

PT()

LAB IV/VII

SCHEDULE

Lab II

- ❖ Command line
- ❖ Local machine
- ❖ Branding
- ❖ Job Stories

Lab IV (today)

- I. Feedback
- II. Coding Standards
- III. Linters
- IV. Extensions
- V. Requirements list
- VI. Homework

Assignments

- ❖ Linting
- ❖ Extensions
- ❖ Research

FEEDBACK

I/VI

RESEARCH

WIKI

The screenshot shows a web browser window displaying a GitHub Wiki page. The browser's address bar shows the URL `https://github.com/Sandroost/blok-tech/wiki`. The GitHub navigation bar at the top includes a search field, links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore', and user profile icons. The repository name 'Sandroost / blok-tech' is displayed, along with statistics: 1 Watch, 0 Stars, and 0 Forks. Below this, a tabbed interface shows 'Code', 'Issues 2', 'Pull requests 0', 'Projects 0', 'Wiki' (selected), and 'Insights'. The main content area is titled 'Home' and includes a note: 'Sander edited this page 2 days ago · 4 revisions'. The text on the page reads: 'Welkom op mijn Wiki van blok tech.' followed by 'Hier vind je mijn voortgang en documentatie van het project.' and a paragraph about the project's goal: 'De bedoeling van dit project is om een functionaliteit uit te werken van een datingsite. Nadat we deze hebben uitgewerkt gaan we samenwerken als groep om dit tot 1 mooi geheel te samen te voegen.' On the right side, there is a 'Pages 4' section with a search bar 'Find a Page...' and a list of links: 'Home', 'Concurrentie Dating apps', 'Job story', and 'Npm Packages'. At the bottom right, a 'Clone this wiki locally' section provides the URL `https://github.com/Sander` and a clone icon.

People are behind, pick up the pace.

RESEARCH

WIKI

- ❖ Job Stories are too generic
- ❖ Not many started working on the static HTML. You can combine that with the templating for this week.
- ❖ No branding research
- ❖ Many people focus on the **account page**
- ❖ Draw conclusions from your research

CODING STANDARDS

II/VI

II CODING STANDARDS

QUALITY

Code quality: how do we define “bad” code?

- The code looks overly complex, unlogical and messy. It is not consistent and worse of all: it doesn't work (worst case).
- The code looks clean and well organised, but doesn't work.
- The code looks overly complex, unlogical and messy, but everything (seems to) work fine.
- The code looks clean, logical and consistent. It is well organised and appropriately documented (ideal case).

II CODING STANDARDS

QUALITY

In conclusion: "bad" code can mean multiple things (simultaneously). Code can look horrible (to a developer), but still do what it is supposed to do. Is this bad code?

II CODING STANDARDS

STYLE

Why is maintaining a good coding style important?

- To avoid hard to catch / deeply nested errors and bugs as much as possible.
- So that other developers can easily understand what your code does (and doesn't do).
- To save time and avoid stress (bug solving, refactoring).
- Well organised, clean written code **is** your documentation!

LINTERS

III/VI

III LINTERS

?

Lint, or a linter, is a tool that **analyzes source code to flag programming errors**, bugs, stylistic errors, and suspicious constructs.

[wikipedia.org](https://en.wikipedia.org/wiki/Lint_(software))

III LINTERS

?

[...] the tools can also be used as simple **debuggers for common errors** (e.g. syntactic discrepancies) as well as hard-to-find errors

[wikipedia.org](https://en.cppreference.com/w/cpp/string/basic/basic_string_view)

III LINTERS

?

Linting is the process of checking the source code for **programmatic** as well as **stylistic errors**. Furthermore, a linter can help with maintaining a **consistent** coding style throughout your project and reduces your chances of making logical errors.

III LINTERS

TYPES

```
var foo = 'string' ;
```

```
var foo = 'string'
```

III LINTERS

TYPES

```
var foo = 'string' ;
```

```
var foo = 'string'
```



Stylistic?

