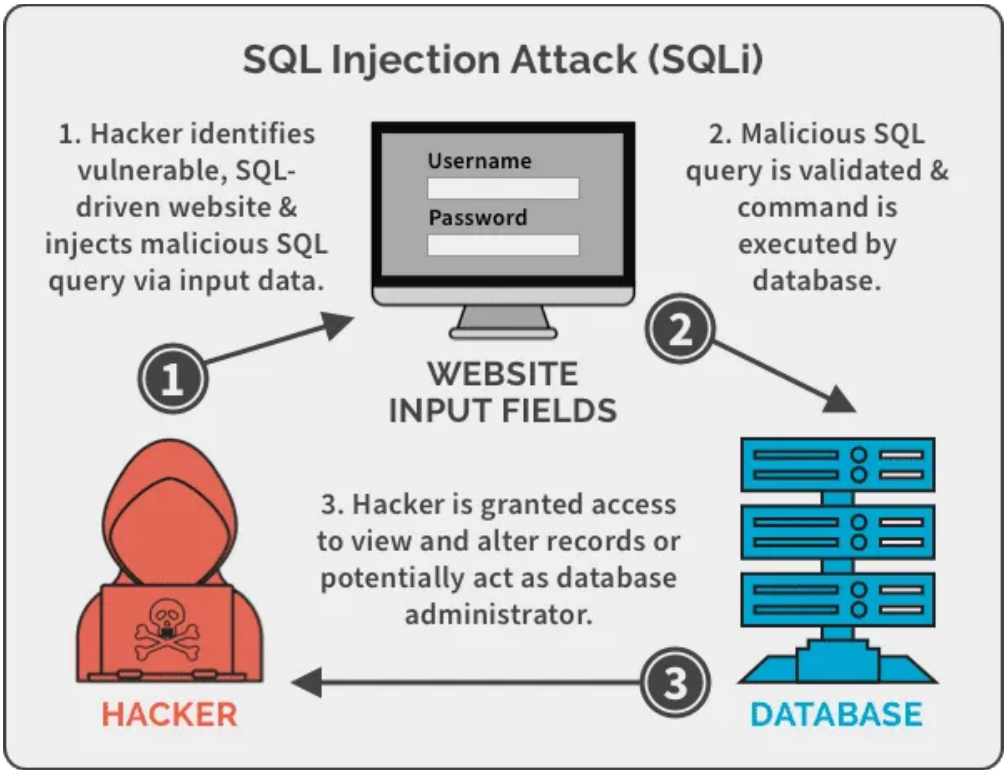
Attackers exploit the trust a site has in a user's browser by crafting malicious requests, often hidden within image tags or iframes, that are automatically submitted when the user visits a compromised page



To mitigate CSRF attacks, front-end developers should implement anti-CSRF tokens within forms and AJAX requests. These tokens should be unique per session and validated on the server side before processing any sensitive actions

Injection Attacks

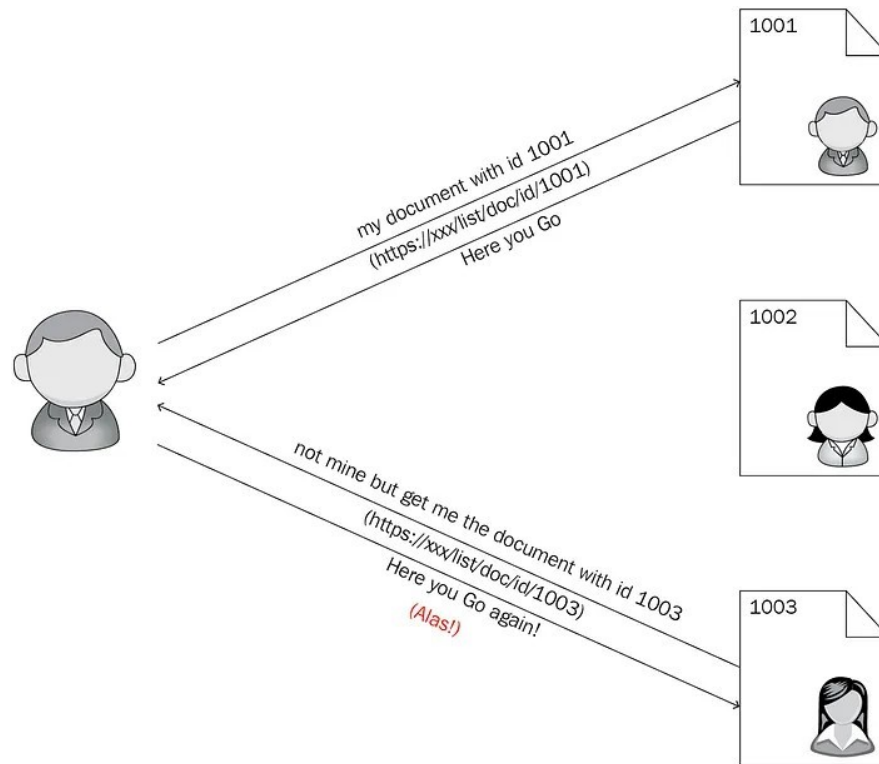
Injection attacks occur when untrusted data is sent to an interpreter as part of a command or query, e.g. SQL injection exploits poorly sanitized SQL queries to manipulate or extract data from a database



