

project-tech

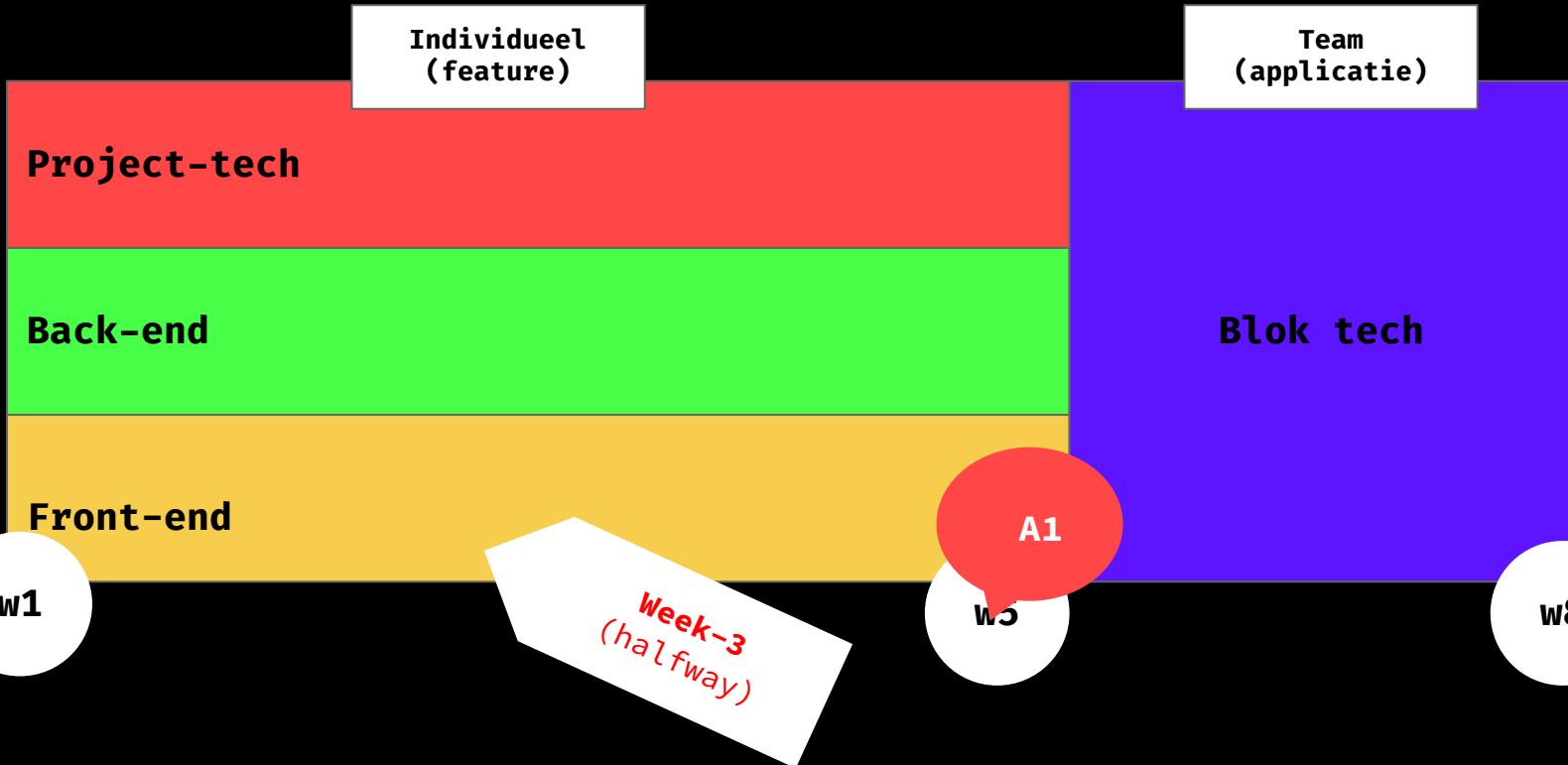
Env && Linting && Build

lab 3/8



Show what
you did

Stand-up!



Back Computer Mobile

🚀 Tussentijdse Feedback - pt() - tech_2

👉 We zijn nu een paar weekjes aan de slag met Project-tech. Ik wil heel graag weten wat je mening over dit vak tot nu toe is. Er staan vragen in over het vak maar ook over mij als docent. Je helpt mij maar ook alle andere docenten er heel erg mee als je even de tijd neemt om dit in te vullen!

