

# front-end Build & Performance

lab 5/8

# today

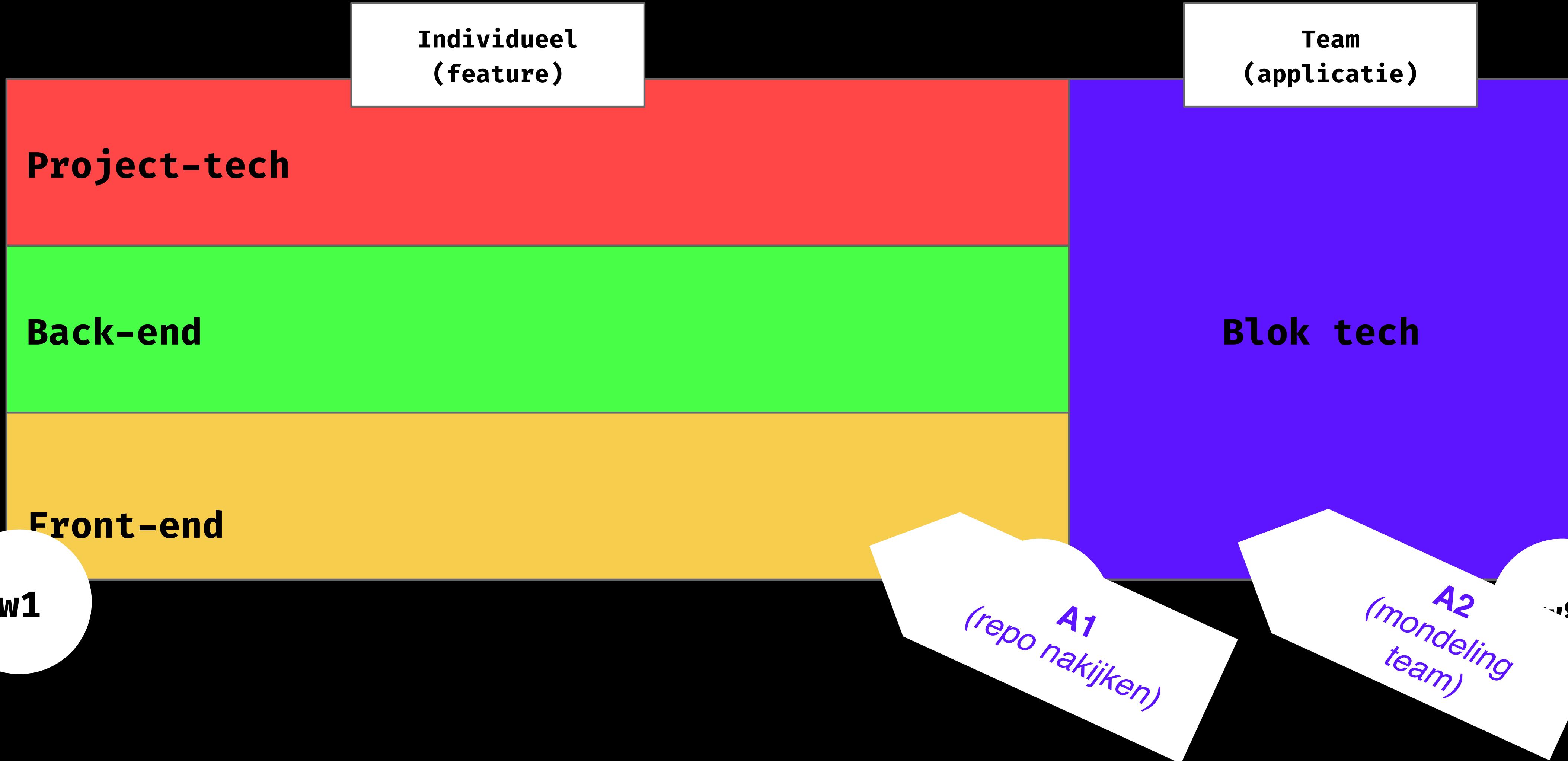
I . ~~Standup~~

II . Team assignment (A2)

III . Build Tools

IV . Performance

# Team assignment



	1-2	3-4	5-6	7-8	9-10
Application	There is no production optimization	The production optimization isn't implemented in code or partially implemented but gives errors and is buggy	There is a production optimization and the technical implementation works. The set-up is integrated into the project and merged.	There is a production optimization and the technical implementation works and is merged. All team members benefitted of code you've written	There is a production optimization and the technical implementation works and is merged. All team members benefitted of code you've written. The application is resilient by streamlining the asset pipeline.
Research	There is no research in the wiki	There is research in the wiki but very limited, not enough to understand your process	There is research about what production topic you've implemented in the team repo, you've documented your process and ideas and you've researched your topic. You've written about how you implemented your production topic.	The research is in-depth and well-documented. You've written about the code and did research on alternatives	The wiki reads like a book. Your production documentation is progressive enhancement and code.
Quality	There is no cohesion in the front-end code of the application. All features are individually made and don't	There was an attempt to unify the front-end in a single application, but it's buggy or not	Front-end architecture has been normalised among all team members. There are basic guidelines each team member follows. The HTML, CSS and JavaScript are written using previous standard.	The application is robust and reliable. The application has a solid strategy and code is modular and applies only	The application is professional development agency.