Avoid building SQL queries dynamically by concatenating strings with user input. Instead, use parameterized queries or ORM frameworks that automatically handle input sanitization. Additionally, input validation and proper encoding of user input can help prevent injection attacks

Insecure Direct Object References (IDOR)

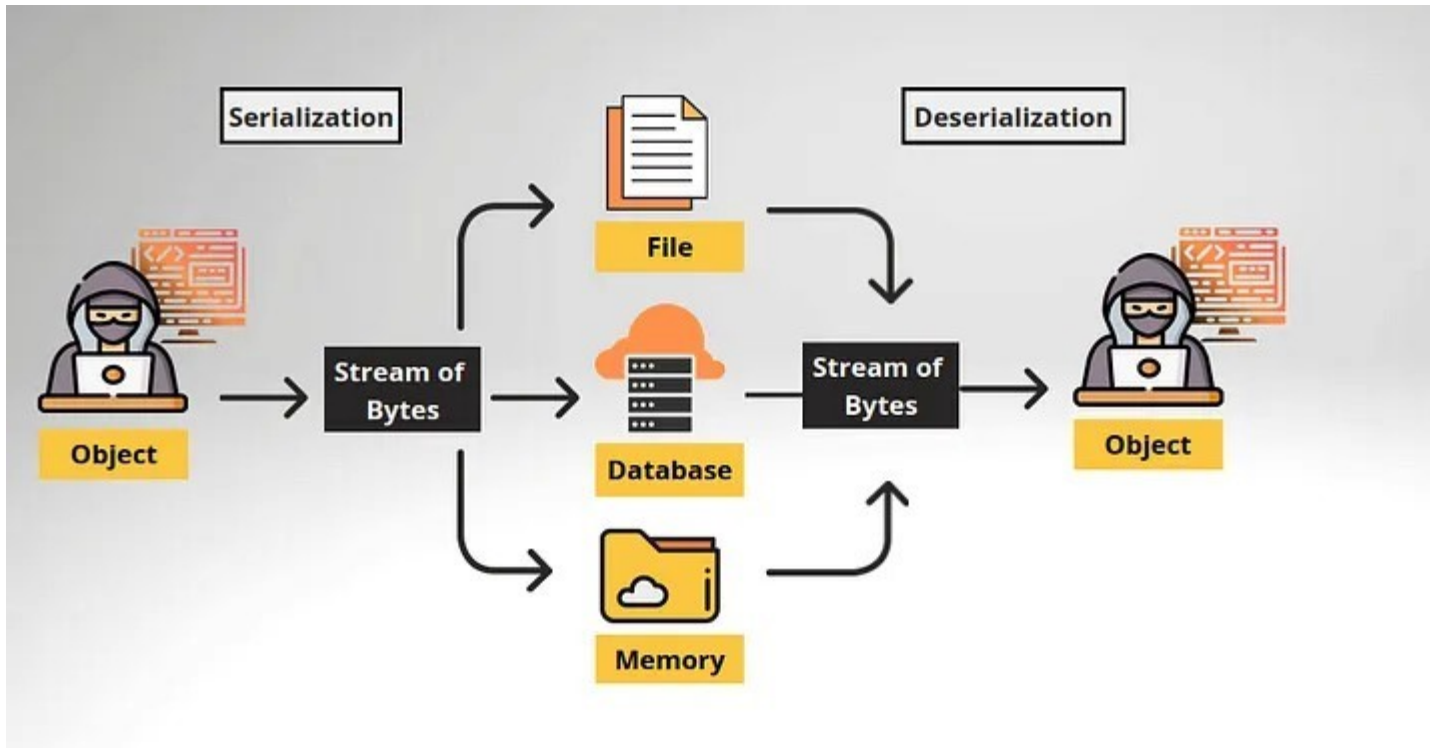
IDOR vulnerabilities arise when an application exposes internal implementation details, such as database/api keys or file paths, in its frontend interfaces