Elke sectie heeft een comments tabje, gebruik het voor extra opmerkingen als je iets kwijt wilt! Je kan ervoor kiezen om dit anoniem te doen (door niet je naam in te voeren) maar ik zou het fijn vinden als je dat wel doet zodat ik je eventueel kan bereiken en vragen kan stellen over opmerkingen en feedback die je hebt gegeven.

En nog een klein dingetje; als er iets niet goed is probeer niet teveel te 'ranten' en te 'externaliseren' probeer het opbouwend te houden. 'Ik vind het stom' daar heb ik niet zoveel aan ;-)' laat dan ook even weten wat er stom is,

Maar liever natuurlijk alleen maar positieve dingen 😊

Thanks!

- Danny

Hi Danny, when you submit this form, the owner will be able to see your name and email address.

* Required

1. Hoi! Wie ben jij?

Enter your answer

Geef je feedback!

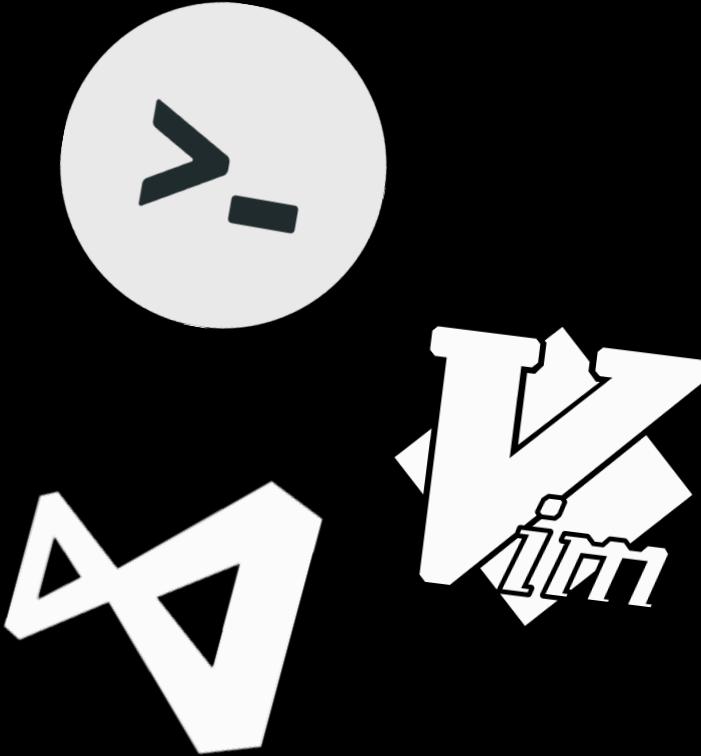
today

I. Standup

II. Local Dev environment

III. Linters & Formatters

IV. Build Tools



Dev Environment





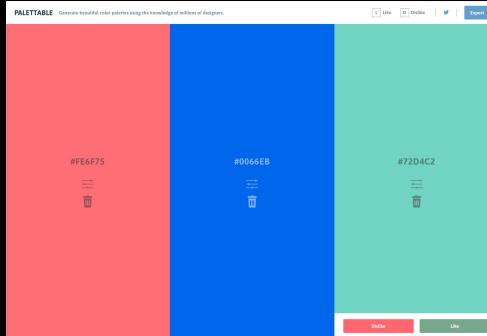
Bucketlist

1. Code Editors

2. Linters/Formatters

(typescript-valin/explicit-function-return-type
typescript-prettier/prettier
typescript-prettier/prettier)

3. Build Tools



4. Non-code tools

The image shows two side-by-side code editors on a Mac OS X desktop. A blue arrow points from the left editor to the right editor, indicating a comparison or migration process.

Left Editor (files.js):

```
async function getCount(context) {
  const count = await apiV8.get("/files", {
    params: {
      limit: 1,
      meta: "total_count",
    },
  });
  context.fileCount = count.data.meta.total_count;
  context.fileMap = {};
}

async function uploadFiles(context) {
  const pages = Math.ceil(context.fileCount / 100);

  const tasks = [];
  for (let i = 0; i < pages; i++) {
    tasks.push({
      title: `Uploading files ${i * 100 + 1}-${(i + 1) * 100}`,
      task: uploadBatch(i),
    });
  }
  return new Listr(tasks);
}

function uploadBatch(page) {
  return async (context, task) => {
    const records = await apiV8.get("/files", {
      params: {
        offset: page * 100,
        limit: 100,
      },
    });

    for (const fileRecord of records.data.data) {
      task.output = fileRecord.filename_download;

      const savedFile = await apiV9.post("/files/import", {
        url: fileRecord.data.full_url,
        data: {
          filename_download: fileRecord.filename_download,
          title: fileRecord.title,
          description: fileRecord.description,
        },
      });
      context.fileMap[fileRecord.id] = savedFile.data.data.id;
    }
  };
}
```