**Topic**

**Docs**

**Quality**

Filters  is:issue sort:updated-desc is:closed

Clear current search query, filters, and sorts

1 Open ✓ 16 Closed

⚠ Add CSS to authentication files  
#38 by HappyPantss was closed on 14 Apr 2020  updated on 14 Apr 2020

⚠ Only  
#32 by StanBankras  enhancement

⚠ Indentation of the Javascript code  
#26 by StanBankras was closed on 14 Apr 2020  updated on 14 Apr 2020

⚠ Dashboard page  
#30 by StanBankras was closed on 13 Apr 2020  updated on 13 Apr 2020

⚠ Full profile preview page  
#31 by StanBankras was closed on 13 Apr 2020  updated on 13 Apr 2020

⚠ Connect filter feature with match slider users  
#35 by StanBankras  enhancement

Pull requests Issues Trending Explore

StanBankras / amatch

Code Issues Pull requests Actions Projects Wiki More

## Topic: Split route logic

Stan Bankras edited this page on Apr 7 · 6 revisions

### Backend topic: Split route logic

This topic was worked on by Stan Bankras

A team project can become quite a mess if everyone works in the same server.js or index.js file: the file quickly adds a lot of lines and it takes long to read and understand what is happening. That's why I wanted to separate as much as possible

- Moving route logic to other files than the server
- Exporting the database connection from one central place

### Moving route logic to other files than the server

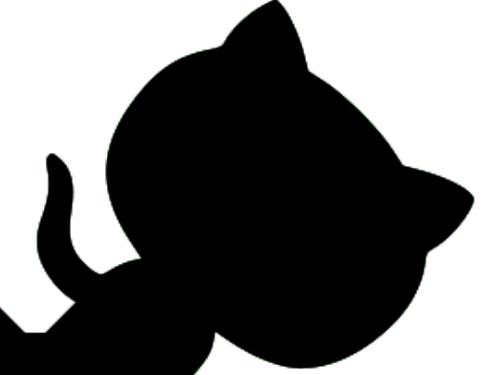
In Javascript you can require other javascript files using require(). This can be used to import different files that host routes in the server.js to clear up some space:

```
const matchRouter = require('./routes/matches');
const chatRouter = require('./routes/chatting');
const authRouter = require('./routes/authentication');
```

# Assignment (45m)

Think about a topic (production optimisation) you want to work on for front-end.

Tip; check resources on FeD repo or have a look at example repo's from previous groups.



# Build Tools

# Components

CSS



```
<style type="text/css" src="styles/main.css"></style>
<style type="text/css" src="styles/partials/header.css"></style>
<style type="text/css" src="styles/partials/footer.css"></style>
<style type="text/css" src="styles/modules/main.css"></style>
<style type="text/css" src="styles/main.css"></style>
```

5(!) http requests?????

# Components

# Bundler

A screenshot of a Google search results page. The search bar at the top contains the query "css bundler". Below the search bar, there are navigation links for "All", "Images", "Videos", "Shopping", "News", and "More", followed by a "Tools" link. The main content area displays search results. The first result is a link to the Parcel website, titled "Parcel", with a description: "The zero configuration build tool for the web. JavaScript. **CSS**. HTML. TypeScript. React. images. SASS. SVG. Vue. libraries. Less. CoffeeScript. Node." and links to "Announcing Parcel CSS · Parcel Documentation · Building a web app with Parcel". The second result is a link to the PostCSS documentation, titled "CSS - Parcel", with a description: "PostCSS is a tool for transforming **CSS** with plugins, like autoprefixer, Preset Env, and many others. You can configure PostCSS with Parcel by creating a ...".

Google

css bundler

All Images Videos Shopping News More Tools

About 757.000 results (0,47 seconds)

<https://parceljs.org> ::

**Parcel**

The zero configuration build tool for the web. JavaScript. **CSS**. HTML. TypeScript. React. images. SASS. SVG. Vue. libraries. Less. CoffeeScript. Node.