To prevent IDOR attacks implement access controls and enforce authorization rules on both the client and server sides and rely on secure, indirect references or unique tokens

Insecure Deserialization

Insecure deserialization occurs when untrusted data is deserialized by a program in a way that allows attackers to execute arbitrary code or manipulate the application's logic. Attackers exploit this vulnerability to gain unauthorized access, escalate privileges, or execute remote code execution attacks



Insecure Deserialization

Be cautious when handling serialized data from untrusted sources. Using standardized serialization formats and libraries with built-in validation mechanisms can help prevent insecure deserialization attacks

Additionally, implementing proper input validation and restricting deserialization to trusted sources can further mitigate this risk

Other Security Measures

1. Content Security Policy (CSP):

Utilize Content Security Policy headers to specify which resources are allowed to load on your website. This can help prevent unauthorized script execution

2. Secure Communication: Always use `HTTPS` to encrypt data transmitted between the client and server. Ensure that your API endpoints and external resources are also secured with HTTPS

3. Cross-Origin Resource Sharing (CORS): Configure `CORS` headers on your server to specify which domains are allowed to access your frontend resources. This mitigates Cross-Origin Request Forgery (CSRF) and Cross-Site Script Inclusion (XSSI) attacks

4. Authentication and Authorization: Implement strong user authentication and authorization mechanisms. Ensure that only authorized users can access sensitive parts of your frontend

5. Dependency Scanning: Regularly scan and update frontend dependencies to patch security vulnerabilities. Tools like `npm audit` or `yarn audit` can help identify and fix issues

2. Session Management: Implement secure session management practices, including session timeouts, secure cookies, and the use of HTTP-only cookies to prevent session theft

Other security measures


- 8. **Error Handling:** Avoid exposing sensitive information in error messages. Customize error handling to display user-friendly messages without revealing implementation details
- 9. **Protect Against CSRF:** Use anti-CSRF tokens to protect against Cross-Site Request Forgery attacks. Ensure that all state-changing requests require this token
- 10. **Security Headers:** Set security headers in your web server configuration to enhance frontend security. Examples include `X-Content-Type-Options` , `X-Frame-Options` , and `X-XSS-Protection`
- 11. **Regular Security Audits:** Conduct regular security audits of your frontend codebase to identify vulnerabilities and address them promptly
- 12. **Security Training:** Educate your development team about security best practices and keep them informed about emerging threats

Other security measures

- 13. Rate Limiting:** Implement rate limiting to restrict the number of requests a user can make in a given time frame, which helps protect against brute force attacks
- 14. Third-Party Integrations:** Thoroughly review and vet any third-party scripts or libraries you include in your frontend. Ensure they follow security best practices
- 15. Keep Abreast of Security News:** Stay informed about security threats and vulnerabilities relevant to frontend development by following security news and mailing lists
- 16. Security Headers:** Leverage security headers like `Content Security Policy (CSP)` and `Strict-Transport-Security (HSTS)` to add layers of security to your frontend
- 17. Security Automation:** Integrate security tools and automated scans into your development pipeline to catch vulnerabilities early in the development process