Right Editor (migration-tool/files.js):

```
async function getCount(context) {
  const count = await apiV8.get("/files", {
    params: {
      limit: 1,
      meta: "total_count",
    },
  });
  context.fileCount = count.data.meta.total_count;
  context.fileMap = {};
}

async function uploadFiles(context) {
  const pages = Math.ceil(context.fileCount / 100);

  const tasks = [];

  for (let i = 0; i < pages; i++) {
    tasks.push({
      title: `Uploading files ${i * 100 + 1}-${(i + 1) * 100}`,
      task: uploadBatch(i),
    });
  }
  return new Listr(tasks);
}

function uploadBatch(page) {
  return async (context, task) => {
    const records = await apiV8.get("/files", {
      params: {
        offset: page * 100,
        limit: 100,
      },
    });

    for (const fileRecord of records.data.data) {
      task.output = fileRecord.filename_download;

      const savedFile = await apiV9.post("/files/import", {
        url: fileRecord.data.full_url,
        data: {
          filename_download: fileRecord.filename_download,
          title: fileRecord.title,
          description: fileRecord.description,
        },
      });
      context.fileMap[fileRecord.id] = savedFile.data.data.id;
    }
  };
}
```

A status bar at the bottom of the right editor indicates: uploadFilesLn 29, Col 38 +0,-0 JavaScript 0 2 Spaces ~

code

editors

- ❖ Code is “just” text
- ❖ Computers don’t care what your code
- ❖ Make writing (reading) source code easier
- ❖ Don’t work hard, work smart #tailopez
- ❖ Personal preference!