[Announcing Parcel CSS](#) · [Parcel Documentation](#) · [Building a web app with Parcel](#)

<https://parceljs.org/languages/css> ::

**CSS - Parcel**

PostCSS is a tool for transforming **CSS** with plugins, like autoprefixer, Preset Env, and many others. You can configure PostCSS with Parcel by creating a ...

# Components

# Bundler

The screenshot shows the Parcel documentation website. The top navigation bar includes a logo, a search bar, and links for 'Docs' and 'Blog'. On the left, there's a sidebar with sections for 'GETTING STARTED' (selected 'Web app'), 'FEATURES' (including Development, Code Splitting, Dependency resolution, Bundle inlining, Targets, Production, Scope hoisting, Node Emulation, CLI, Parcel API, Plugins, and Plugin Search), and 'ON THIS PAGE' (Installation, Project setup, Package scripts, Declaring browser targets, and Next steps). The main content area features a large heading 'Building a web app with Parcel', a 'Installation' section with instructions for Yarn and npm, a 'Project setup' section with a code snippet for index.html, and a 'Declaring browser targets' section.

**ON THIS PAGE**

- Installation
- Project setup
- Package scripts
- Declaring browser targets
- Next steps

**Building a web app with Parcel**

## Installation

Before we get started, you'll need to install Node and Yarn or npm, and create a directory for your project. Then, install Parcel into your app using Yarn:

```
yarn add --dev parcel
```

Or when using npm run:

```
npm install --save-dev parcel
```

## Project setup

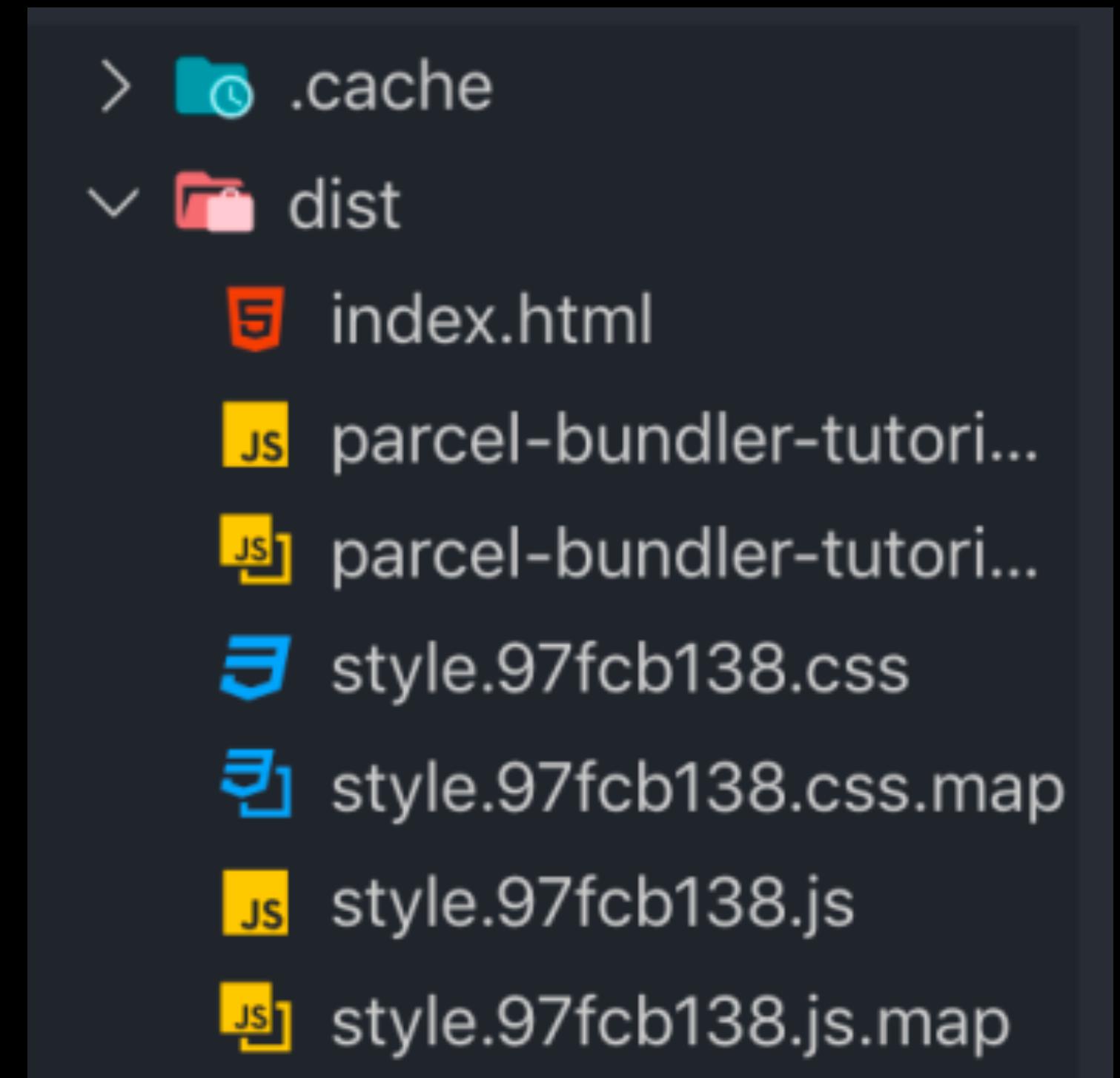
Now that Parcel is installed, let's create some source files for our app. Parcel accepts any type of file as an entry point, but an HTML file is a good place to start. Parcel will follow all of your dependencies from there to build your app.

```
src/index.html:  
<!doctype html>  
<html lang="en">
```