Demo - <https://securityheaders.com/>

Security Headers

Powered by  Probely


HomeAboutAPI

Scan your site now

ucdpa.ie

Scan

☒ Hide results ☒ Follow redirects

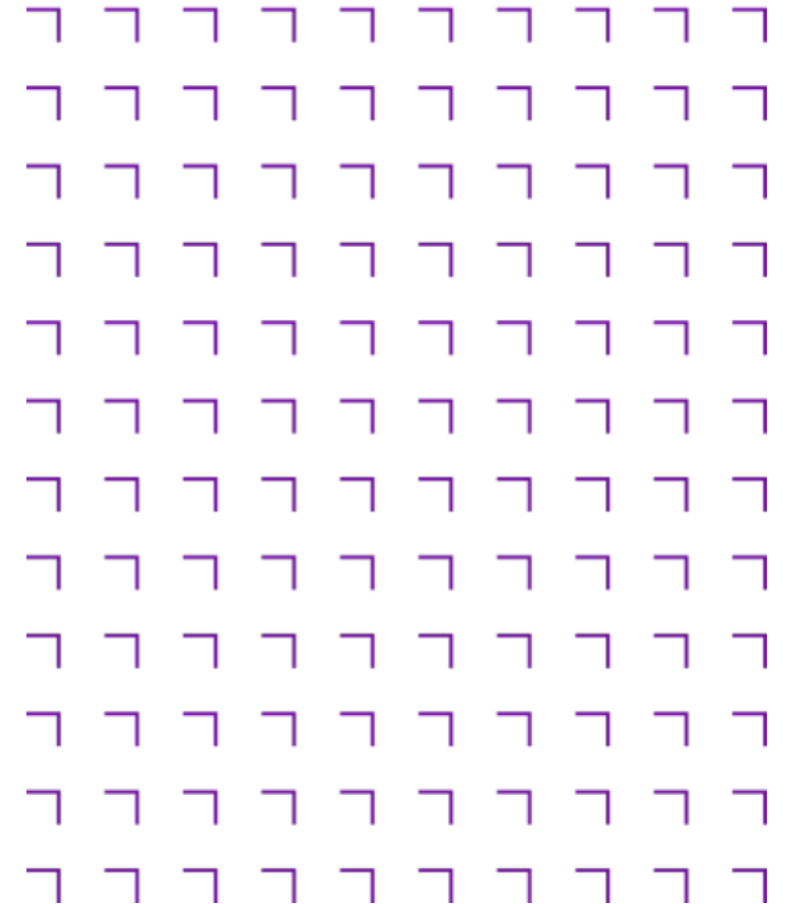
Security Report Summary	
	Site: https://www.ucd.ie/professionalacademy/?utm_source=ucdpa_referral&utm_medium=referral&utm_campaign=DoorDropURLJan
	IP Address: 34.240.51.193
	Report Time: 26 Jun 2024 12:03:38 UTC
	Headers: <div><div>✔ Strict-Transport-Security</div><div>✖ Content-Security-Policy</div><div>✖ X-Frame-Options</div><div>✖ X-Content-Type-Options</div><div>✖ Referrer-Policy</div><div>✖ Permissions-Policy</div></div>
	Advanced: Your site could be at risk, let's perform a deeper security analysis of your site and APIs: <div>Start Now</div>