Visual Studio Code



Brackets



WebStorm



Sublime Text



nova™



ATOM

Visual Studio



GNU Emacs



Xcode 12

.eleventy.js — personal-website

EXPLORER FILES TERMINAL

PEERCLIPBOARD

> _includes

> _layouts

> _sass

> _css

> node_modules

> pages

> posts

> scripts

> static

↳ .browserlistrc

⚙ .editorconfig

JS .eleventy.js 1

↳ .eleventyignore

↳ .env.sample

⚙ .eslintrc.js

↳ .gitignore

↳ 404.md

↳ index.njk

↳ sitemap.xml.njk

> TIMELINE

master* ↵ 0 △ 1

```
1 // Import packages
2 const path = require("path");
3 const fs = require("fs");
4 const markdownIt = require('markdown-it');
5 const markdownItAnchor = require("markdown-it-anchor"); 2.3K (gzipped: 1.1K)
6 const { DateTime } = require("luxon")
7
8 // Eleventy plugins
9 const pluginRss = require("@11ty/eleventy-plugin-rss");
10 const eleventyNavigationPlugin = require("@11ty/eleventy-navigation"); 5.8K (gzipped: 2K)
11 const readingTime = require('eleventy-plugin-reading-time'); 781 (gzipped: 488)
12 const Image = require("@11ty/eleventy-img");
13
14 // Import filters
15 const randomLink = require("./libs/randomLink.js");
16
17
18 module.exports = {
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL

personal-website on 🏙 master [!] is 📦 v1.0.0 via ● v15.3.0

1: zsh

Spaces: 2 JavaScript ⚡ Go Live Prettier

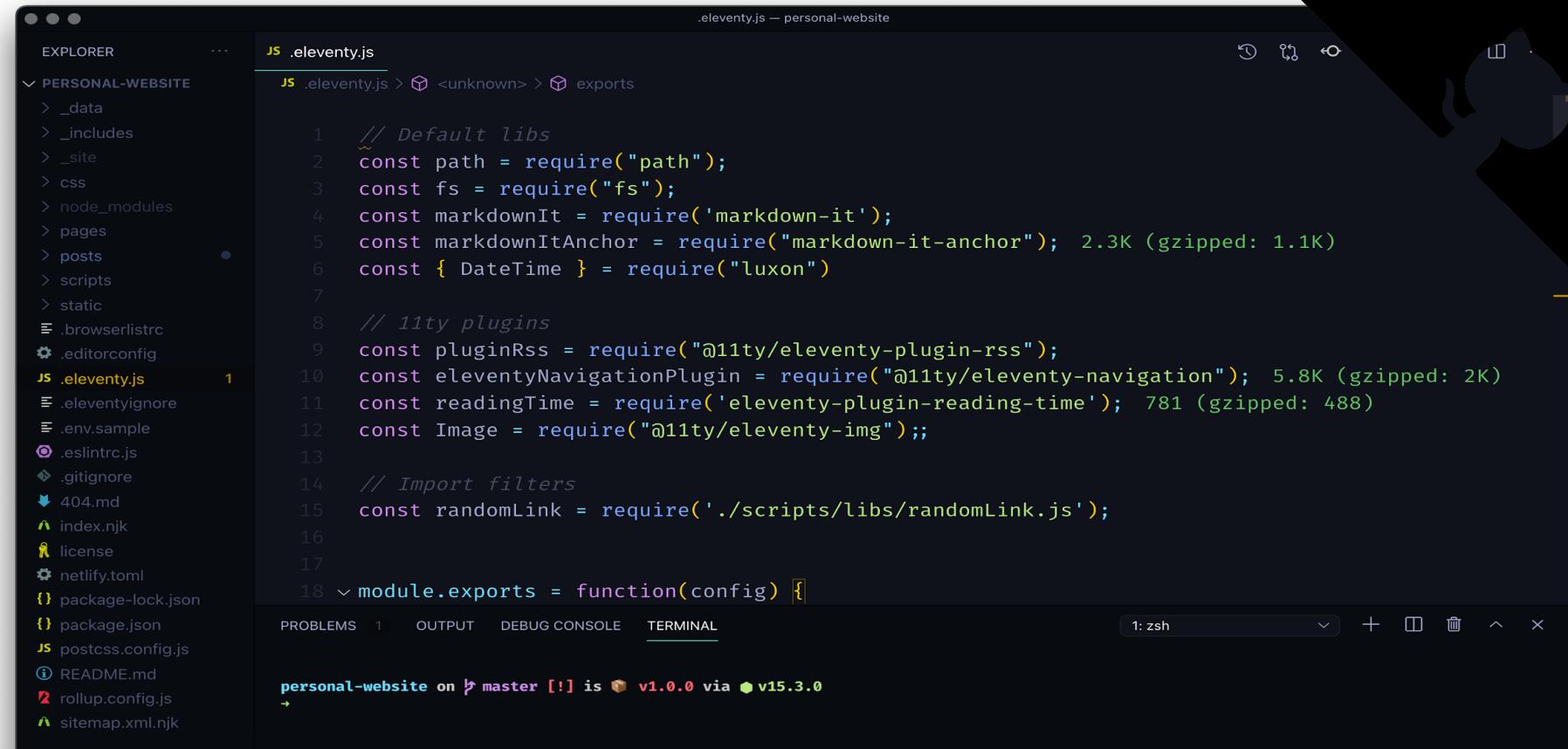
Extensions

Coding font

Prompt

Syntax theme

settings.json



.eleventy.js — personal-website

EXPLORER ... JS .eleventy.js

PERSONAL-WEBSITE

- > _data
- > _includes
- > _site
- > css
- > node_modules
- > pages
- > posts
- > scripts
- > static
- browserlistrc
- editorconfig
- JS .eleventy.js 1
- eleventyignore
- env.sample
- eslintrc.js
- gitignore
- 404.md
- index.njk
- license
- netlify.toml
- package-lock.json
- package.json
- postcss.config.js
- README.md
- rollup.config.js
- sitemap.xml.njk

.eleventy.js > <unknown> > exports

```
1 // Default libs
2 const path = require("path");
3 const fs = require("fs");
4 const markdownIt = require('markdown-it');
5 const markdownItAnchor = require("markdown-it-anchor"); 2.3K (gzipped: 1.1K)
6 const { DateTime } = require("luxon")
7
8 // 11ty plugins
9 const pluginRss = require("@11ty/eleventy-plugin-rss");
10 const eleventyNavigationPlugin = require("@11ty/eleventy-navigation"); 5.8K (gzipped: 2K)
11 const readingTime = require('eleventy-plugin-reading-time'); 781 (gzipped: 488)
12 const Image = require("@11ty/eleventy-img");
13
14 // Import filters
15 const randomLink = require('./scripts/libs/randomLink.js');
16
17
18 module.exports = function(config) {
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL 1: zsh

```
personal-website on ✨ master [!] is 🎉 v1.0.0 via ● v15.3.0
→
```

live demo **dev environment**

Linters & Formatters

linters

?

