front-end

HOF & Refactoring

lab 4/8

Stand-up!

Progressive enhancement

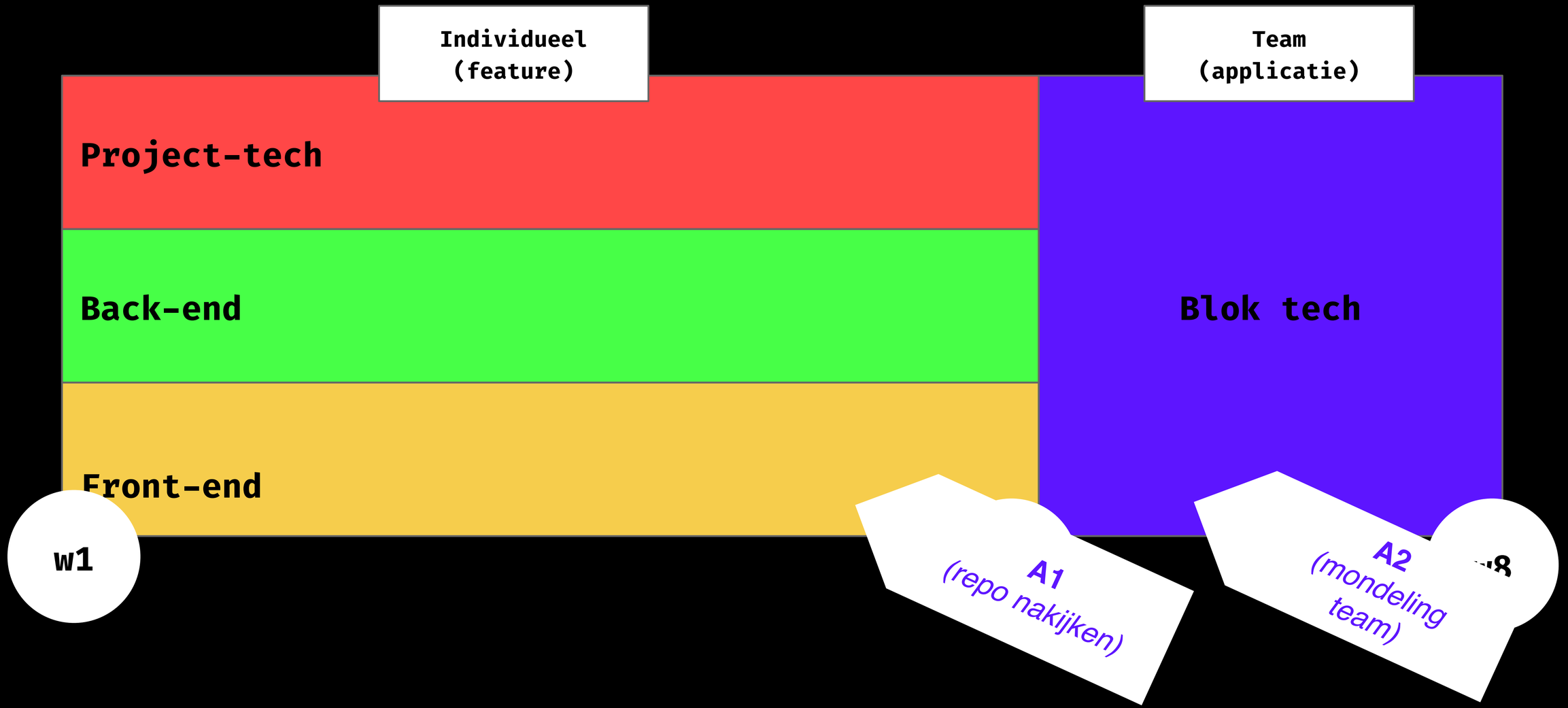
today

I. ~~Standup~~

II. Higher Order Functions

III. Coding Principles

IV. A1 explanation (+Q&A)


Individueel
(feature)
Team
(applicatie)
Project-tech
Back-end
Front-end
Blok tech
w1
A1
(repo nakijken)
A2
(mondeling
team)

Questions about…

?

- Events in JavaScript
- Functions & Scoping?
- The Document Object Model
- Web API’s

Higher Order Functions

Higher-order

Definition

Functions that operate on other functions, either by taking them as arguments or by returning them, are called higher-order functions.