Activity

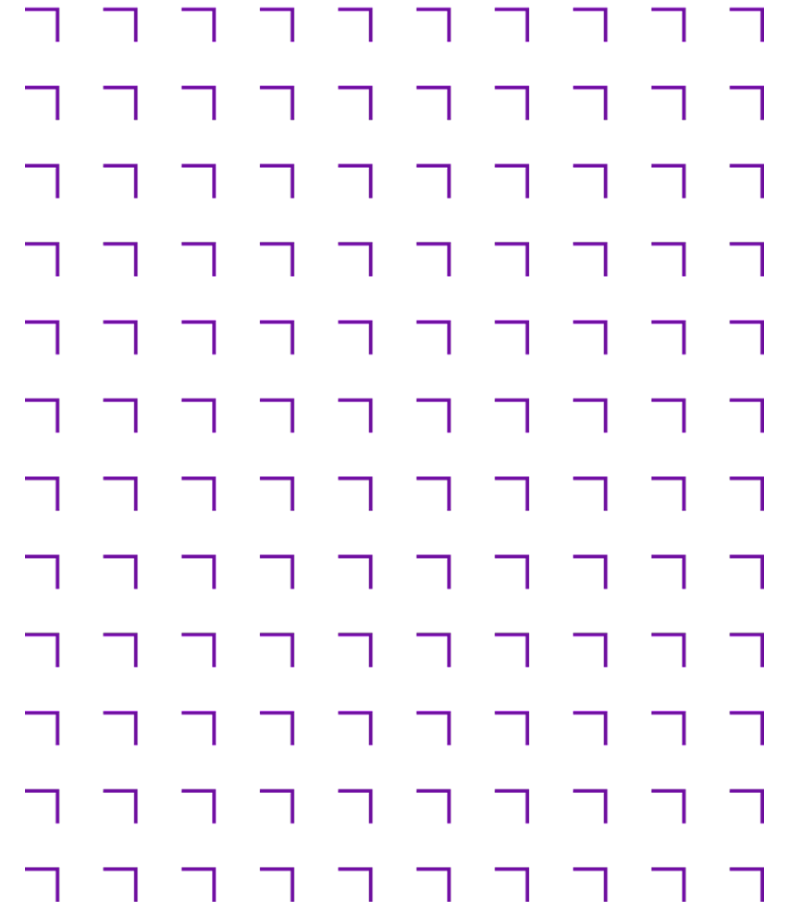


Breakout

- Join a breakout room
- Continue working on the unit 15 exercises
- You have 35 minutes
- Lecturer will visit each room in turn, etc...
- Will start next topic on the hour



Web Performance



Web Performance

- ◆ Web performance is a crucial aspect of web development that focuses on the speed at which pages load, as well as how responsive they are to user input
- ◆ When you optimize your website for performance, you're giving users a better experience
- ◆ Performance plays a significant role in the success of any online venture, as high performing sites engage and retain users better than poorly performing ones
- ◆ When talking about performance, it's important to be precise and to refer to performance in terms of metrics
- ◆ To help ensure the metrics are relevant to users, we frame them around a few key questions:
 - Is it happening? Did the navigation start successfully? Has the server responded?
 - Is it useful? Has enough content rendered that users can engage with it?
 - Is it usable? Can users interact with the page, or is it busy?
 - Is it delightful? Are the interactions smooth and natural, free of lag?

<https://web.dev/explore/learn-core-web-vitals>

Improving Web Performance leads to:

Better Conversion

Users expect a quick, frictionless journey to their destination. And if your page loads too slowly, most visitors will leave

Improved website performance leads to an increase in conversion rate, the key metric for most online businesses. In a study, [46% of users](#) bounce if a landing page loads longer than five seconds and ideally be less than **three seconds**

Improved SEO Strategy

Good front-end performance is an integral part of the Search Engine Optimization (SEO) strategy. In 2010, [Google announced](#) that site speed would greatly impact site ranking. So if your page loads slowly, it has no chance of appearing at the top of search results.

Improved Usability and Accessibility

Finally, improving your front-end performance is closely related to user experience and accessibility. Users don't appreciate poor loading speed. Your site may also be entirely inaccessible for specific audience groups. E.g., a [Microsoft study](#) claims that 71% of people with disabilities will leave your website immediately if it's not accessible for them

Types of metrics

There are several other types of metrics that are relevant to how users perceive performance

- ♦ **Perceived load speed:** how quickly a page can load and render all of its visual elements to the screen
- ♦ **Load responsiveness:** how quickly a page can load and execute any required JavaScript code in order for components to respond quickly to user interaction
- ♦ **Runtime responsiveness:** after page load, how quickly can the page respond to user interaction
- ♦ **Visual stability:** do elements on the page shift in ways that users don't expect and potentially interfere with their interactions?
- ♦ **Smoothness:** do transitions and animations render at a consistent frame rate and flow fluidly from one state to the next?