III LINTERS

SETUP

There are basically two types of setup:

- As an extension in your editor
- As an **npm** script in your `package.json`

You can also do both (recommended).

III LINTERS

CONFIGURE

[...]designed to be completely **configurable**, meaning you can turn off every rule and run only with basic syntax validation.

``.eslintrc`` file for example

III LINTERS

CONFIGS

static	setup server, rename 'src' to 'static', let server serve static files	2 days ago
view	Renamed 'view/modules' and 'view/pages'	5 hours ago
.editorconfig	Changed .editorconfig, installed Express and EJS, renamed main.js to ...	3 days ago
.gitignore	Rewrite .gitignore, update readme	7 days ago
LICENSE.md	Add license	7 days ago
README.md	Used EJS to dynamically render pages, put pages in 'pages' directory ...	2 days ago
index.js	Renamed 'view/modules' and 'view/pages'	5 hours ago
package-lock.json	Changed .editorconfig, installed Express and EJS, renamed main.js to ...	3 days ago
package.json	Changed .editorconfig, installed Express and EJS, renamed main.js to ...	3 days ago

`.editorconfig` file for example. Usually referred to as dotfiles.

Assignments

Linting (research)

- Research
- Time: 0:45h
- Goals: subgoal 5

Take about 0:45h to investigate what linting is and why you might want to look for examples.

- Linting is commonly used to ensure code quality and consistency that?
- What are common tools for 'linting'?
- How can you implement these tools in your code?
- What are the difference between different linting tools?

Homework

Linting (implementation)

- Homework
- Time: 3h
- Goals: subgoal 5

Based on the research you conducted setup a (or multiple) linters in your own project. Ask yourself which tools are useful for your own project. If you're not sure which one to pick, the ones below are solid choices:

- `eslint` - pluggable linting utility for JavaScript
- `stylelint` - modern linter that helps you avoid errors and enforce conventions in your styles



TAKE ABOUT 45M TO INVESTIGATE WHAT 'LINTING' IS

DEMO

LINTERS & CONFIGS

EXTENSIONS

IV / VI

IV EXTENSIONS

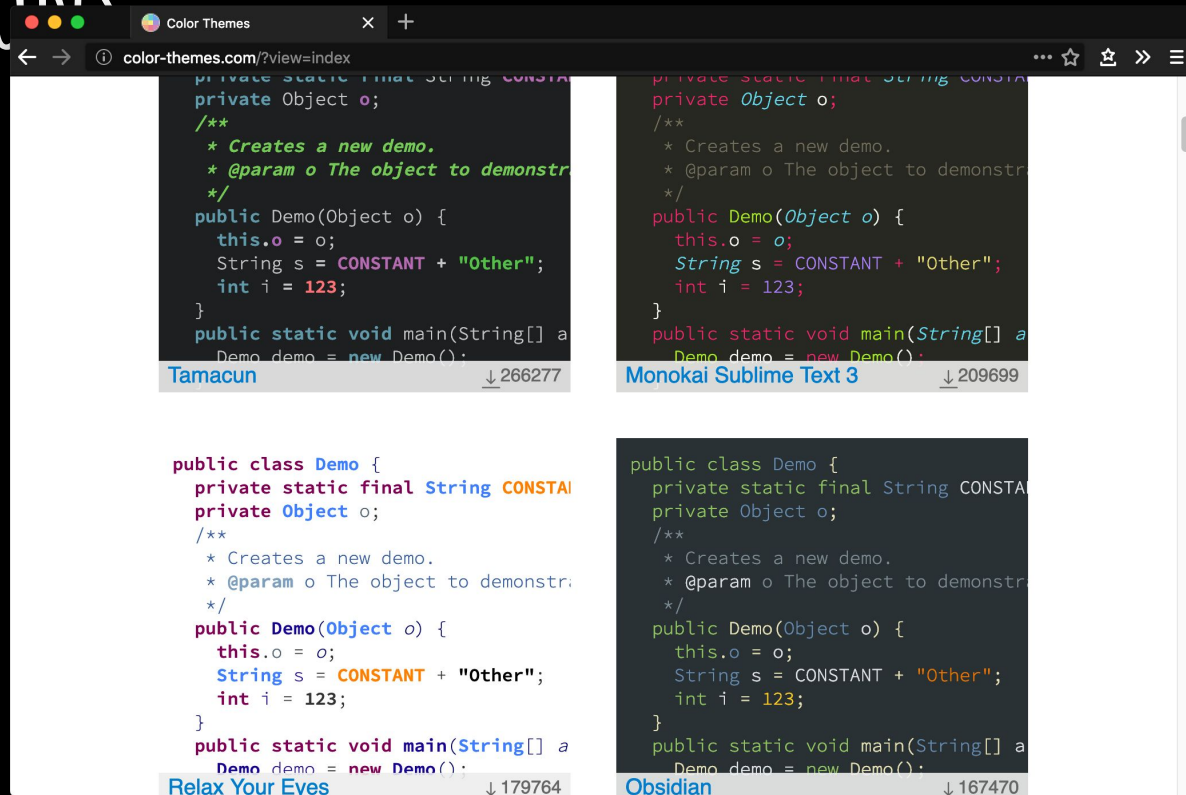
?