Higher-order

Definition

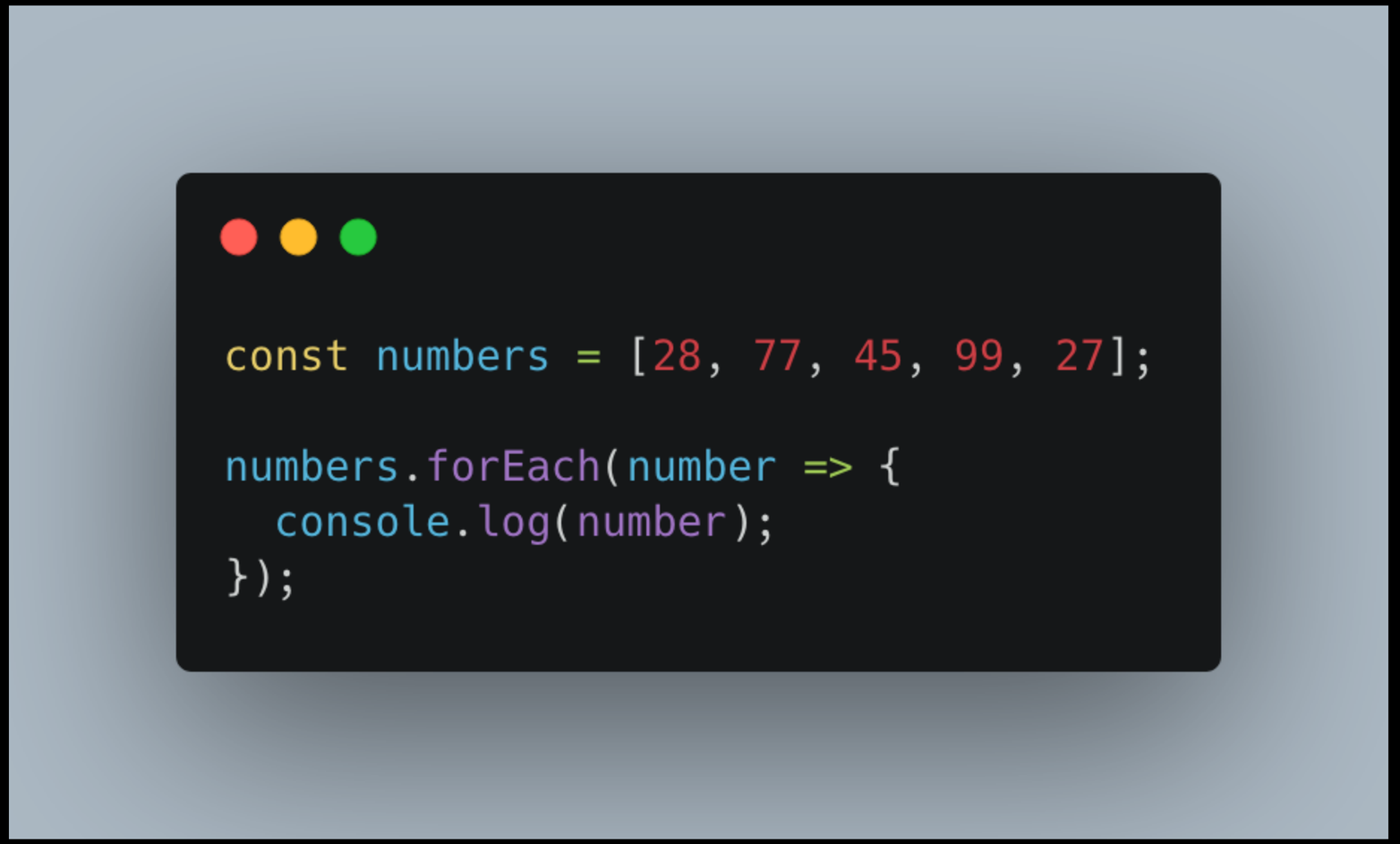
[…] They are usually used for **looping and iterating over datasets** and are useful for transforming that data along the way.

Higher-order

Definition

[…] They are usually used for **looping and iterating over datasets** and are useful for transforming that data along the way.