Metrics can be measured in two primary ways:

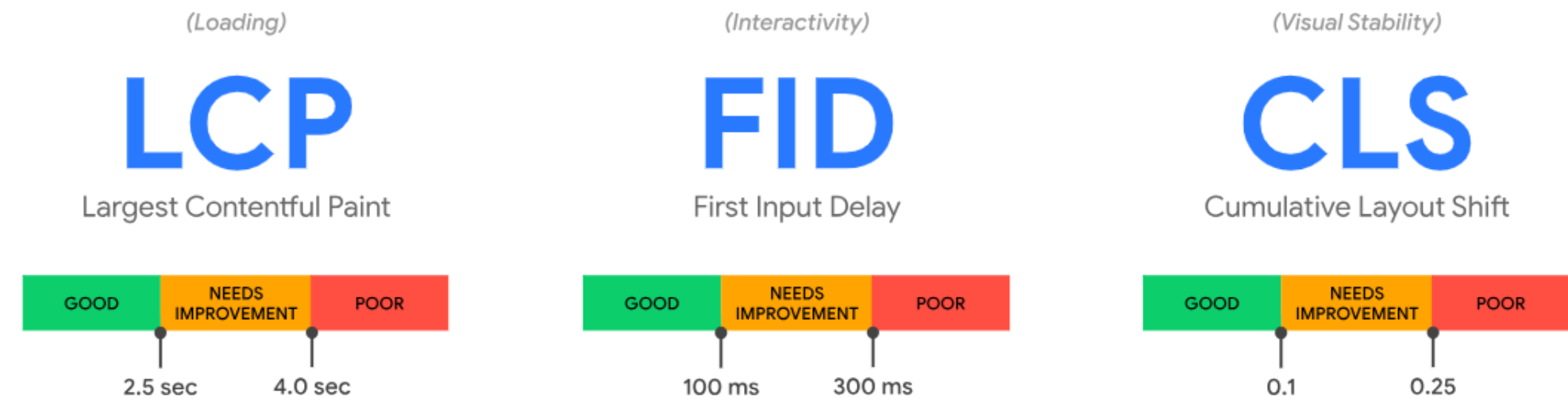
- ♦ **in the lab:** Testing performance in the lab is essential when developing new features
- ♦ **in the field:** The performance of a site can vary dramatically based on an individual user's device capabilities and their network conditions. It can also vary based on whether (or how) a user is interacting with the page

Important metrics to measure

- **First Contentful Paint (FCP)**: measures the time from when the page starts loading to when any part of the page's content is rendered on the screen. (lab, field)
- **Largest Contentful Paint (LCP)**: measures the time from when the page starts loading to when the largest text block or image element is rendered on the screen. (lab, field)
- **Interaction to Next Paint (INP)**: measures the latency of every tap, click, or keyboard interaction made with the page, and—based on the number of interactions—selects the worst interaction latency of the page (or close to the highest) as a single, representative value to describe a page's overall responsiveness. (lab, field)
- **Total Blocking Time (TBT)**: measures the total amount of time between FCP and TTI where the main thread was blocked for long enough to prevent input responsiveness. (lab)
- **Cumulative Layout Shift (CLS)**: measures the cumulative score of all unexpected layout shifts that occur between when the page starts loading and when its lifecycle state changes to hidden. (lab, field)
- **Time to First Byte (TTFB)**: measures the time it takes for the network to respond to a user request with the first byte of a resource. (lab, field)
- **First Input Delay (FID)**: measures the time from when a user first interacts with a page (that is, when they click a link, tap on a button, or use a custom, JavaScript-powered control) to the time when the browser is actually able to begin processing event handlers in response to that interaction

Core Web Vitals metric and thresholds

Core Web Vitals are a set of field metrics that measure important aspects of real-world user experience on the web