# Components

# Bundler

Built in 444ms.		
dist/parcel-bundler-tutorial.8bdf8f45.js	1.48 KB	241ms
dist/style.134bbd53.css.map	779 B	5ms
dist/parcel-bundler-tutorial.8bdf8f45.js.map	756 B	2ms
dist/index.html	639 B	5ms
dist/style.134bbd53.css	337 B	13ms



# Components

# Bundler

**Gulp** GET STARTED API PLUGINS DONATE ENTERPRISE

[Twitter](#) [Medium](#) [Search](#)

TypeScript Develop in any language

PNG Create assets with any tool

Markdown Write using any format

A toolkit to automate & enhance your workflow

Leverage gulp and the fast, reliable, and repeatable workflow build pipelines.

 rollup.js

- » [Introduction](#)
  - Overview
  - Installation
  - Quick start
  - The Why
  - Tree-Shaking
  - Compatibility
- » [Command Line Interface](#)
  - Configuration Files
  - Differences to the JavaScript API
  - Loading a configuration from a Node package
  - Using untranspiled config files
  - Command line flags
  - Reading a file from stdin
- » [JavaScript API](#)
  - rollup.rollup
  - rollup.watch

# rollup.js

## Introduction

- [Overview](#)
- [Installation](#)
- [Quick start](#)
- [The Why](#)
- [Tree-Shaking](#)
- [Compatibility](#)
  - [Importing CommonJS](#)
  - [Publishing ES Modules](#)

Black Lives Matter

→ Getting Started ⚙ Configuring Tasks 🌐 Plugins



# GRUNT

## The JavaScript Task Runner

### Latest Version

- Stable: [v1.4.1](#) (npm)
- Development: [HEAD](#) (GitHub)

Ads by [Bocoup](#).

### Latest News

[Grunt 1.4.0 released](#)

### Why use a task runner?

In one word: automation. The less work you have to do when performing repetitive tasks like minification, compilation, unit testing, linting, etc, the easier your job becomes. After you've configured it through a [Gruntfile](#), a task runner can do most of that mundane work for you—and your team—with basically zero effort.

### Why use Grunt?

The Grunt ecosystem is huge and growing every day. With literally hundreds of thousands of Grunt plugins to choose from, you can use Grunt to do almost anything with a task. If someone hasn't already created a plugin for what you need, authoring and publishing your own Grunt plugin to npm is a breeze. Get started.

# Build tools

Ideas

- Bundle CSS (post/pre processor)
- Bundle JS (bundler, npm scripts)
- Use a task runner (gulp, grunt)