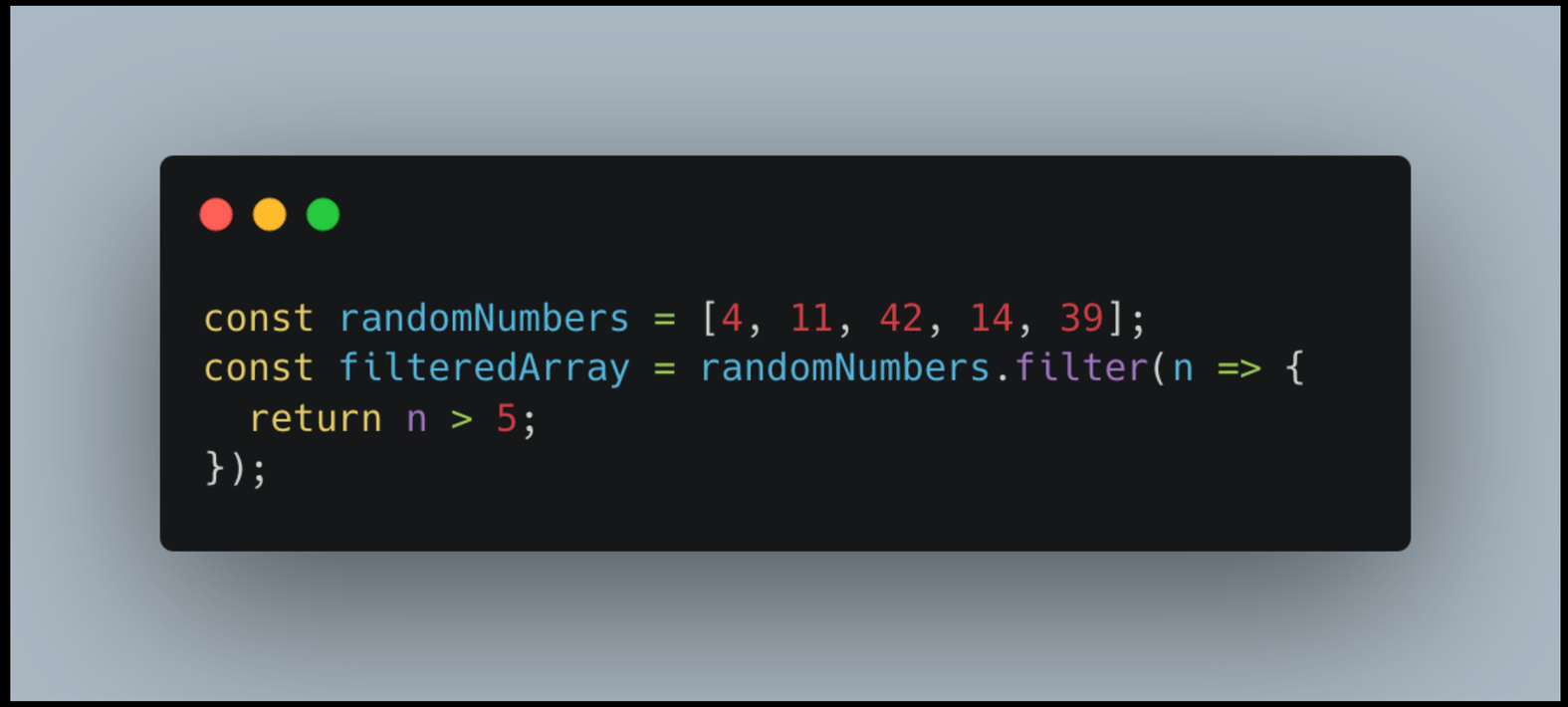
Higher-order .foreach (iterating)



```
const numbers = [28, 77, 45, 99, 27];

numbers.forEach(number => {
  console.log(number);
});
```

Higher-order .filter



```
const randomNumbers = [4, 11, 42, 14, 39];
const filteredArray = randomNumbers.filter(n => {
  return n > 5;
});
```