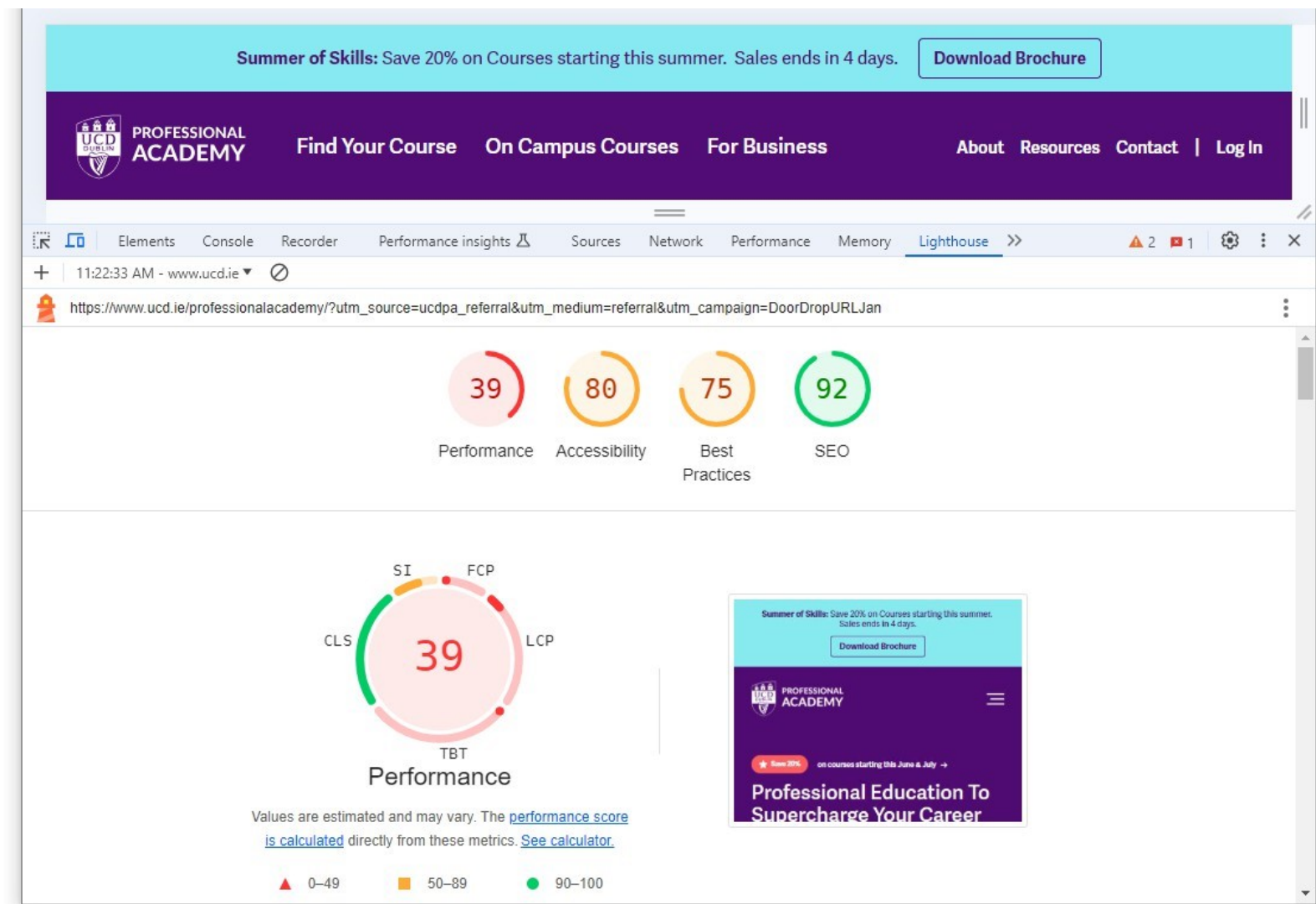


	Good	Poor	Percentile
Largest Contentful Paint	≤2500ms	>4000ms	75
First Input Delay	≤100ms	>300ms	75
Cumulative Layout Shift	≤0.1	>0.25	75

