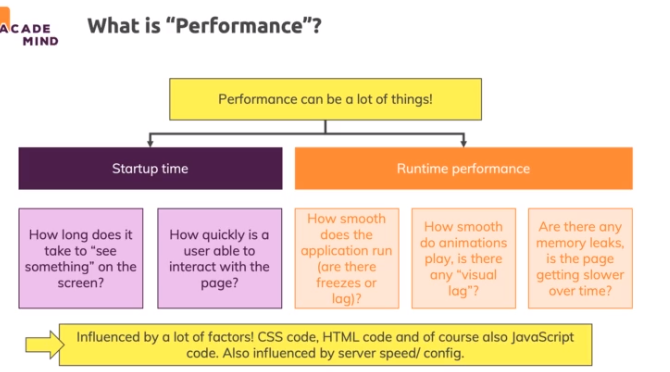
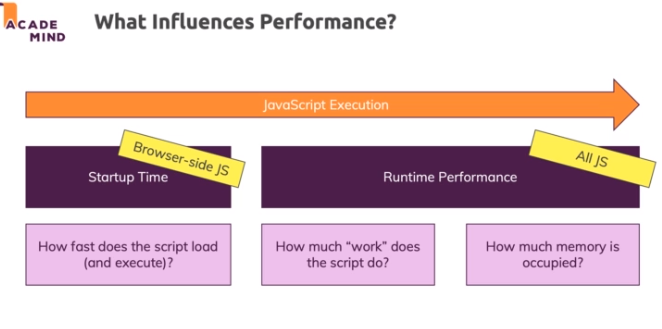
**Performance & optimizations**

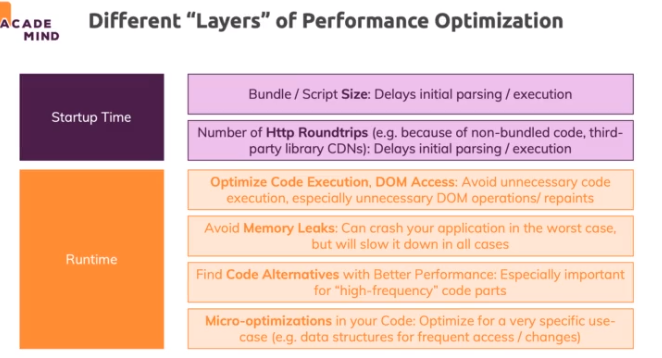


**What is "Performance Optimization" About?**

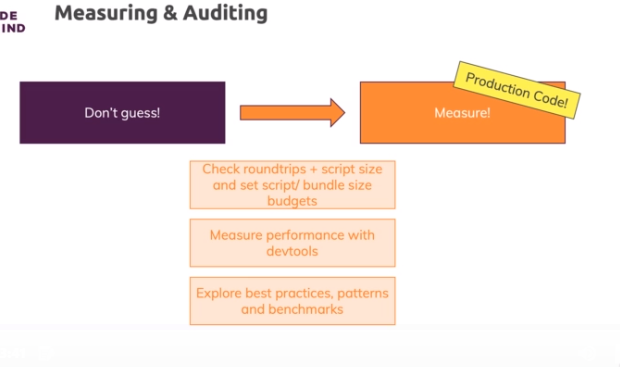


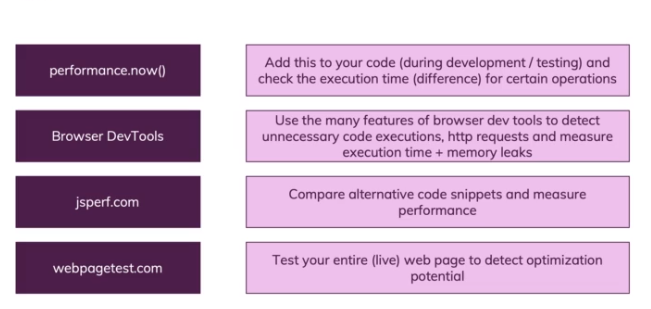


**Optimization Potentials**



**Measuring Performance**





**Server-side Performance Optimizations**

**In this course section we - of course - discussed JavaScript performance.**

**When it comes to the overall performance of a website, it's not just JavaScript that matters though. Besides other client-side optimizations (e.g. CSS, images => see:** [**https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency**](https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency)**), it's also the server-side where you can improve performance.**

**Executing code on the server-side (e.g. NodeJS) is one thing but the server configuration is also important.**

**Specifically, there are three main areas of improvement which you might want to look into:**

* + Compression of served assets
  + Caching (client-side and server-side)
  + HTTP/2

**Compression**

Compression is about zipping static assets (CSS, JS, images) before serving them. Modern browsers know how to unzip such files and will automatically do so. Since zipped assets are transferred, less data is sent from server to client => Faster load time.

How you set up compression depends on which server/ service you're using. For example on Firebase, static assets are automatically compressed.

When having your own NodeJS server-side code, you would have to manually ensure that static assets are compressed (<https://github.com/expressjs/compression>).

Caching

Caching is a complex topic - it's about saving data or files for re-use and it can be done on different levels.

For example, browser automatically cache files (e.g. JS files) for you - based on the caching headers set by the serving host (<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Cache-Control>). So controlling these headers on the server-side config, allows you to control how browsers will cache such files. This can help you avoid unnecessary data transfer but of course you also have to make sure that visitors of your site don't miss out on important updates.

Server-side caching is all about storing data you work with on the server (e.g. fetched from a database) such that multiple requests requesting the same data can get that cached data.

You can learn more about caching here: <https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/http-caching>

And here: <https://wp-rocket.me/blog/different-types-of-caching/>

HTTP/2

HTTP/2 is the latest "form" of the Http protocol and unlike HTTP 1, it allows for "server push". That means that servers can push required assets/ files actively to a client (instead of waiting for the client to request them).

You can learn more about it here: <https://developers.google.com/web/fundamentals/performance/http2>