Higher-order .map

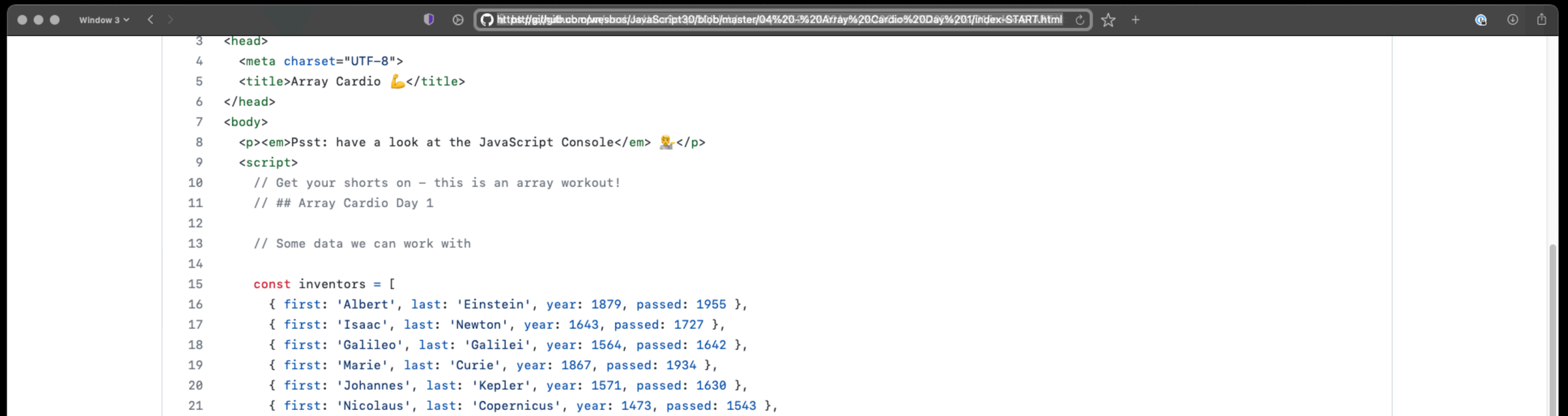
```
const finalParticipants = ['Taylor', 'Donald', 'Don', 'Natasha', 'Bobby'];

// add string after each final participant
const announcements = finalParticipants.map(member => {
  return member + ' joined the contest.';
})

console.log(announcements);
```

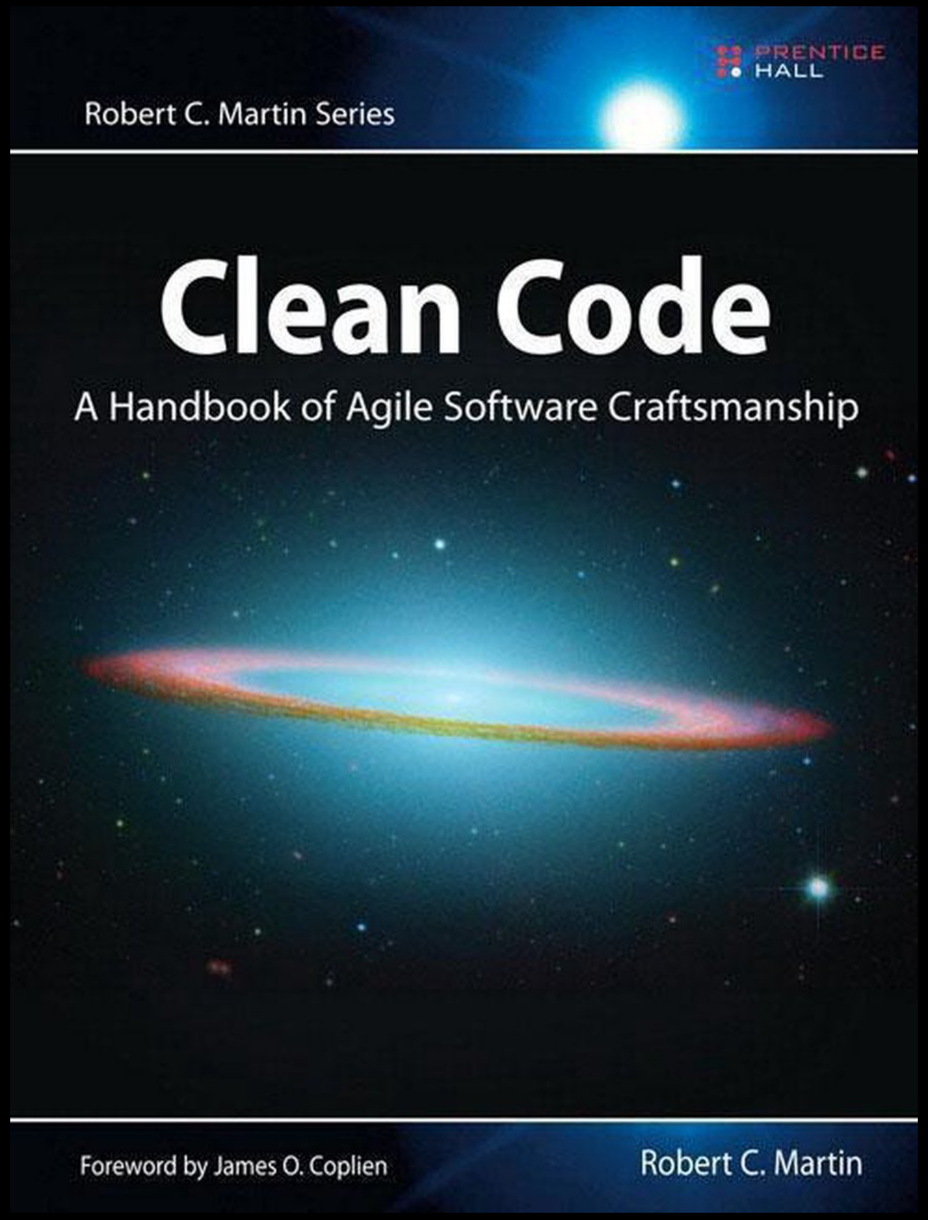
Assignment (30m)