A **linter** catches inconsistencies throughout your project and reduces your *chances of making logical errors*.

Syntax: Linters check the source code for programmatic mistakes (bugs, errors)

formatter

?

A **formatter** can help maintain a *consistent coding style*. It's basically a 'style guide' for how you write code.

Stylistic: beautifies (and formats) your code to make it more readable.

Linters & formatters

why

Ensuring a good coding style has a couple upsides:

- It helps you avoid bugs
- Help other developers read through your code
- Save time and avoid stress
- Well organized, cleanly written code can be its own documentation!
- Improves team collaboration

linter example

- ❖ Checks for a wide range of syntax- and stylistic errors
- ❖ Can help you write better code by enforcing best practices
- ❖ Makes sure everybody codes in the same “dialect”

<https://eslint.org>

ESlint

enforce a maximum depth that blocks can be nested (max-depth)

Many developers consider code difficult to read if blocks are nested beyond a certain depth.

This rule enforces a maximum depth that blocks can be nested to reduce code complexity.

ⓘ Examples of incorrect code for this rule with the default `{"max": 4}` option:

```
/*eslint max-depth: ["error", 4]*/
/*eslint-env es6*/

function foo() {
  for (;;) { // Nested 1 deep
    while (true) { // Nested 2 deep
      if (true) { // Nested 3 deep
        if (true) { // Nested 4 deep
          if (true) { // Nested 5 deep
            }
          }
        }
      }
    }
}
```

ⓘ Examples of correct code for this rule with the default `{"max": 4}` option:

```
/*eslint max-depth: ["error", 4]*/
/*eslint-env es6*/

function foo() {
  for (;;) { // Nested 1 deep
    while (true) { // Nested 2 deep
      if (true) { // Nested 3 deep
        if (true) { // Nested 4 deep
          }
        }
      }
    }
}
```

formatter

example

- ❖ Makes sure everybody's **editor settings are the same**
- ❖ Ensures you never end up **with mixed tabs/spaces**

<https://editorconfig.org>

editorconfig

```
# EditorConfig is awesome:  
https://EditorConfig.org  
  
# top-most EditorConfig file  
root = true  
  
# Unix-style newlines with a newline ending  
every file  
[*]  
end_of_line = lf  
insert_final_newline = true  
  
# Matches multiple files with brace expansion  
notation  
# Set default charset  
[*.js,py]  
charset = utf-8  
  
# 4 space indentation  
[*.py]  
indent_style = space  
indent_size = 4  
  
# Tab indentation (no size specified)  
[Makefile]  
indent_style = tab  
  
# Indentation override for all JS under lib  
directory  
[lib/**.js]  
indent_style = space  
indent_size = 2
```

linters

configuration

Linters & formatters **need configuration files to tell them what rules to enforce.**
(*.rc = runtime configuration*)

- ❖ .editorconfig
- ❖ .eslintrc
- ❖ .stylelint

- ≡ .browserlistrc
- ⚙ .editorconfig
- js .eleventy.js
- ≡ .eleventyignore
- ≡ .env.sample
- ⌚ .eslintrc.js
- ❖ .gitignore

linters

how to use

- ❖ In **an extension** in your text editor
- ❖ **Using commands** in the CLI
- ❖ **Run scripts** in package.json

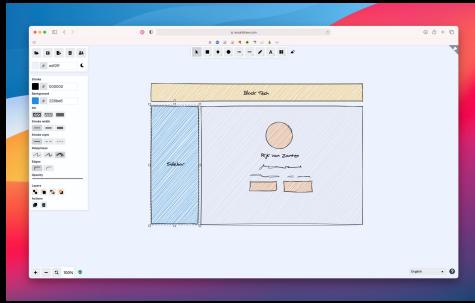
Danny de Vries, 2 years ago | 1 author (Danny de Vries)

```
1  module.exports = {  
2    env: {  
3      browser: true,  
4      commonjs: true,  
5      es6: true  
6    },  
7    extends: "eslint:recommended",  
8    rules: {  
9      'no-console': 1,  
10     },  
11   };
```

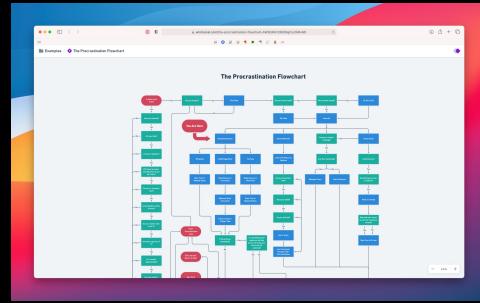
Danny de Vries, 2 years ago • Initial co...