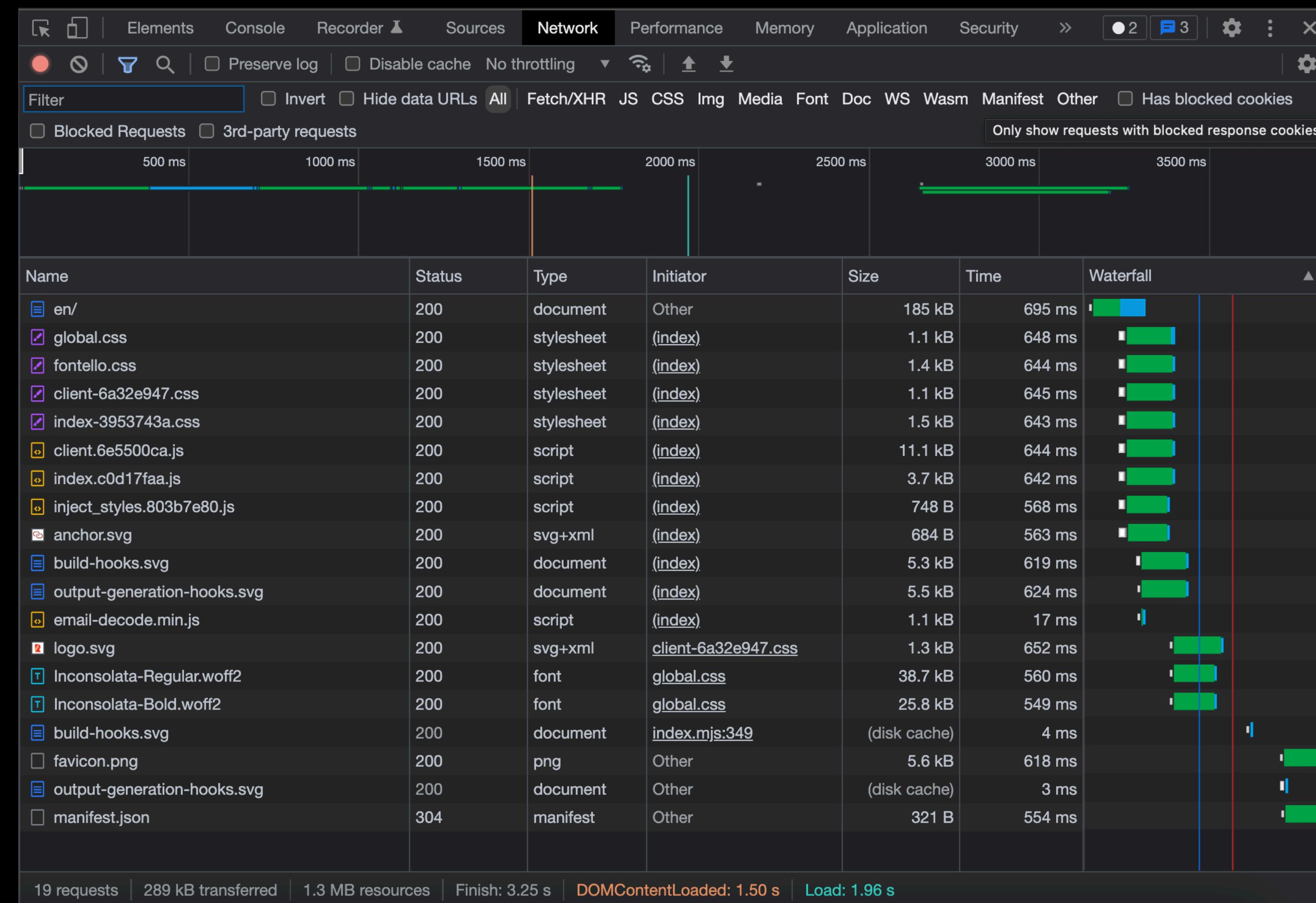


# Break!

# Performance

# Performance

What ?



# Performance

What ?

Web performance refers to how quickly site content **loads and renders** in a web browser, and how well it responds to user interaction. Bad performing sites are **slow to display** and **slow to respond to input**. Bad performing sites increase site abandonment.