Try to solve the Array Cardio higher order function challenges from Wes Bos JavaScript30 course.



Break!

Coding Principles


PRENTICE HALL
Robert C. Martin Series
Clean Code
A Handbook of Agile Software Craftsmanship
Foreword by James O. Coplien
Robert C. Martin

Coding principles

What ?

Coding principles are a set of **guidelines** we use while writing code. It allows us to write cleaner, better, easier maintainable and transferrable code.

Coding principles

Variables

- Use meaningful and pronounceable variable names
- Use the same vocabulary for the same type of variable
- Use searchable names
- Use explanatory variables
- Avoid Mental Mapping
- Don't add unneeded context

Coding principles

Functions

- Function arguments (2 or fewer ideally)
- Functions should do one thing
- Function names should say what they do
- Functions should only be one level of abstraction
- Remove duplicate code
- Don't over-optimize

General principles

KISS/DRY

Can I going ice-skating?

Water can have many forms, gas as in steam, liquid as we usually know and frozen, also known as ice. During different temperatures, water behaves differently. When water reaches a temperature of 0 or lower, it starts to freeze. When this proces has been going on for a while, the layer of water frozen may be thick enough to support the weight of an adult. During last night, there was an approximate temperature of -2 degrees for a continues period of 2 hours. The given amount of ice that aggregated is insufficient to support an adult male for ice skating

General principles

KISS/DRY

Can I going ice-skating?

No.

General principles

KISS/DRY

Warning! Falling rocks ahead! Warning! Falling rocks ahead! Warning! Falling rocks ahead!
Warning! Falling rocks ahead! Warning! Falling rocks ahead! Warning! Falling rocks ahead!
Warning! Falling rocks ahead! Warning! Falling rocks ahead! Warning! Falling rocks ahead!
Warning! Falling rocks ahead! Warning! Falling rocks ahead! Warning! Falling rocks ahead!
Warning! Falling rocks ahead! Warning! Falling rocks ahead! Warning! Falling rocks ahead!
Warning! Falling rocks ahead! Warning! Falling rocks ahead! Warning! Falling rocks ahead!
Warning! Falling rocks ahead! Warning! Falling rocks ahead! Warning! Falling rocks ahead!
Warning! Falling rocks ahead! Warning! Falling rocks ahead! Warning! Falling rocks ahead!
Warning! Falling rocks ahead! Warning! Falling rocks ahead! Warning! Falling rocks ahead!
Warning! Falling rocks ahead! Warning! Falling rocks ahead! Warning! Falling rocks ahead!
Warning! Falling rocks ahead! Warning! Falling rocks ahead! Warning! Falling rocks ahead!