The features that Visual Studio Code or Atom includes **out-of-the-box** are just the start. VS Code extensions and Atom packages let you add languages, debuggers, and tools to your **installation to support your development workflow.**

<https://code.visualstudio.com/docs/editor/extension-gallery>

EXTENSIONS

COLOR THEMES



RESEARCH

V / VI

IV REQUIREMENTS LIST

METHODS

Narrow down your Job Story in specific **small functionalities** and make a list of requirements your feature should have. You can then apply the **MoSCoW method** on your list to see which functionalities are **must-haves** and which ones are **enhancements**.

HOMEWORK

VI/VI

Assignments

Linting (research)

- Research
- Time: 0:45h
- Goals: subgoal 5

Take about 0:45h to investigate why linting is used and how to look for examples.

- Linting is commonly used to ensure code quality and consistency.
- What are common tools for 'linting'?
- How can you implement these tools in your project?
- What are the difference between different linting tools?

LINTERS

Homework

Linting (implementation)

- Homework
- Time: 3h
- Goals: subgoal 5

Based on the research you conducted setup a (or multiple) linters in your own project. Ask yourself which tools are useful for your own project. If you're not sure which one to pick, the ones below are solid choices:

- `eslint` - pluggable linting utility for JavaScript
- `stylelint` - modern linter that helps you avoid errors and enforce conventions in your styles



WORK ON THE LINTING ASSIGNMENT

If you've configured a linter and want to use it in your editor, make sure you also install the corresponding extension.

Extensions

- Homework
- Time: 2h
- Goals: subgoal 5

Previously you've customized the look and feel of your command line. We are going to do the same thing with your code editor. Since you still using Brackets, this might be a good idea to look at [Atom](#) or [VS Code](#).

- Try a different editor. There are a lot of other communities out there.
 - In VS Code, you can find a list of extensions [here](#) is a good starting point.
- Install some extensions. [VS Code marketplace](#) and [Atom Packages](#) are the corresponding pages for extensions for both editors.
- Get the most out of your editor by tweaking your settings and getting to know the shortcuts. Take some time to tweak some settings and learn some keyboard shortcuts. You can read the [Flight Manual](#) for Atom or read the [docs](#) for VS Code.

Wes and Scott (you might have taken one of their courses) have a good podcast called Syntax where they detail their setups. You can listen to [these episodes](#) or look in the show notes for inspiration.