live demo **linters & formatters**

non-code tools

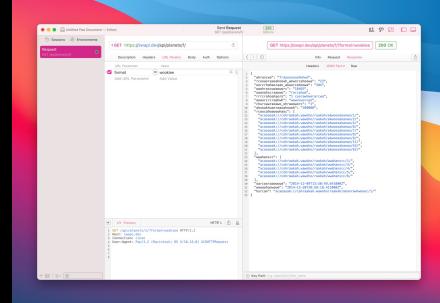


Excalidraw



Whimsical

applications



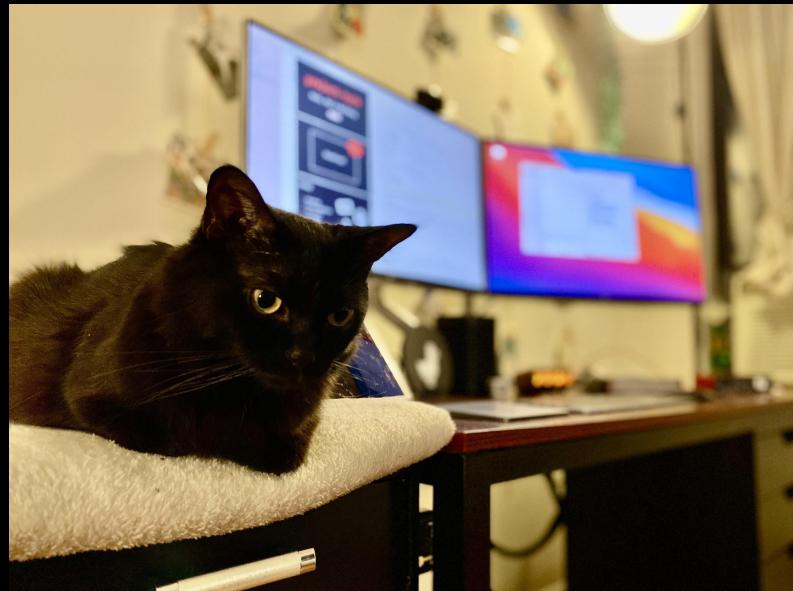
Postman

And many many more ...

non-code tools

healthy

- ❖ #stayhydrated
- ❖ Get enough sleep
- ❖ Reduce your brightness
- ❖ Make your workspace comfortable
- ❖ Take enough breaks



Build Tools

build tools

?

Build tools are additional tooling you can use in your project and are mostly used for ***automating tasks or transforming code.***

preprocessors, bundlers, postprocessor, compiling #buzzword #buzzword, ai, machine learning

build tools

?

Build tools are additional tooling you can use in your project and are mostly used for **automating tasks or transforming code.**

preprocessors, bundlers, postprocessor,

Note: If you don't know what you are doing and build tools feel too much. **Then don't ;-)**
But zwarte piste go ahead.

example

sass

.scss gets processed
(compiled) to .css

SCSS

```
1 section {  
2     height: 100px;  
3     width: 100px;  
4     .class-one {  
5         height: 50px;  
6         width: 50px;  
7         .button {  
8             color: #074e68;  
9         }  
10    }  
11 }  
12 }  
13 }
```

CSS

```
1 section {  
2     height: 100px;  
3     width: 100px;  
4 }  
5 section .class-one {  
6     height: 50px;  
7     width: 50px;  
8 }  
9 section .class-one .button {  
10    color: #074e68;  
11 }  
12 }  
13 }
```

example

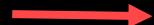
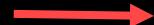
build

// Input

// Build

// Output

```
css/  
└── typography.scss  
└── layout.scss  
└── colors.scss
```



```
css/  
└── style.css
```

building

Start building out the interface (html & css) of your matching-app feature.

Synopsis

- Time: 6:00h
- Due: before week 3

Assignment

Based on the concept, job story, requirement list and wireframe from the previous week start building out the front-end of the feature you are going to make for the matching-application. You can build the interface with the **templating engine** as you learned in back-end as opposed to 'plain' HTML & CSS.

- Turn your wireframe **into HTML pages**. Do a HTML breakdown of your wireframe and use semantic HTML as learned in previous courses.

work on **local, linting, build**

exit ;

see you in lab-4 !