Optimisation Techniques <https://roadmap.sh/best-practices/frontend-performance>

High Priority	Medium Priority
<input type="checkbox"/> Minimize number of iframes <input type="checkbox"/> Minified CSS - Remove comments, whitespaces etc <input type="checkbox"/> CSS files are non-blocking <input type="checkbox"/> Inline the Critical CSS (above the fold CSS) <input type="checkbox"/> Avoid the embedded / inline CSS <input type="checkbox"/> Analyse stylesheets complexity <input type="checkbox"/> Compress your images / keep the image count low <input type="checkbox"/> Choose your image format appropriately <input type="checkbox"/> Minify your JavaScript <input type="checkbox"/> Non Blocking JavaScript: Use async / defer <input type="checkbox"/> Use HTTPs on your website <input type="checkbox"/> Keep your page weight < 1500 KB (ideally < 500 kb) <input type="checkbox"/> Keep your page load time < 3 seconds <input type="checkbox"/> Keep the Time To First Byte < 1.3 seconds <input type="checkbox"/> Minimize HTTP Requests <input type="checkbox"/> Serve files from the same protocol <input type="checkbox"/> Avoid requesting unreachable files (404) <input type="checkbox"/> Set HTTP cache headers properly <input type="checkbox"/>	<input type="checkbox"/> Minified HTML - Remove comments and whitespaces <input type="checkbox"/> Use Content Delivery Network <input type="checkbox"/> Prefer using vector image rather than bitmap images <input type="checkbox"/> Set width and height attributes on images (aspect ratio) <input type="checkbox"/> Avoid using Base64 images <input type="checkbox"/> Offscreen images are loaded lazily <input type="checkbox"/> Ensure to serve images that are close to your display size <input type="checkbox"/> Avoid multiple inline JavaScript snippets <script> <input type="checkbox"/> Keep your dependencies up to date <input type="checkbox"/> Check for performance problems in your JavaScript files <input type="checkbox"/> Service Workers for caching / performing heavy tasks <input type="checkbox"/> Cookie size should be less than 4096 bytes <input type="checkbox"/> Keep the cookie count less than 20
	Low Priority
	<input type="checkbox"/> Pre-load URLs where possible <input type="checkbox"/> Concatenate CSS into a single file <input type="checkbox"/>

Demo (Lighthouse) - <https://ucdpa.ie/>

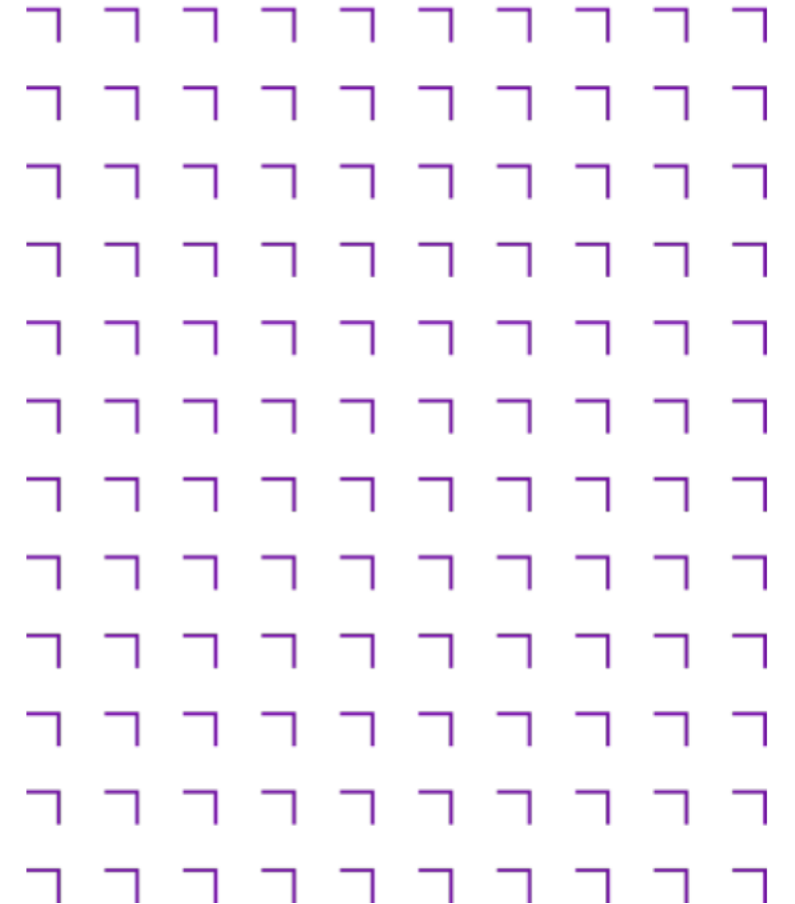


Activity



Breakout

- Join a breakout room
- Continue working on the unit 15 exercises
- You have 35 minutes
- Lecturer will visit each room in turn, etc...
- Will start next topic on the hour



Summary



Completed this Week

- Other Frameworks
- SEO and Web Security
- Web Performance

For Next Week

- Complete the remaining exercises for unit 15 before next class