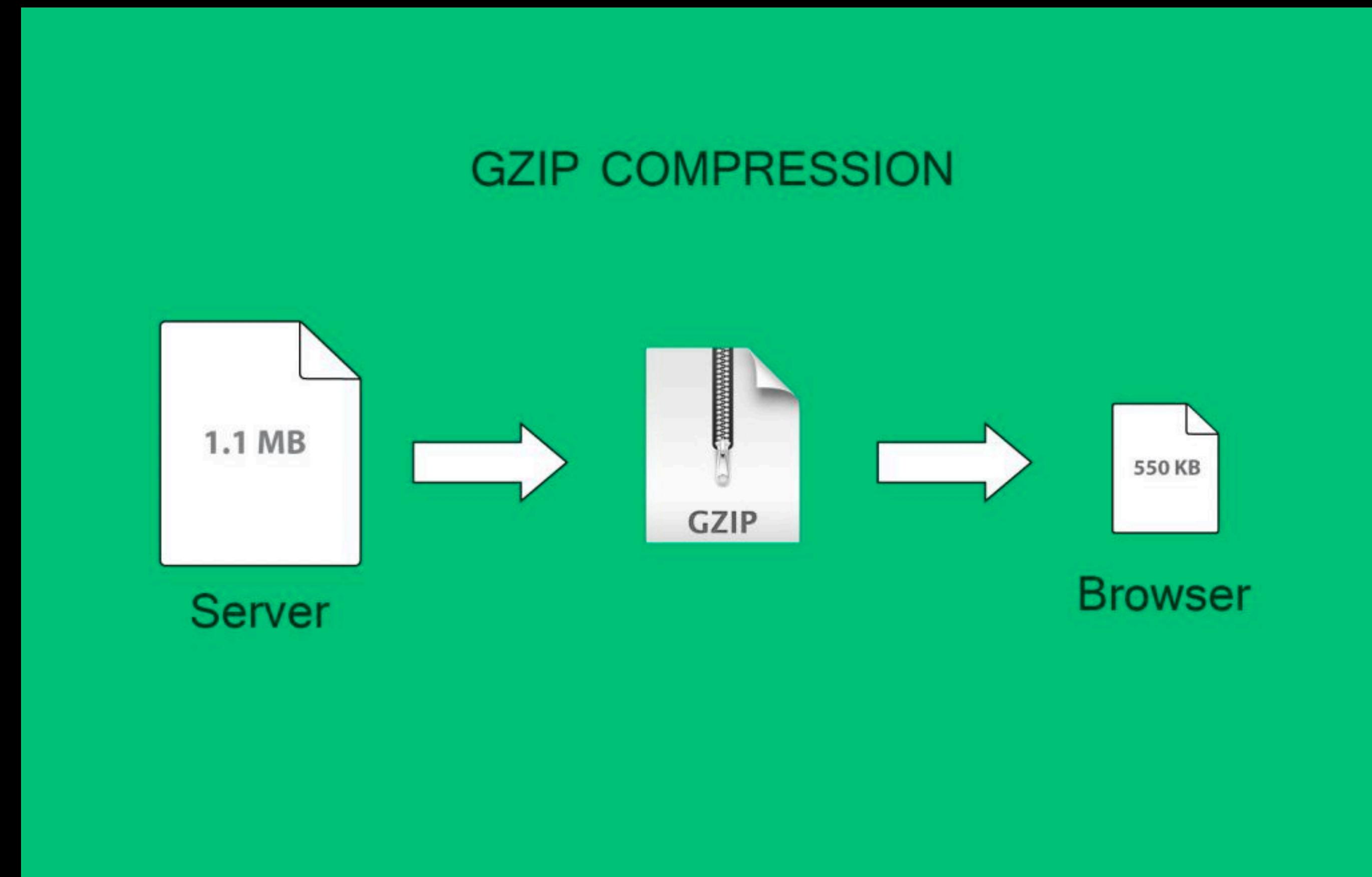
# Performance

What?

- Minimize assets (img / css / js / html)
- Minify requests (img / css / js / ???)
- Remove “render-blocking” resources
- Use caching appropriately
- Compress assets (gzip)