Usability Test

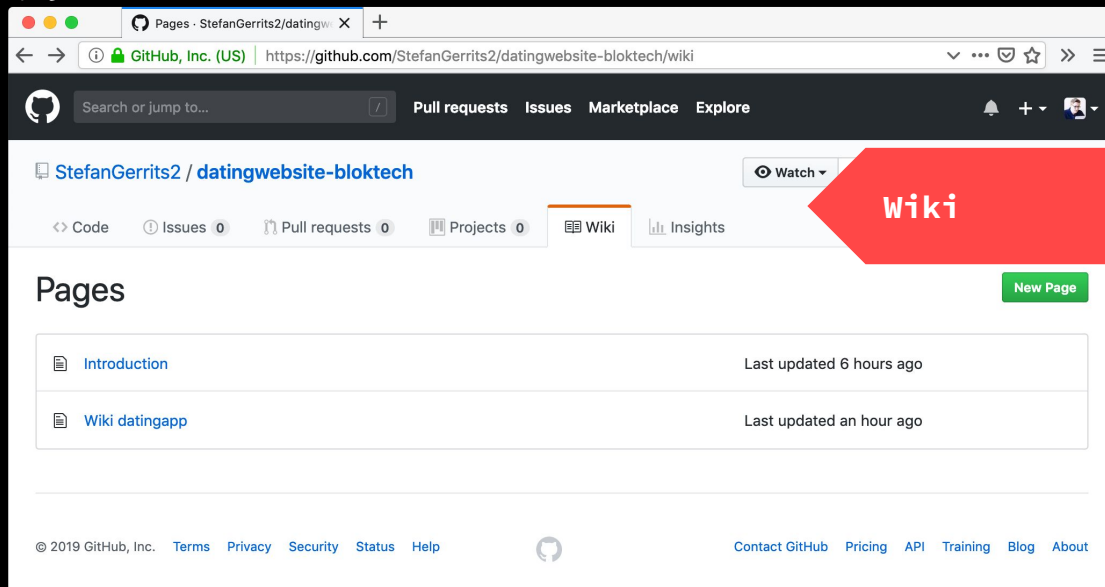
- Homework
- Time: 1h



WORK ON THE EXTENSIONS ASSIGNMENT

HAND-IN

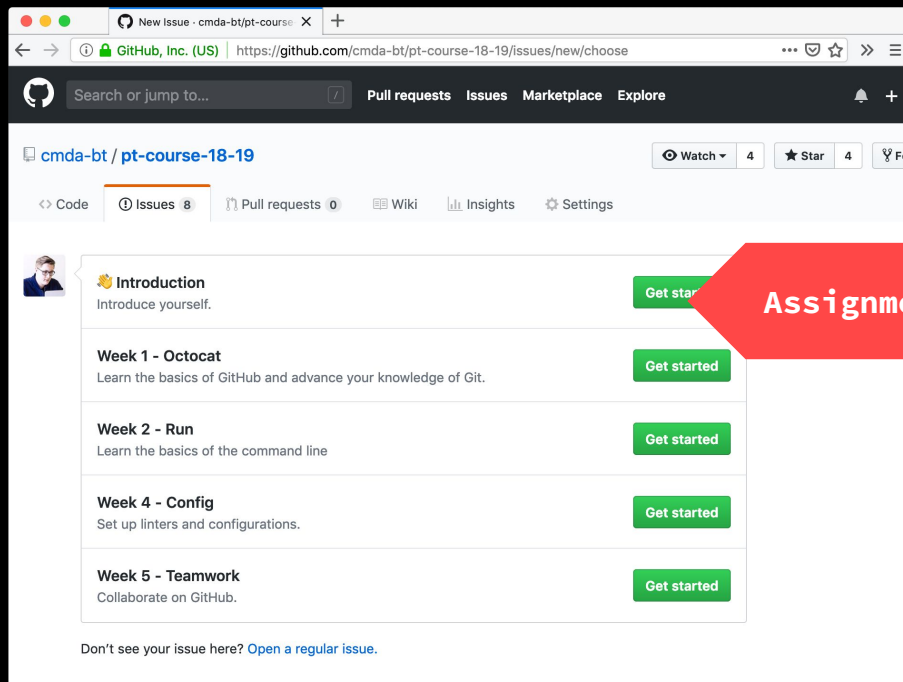
GITHUB



Update wiki!

HAND-IN

GITHUB



Assignments

Saturday - leet o'clock!

EXIT;

SEE YOU IN LAB V/VII!