

# project-tech

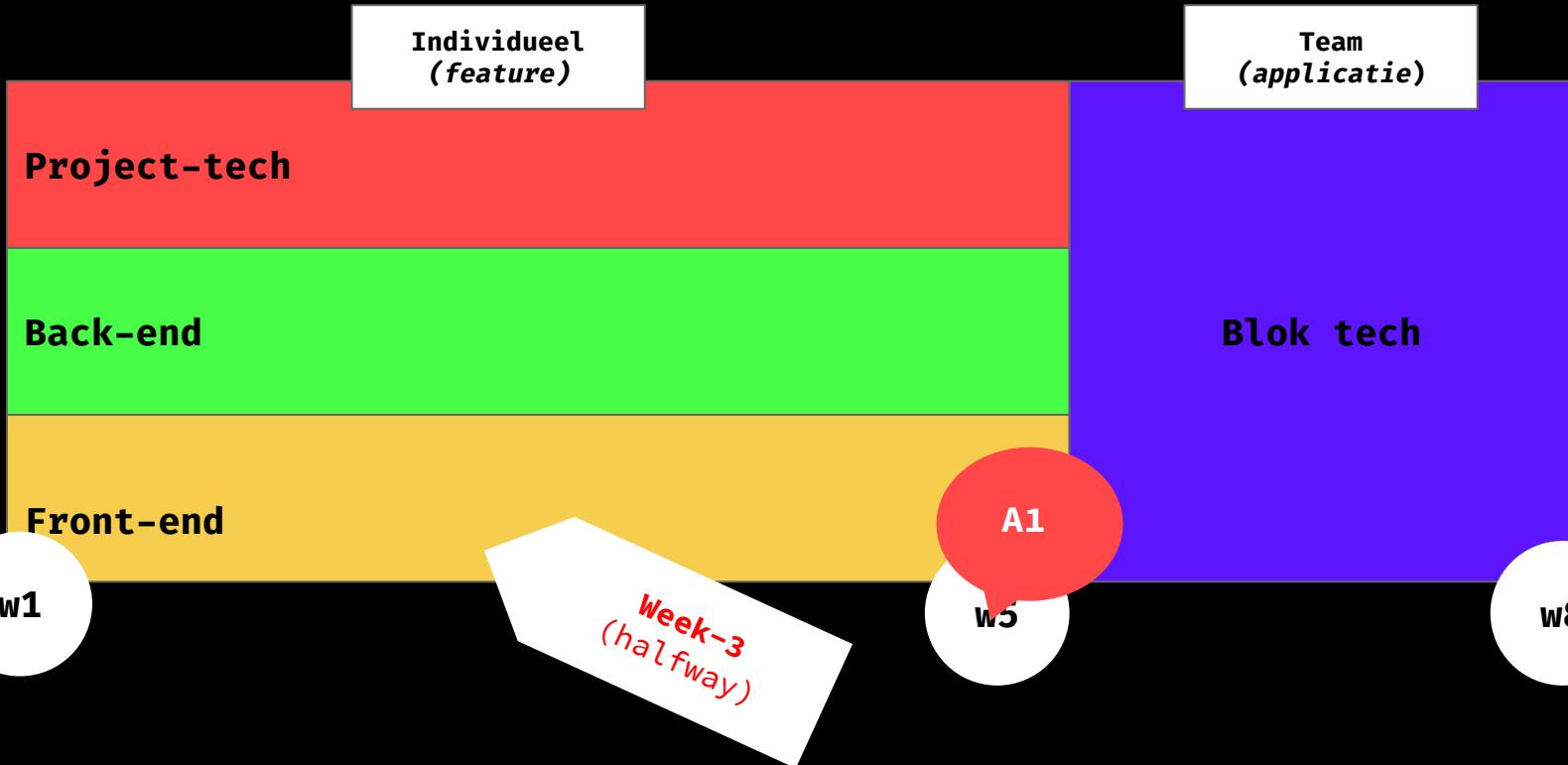
Env & Linting & Build

lab 3/8



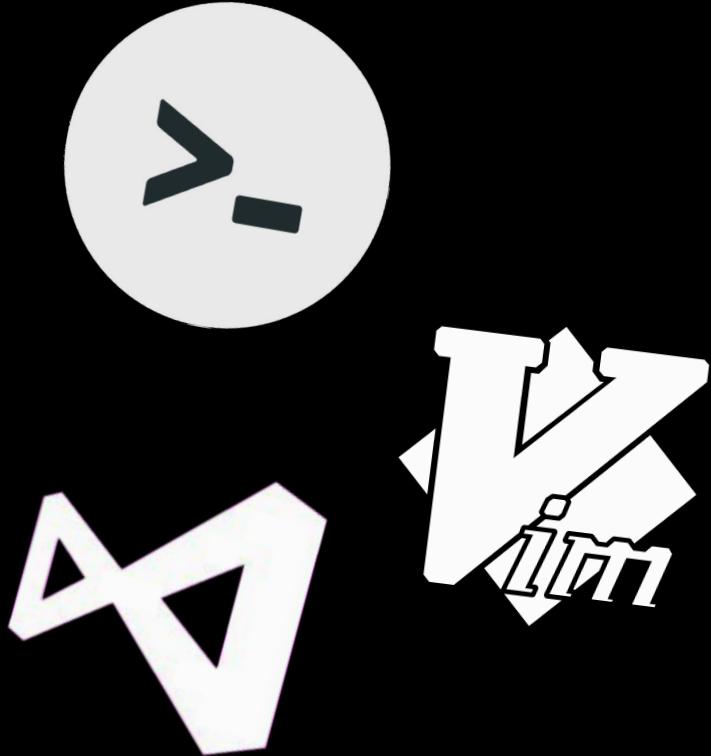
*Show what  
you did*

# Stand-up!



# today

- I. Standup
- II. Local Dev environment
- III. Linters & Formatters
- IV. Build Tools