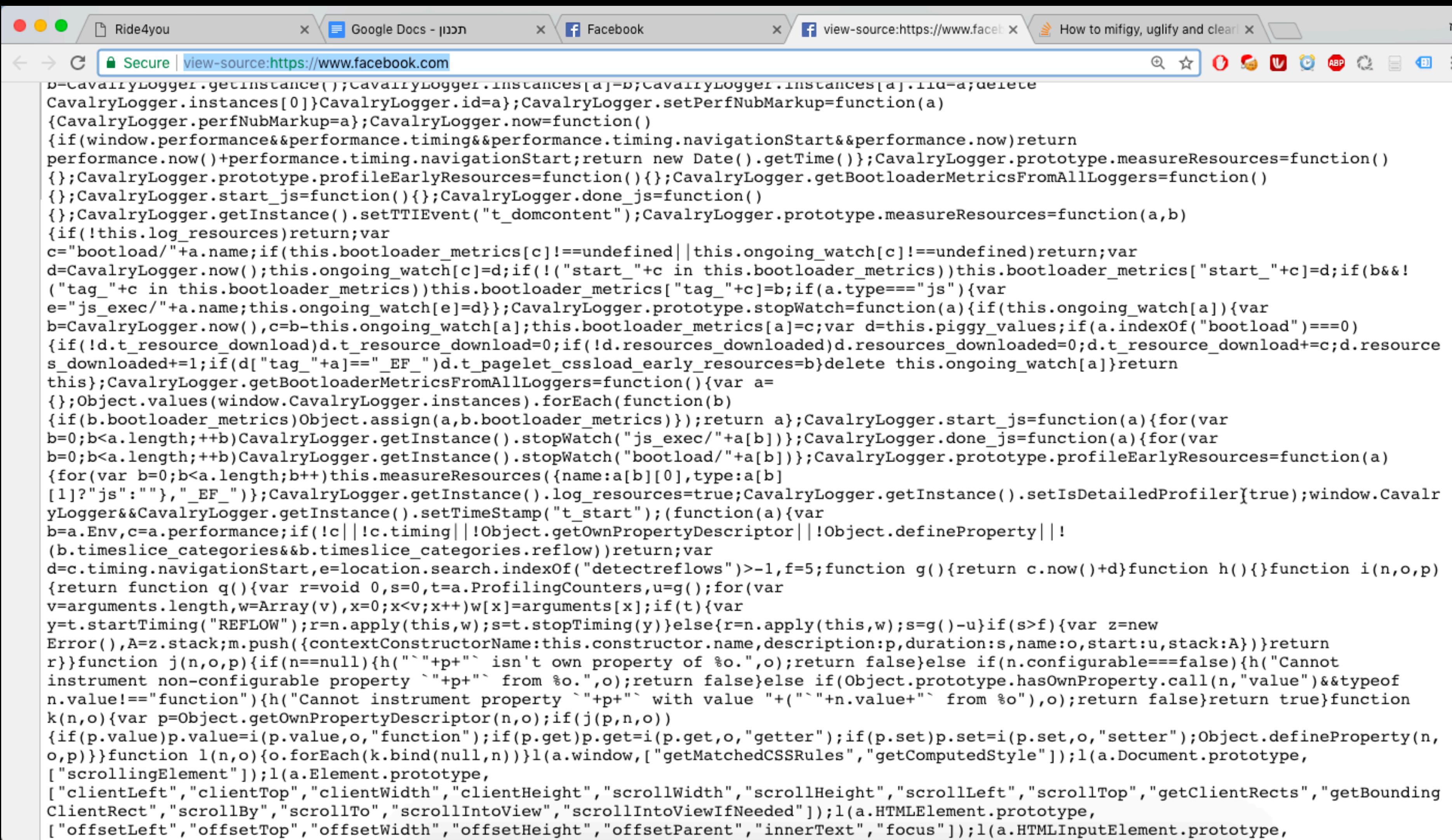
# Performance

What?