General principles

KISS/DRY

WARNING:

FALLING ROCKS AHEAD

General principles

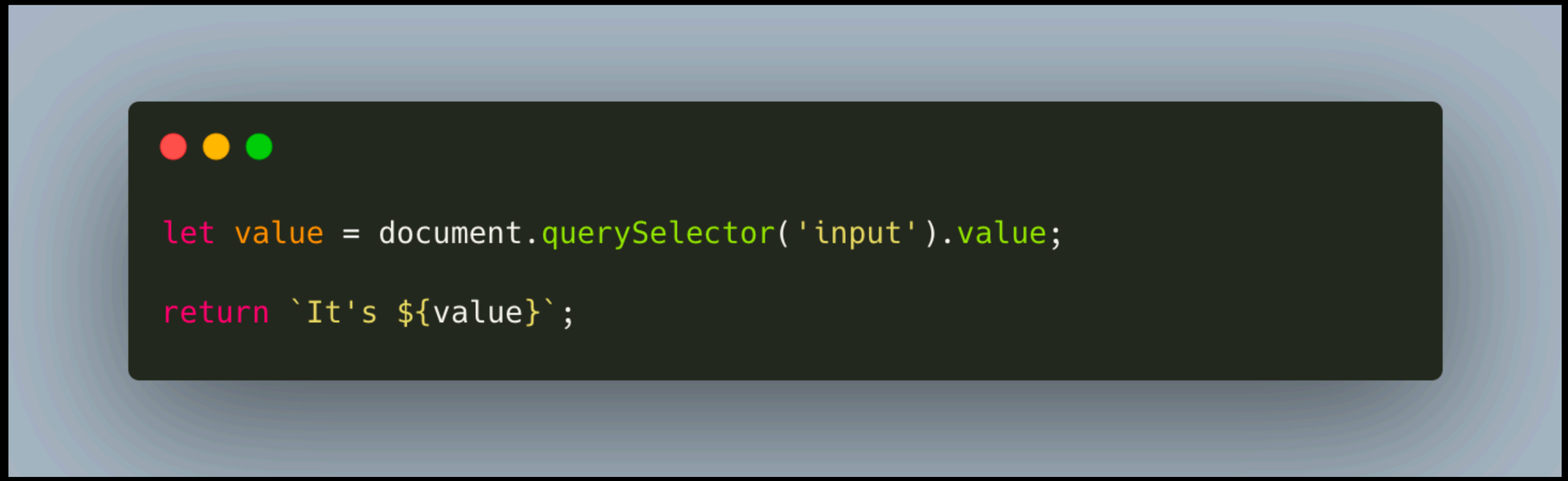
KISS/DRY

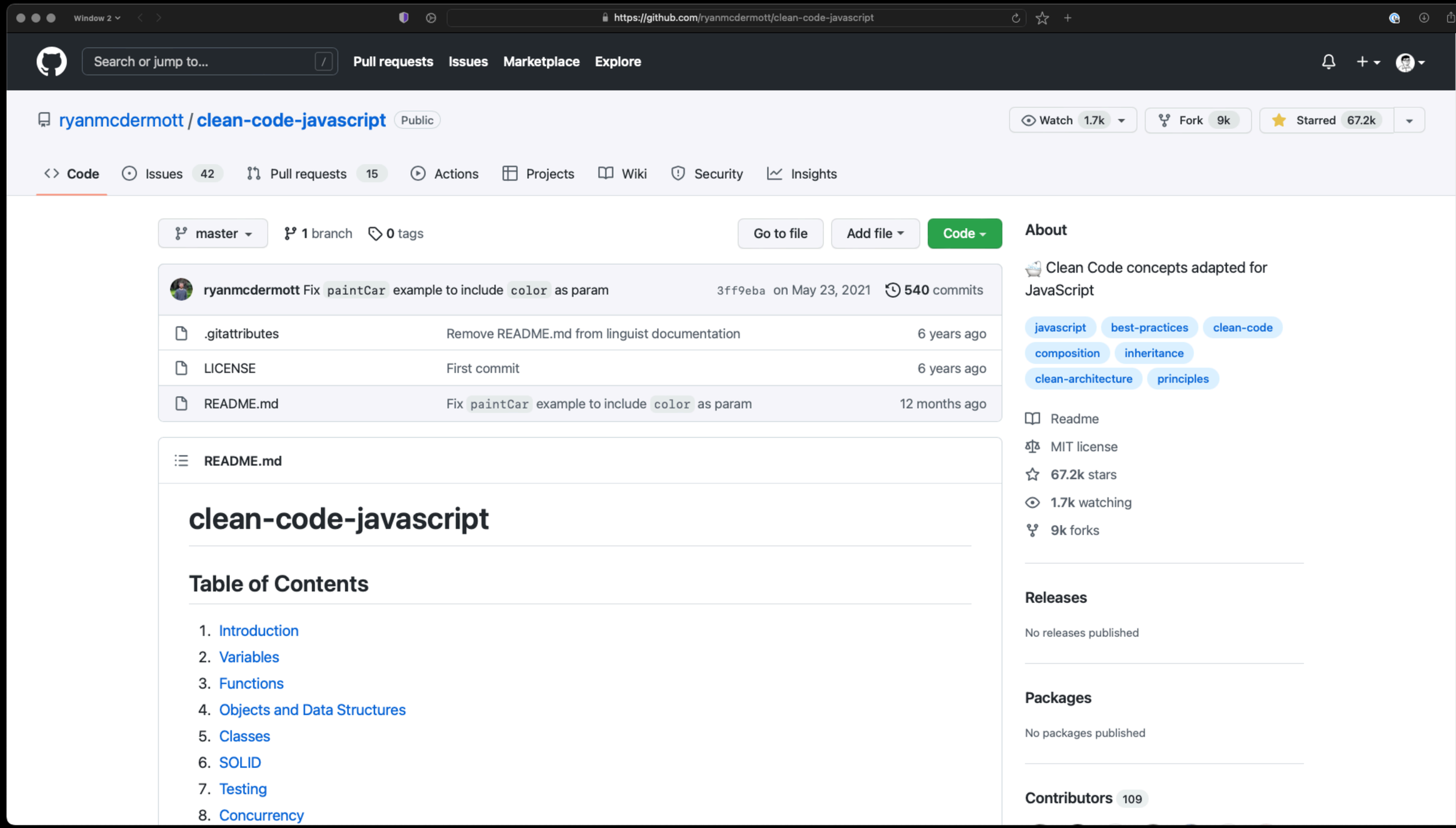
```
let value = document.querySelector('input').value;

if(value == 1) {
  return "it's 1";
} else if(value == 2) {
  return "it's 2"
} else if(value == 3) {
  return "it's 3"
} else if(value == 4) {
  return "it's 4"
} else if(value == 5) {
  return "it's 5"
} else if(value == 6) {
  return "it's 6"
}
```

General principles

KISS/DRY




Window 2
https://github.com/ryanmcdermott/clean-code-javascript
Search or jump to...
Pull requests
Issues
Marketplace
Explore
ryanmcdermott / clean-code-javascript
Public
Watch 1.7k
Fork 9k
Starred 67.2k
Code
Issues 42
Pull requests 15
Actions
Projects
Wiki
Security
Insights
master
1 branch
0 tags
Go to file
Add file
Code
ryanmcdermott Fix paintCar example to include color as param
3ff9eba on May 23, 2021
540 commits
.gitattributes
Remove README.md from linguist documentation
6 years ago
LICENSE
First commit
6 years ago
README.md
Fix paintCar example to include color as param
12 months ago
README.md
clean-code-javascript
Table of Contents
1. Introduction
2. Variables
3. Functions
4. Objects and Data Structures
5. Classes
6. SOLID
7. Testing
8. Concurrency
About
Clean Code concepts adapted for JavaScript
javascript
best-practices
clean-code
composition
inheritance
clean-architecture
principles
Readme
MIT license
67.2k stars
1.7k watching
9k forks
Releases
No releases published
Packages
No packages published
Contributors 109

Assignment (30m)

Have a look at the Clean Code in JavaScript by Ryan McDermott repo. Get a piece of code from your project (either back-end or front-end) **and write down specific rules that you can refactor on your project and then apply them!**

A1

Assessment

description

For your A1 assessment, you're going to implement a *progressively enhanced component.* In short, you're going to **enhance the client-side experience of the use**r by doing research, documenting patterns and implement the principle of *progressive enhancement* using JavaScript.