# Dev Environment

$$g(1) = Y + u(1)$$

PRO-1039

C-<sup>2014</sup>  
2014-2015

卷之三

## **FORTRAN STATEMENT**

EDUCATION



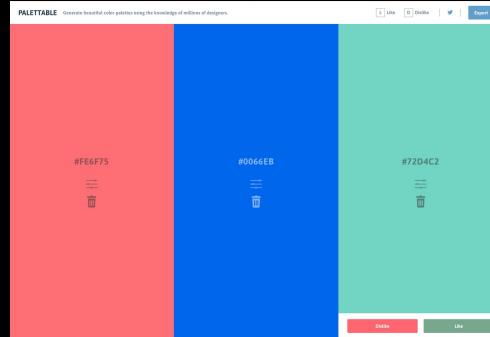


Bucketlist

## 1. Code Editors

## 2. Linters/Formatters

## 3. Build Tools



## 4. Non-code tools

The image shows two side-by-side code editors on a Mac OS X desktop. Both editors are displaying the same file, `files.js`, which contains JavaScript code for file management.

**Left Editor (Nova):**

```
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 = {};
}

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 (Nova):**

```
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 blue arrow points from the left editor to the right editor, indicating a comparison or migration process between the two versions of the code.

# 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



ATOM



nova™

Visual Studio



GNU Emacs



Xcode 12

A screenshot of a dark-themed code editor (VS Code) showing an Eleventy configuration file (`eleventy.js`). The file contains code for setting up the build process, including imports for `path`, `fs`, `markdownIt`, `luxon`, `eleventy` plugins like `rss` and `navigation`, and filters like `randomLink`. The code editor interface includes a sidebar with project files, a bottom navigation bar with tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL, and a status bar at the bottom.

The image features four red speech bubbles with white text:

- Extensions**: Points to the sidebar showing files like `.env.sample`, `.eslintrc`, and `.gitignore`.
- Coding font**: Points to the code editor's main area where the code is displayed.
- Prompt**: Points to the terminal tab in the bottom navigation bar.
- Syntax theme**: Points to the bottom navigation bar, likely referring to the dark theme.
- settings.json**: Points to the status bar at the bottom right.

```
const path = require("path");
const fs = require("fs");
const markdownIt = require('markdown-it');
const markdownItAnchor = require("markdown-it-anchor"); 2.3K (gzipped: 1.1K)
const { DateTime } = require("luxon")

// 11ty plugins
const pluginRss = require("@11ty/eleventy-plugin-rss");
const eleventyNavigationPlugin = require("@11ty/eleventy-navigation"); 5.8K (gzipped: 2K)
const readingTime = require('eleventy-plugin-reading-time'); 781 (gzipped: 488)
const Image = require("@11ty/eleventy-img");;

// Import filters
const randomLink = require("./libs/randomLink.js");

module.exports = {
  dir: {
    input: "src",
    output: "dist"
  },
  path,
  fs,
  markdownIt,
  DateTime,
  pluginRss,
  eleventyNavigationPlugin,
  readingTime,
  Image,
  randomLink
};
```

personal-website on 🐧 master [!] is 🏃 v1.0.0 via 🖥 v15.3.0

Spaces: 2 JavaScript Go Live Prettier

.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  
JS postcss.config.js  
ⓘ README.md  
☒ rollup.config.js  
▲ sitemap.xml.njk

// Default libs

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

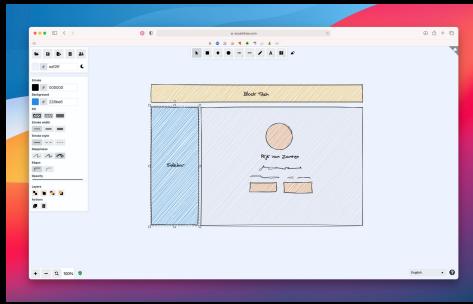
```
1  module.exports = {
2    env: {
3      browser: true,
4      commonjs: true,
5      es6: true
6    },
7    extends: "eslint:recommended",
8    rules: {
9      'no-console': 1,
10     },
11   };
```

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

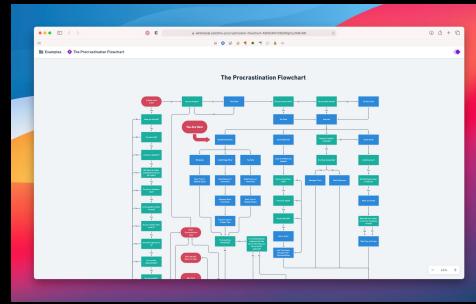
live demo **linters & formatters**

# non-code tools

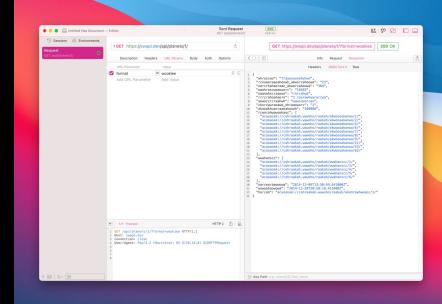
# applications



**Excalidraw**



**Whimsical**



**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

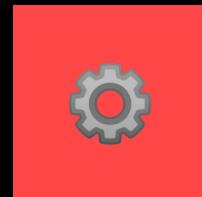
# build

// 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!