# Performance

Gzip



The screenshot shows a web browser window with multiple tabs open. The active tab is titled "Secure | view-source:https://www.facebook.com". The page content is filled with a massive amount of minified JavaScript code, which is the compressed version of the original source code. This minification is a key technique used by Facebook to reduce the size of their JavaScript files, making them faster to load and execute. The code is highly compressed, with many characters removed while maintaining the functionality of the script.

```
b=CavalryLogger.getInstance();CavalryLogger.instances[a]=b;CavalryLogger.instances[a].rid=a;delete CavalryLogger.instances[0]CavalryLogger.id=a};CavalryLogger.setPerfNubMarkup=function(a){CavalryLogger.perfNubMarkup=a};CavalryLogger.now=function(){if(window.performance&&performance.timing&&performance.timing.navigationStart&&performance.now)return performance.now()>performance.timing.navigationStart?new Date().getTime():CavalryLogger.prototype.measureResources=function(){};CavalryLogger.prototype.profileEarlyResources=function(){};CavalryLogger.getBootloaderMetricsFromAllLoggers=function(){};CavalryLogger.start_js=function(){};CavalryLogger.done_js=function(){};CavalryLogger.getInstance().setTTIEvent("t_domcontent");CavalryLogger.prototype.measureResources=function(a,b){if(!this.log_resources)return;var c="bootload/"+a.name;if(this.bootloader_metrics[c]==undefined||this.ongoing_watch[c]==undefined)return;var d=CavalryLogger.now();this.ongoing_watch[c]=d;if(!("start_"+c in this.bootloader_metrics))this.bootloader_metrics["start_"+c]=d;if(b&&("tag_"+c in this.bootloader_metrics))this.bootloader_metrics["tag_"+c]=b;if(a.type==="js"){var e="js_exec/"+a.name;this.ongoing_watch[e]=d};CavalryLogger.prototype.stopWatch=function(a){if(this.ongoing_watch[a]){var b=CavalryLogger.now(),c=b-this.ongoing_watch[a];this.bootloader_metrics[a]=c;var d=this.piggy_values;if(a.indexOf("bootload")==0){if(!d.t_resource_download)d.t_resource_download=0;if(!d.resources_downloaded)d.resources_downloaded=0;d.t_resource_download+=c;d.resources_downloaded+=1;if(d["tag_"+a]=="_EF_")d.t_pagelet_cssload_early_resources=b}delete this.ongoing_watch[a]}return this};CavalryLogger.getBootloaderMetricsFromAllLoggers=function(){var a=Object.values(window.CavalryLogger.instances).forEach(function(b){if(b.bootloader_metrics)Object.assign(a,b.bootloader_metrics)});return a};CavalryLogger.start_js=function(a){for(var b=0;b<a.length;++b)CavalryLogger.getInstance().stopWatch("js_exec/"+a[b]);CavalryLogger.done_js=function(a){for(var b=0;b<a.length;++b)CavalryLogger.getInstance().stopWatch("bootload/"+a[b]);CavalryLogger.prototype.profileEarlyResources=function(a){for(var b=0;b<a.length;b++)this.measureResources({name:a[b][0],type:a[b][1]?"js":""},"_EF_");CavalryLogger.getInstance().log_resources=true;CavalryLogger.getInstance().setIsDetailedProfiler(true);window.CavalryLogger&&CavalryLogger.getInstance().setTimeStamp("t_start");(function(a){var b=a.Env,c=a.performance;if(!c||!c.timing||!Object.getOwnPropertyDescriptor||!Object.defineProperty||!(b.timeslice_categories&&b.timeslice_categories.reflow))return;var d=c.timing.navigationStart,e=location.search.indexOf("detectrefflows")>-1,f=5;function g(){return c.now()+d}function h(){function i(n,o,p){return function q(){var r=void 0,s=0,t=a.ProfilingCounters,u=g();for(var v=arguments.length,w=Array(v),x=0;x<v;x++)w[x]=arguments[x];if(t){var y=t.startTiming("REFLOW");r=n.apply(this,w);s=t.stopTiming(y)}else{r=n.apply(this,w);s=g()-u}if(s>f){var z=new Error(),A=z.stack;m.push({contextConstructorName:this.constructor.name,description:p,duration:s,name:o,start:u,stack:A})}return r}}function j(n,o,p){if(n==null){h("`"+p+"` isn't own property of %o.",o);return false}else if(n.configurable==false){h("Cannot instrument non-configurable property `"+p+"` from %o.",o);return false}else if(Object.prototype.hasOwnProperty.call(n,"value")&&typeof n.value!="function"){h("Cannot instrument property `"+p+"` with value `"+n.value+"` from %o"),o);return false}return true}function k(n,o){var p=Object.getOwnPropertyDescriptor(n,o);if(j(p,n,o)){if(p.value)i(p.value,o,"function");if(p.get)p.get=i(p.get,o,"getter");if(p.set)p.set=i(p.set,o,"setter");Object.defineProperty(n,o,p)}}function l(n,o){o.forEach(k.bind(null,n));l(a.window,[ "getMatchedCSSRules","getComputedStyle"]);l(a.Document.prototype,[ "scrollingElement"]);l(a.Element.prototype,[ "clientLeft","clientTop","clientWidth","clientHeight","scrollWidth","scrollHeight","scrollLeft","scrollTop","getClientRects","getBoundingClientRect","scrollBy","scrollTo","scrollIntoView","scrollIntoViewIfNeeded"]);l(a HTMLElement.prototype,[ "offsetLeft","offsetTop","offsetWidth","offsetHeight","offsetParent","innerText","focus"]);l(a.HTMLInputElement.prototype,
```

# Performance

# Lazyloading

## The loading attribute

Today, Chrome already loads images at different priorities depending on where they're located with respect to the device viewport. Images below the viewport are loaded with a lower priority, but they're still fetched as soon as possible.

In Chrome 76+, you can use the `loading` attribute to completely defer the loading of offscreen images that can be reached by scrolling:

```

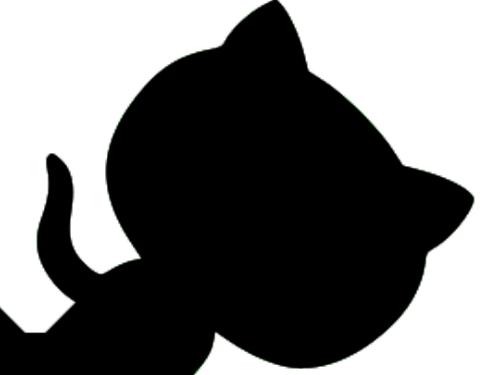
```

Here are the supported values for the `loading` attribute:

- `auto` : Default lazy-loading behavior of the browser, which is the same as not including the attribute.
- `lazy` : Defer loading of the resource until it reaches a [calculated distance](#) from the viewport.
- `eager` : Load the resource immediately, regardless of where it's located on the page.

# Assignment (30 min)

Look into performance / optimisation. Where could your application benefit from it? Can you optimise images on uploading? Lazyload other images? Lazyload fonts? Minify HTML/CSS/JavaScript? Write down your research and include it in your wiki, and if possible, include it in your application.



# Demo

see you in **next** **Q&A** **session.**

**exit**