/readme.md

course

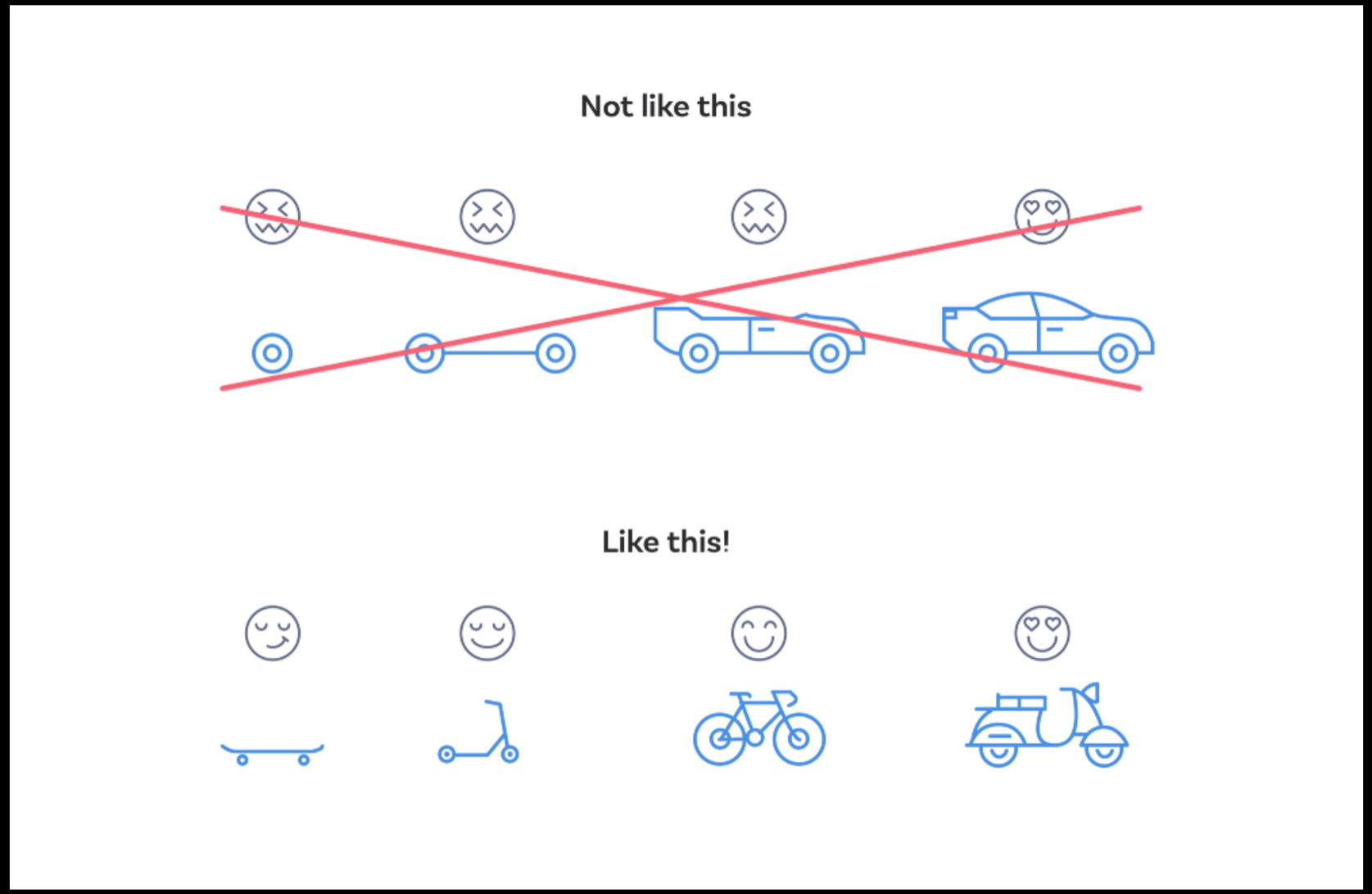
goals

- You improve your knowledge about core JavaScript concepts
- You are able to build progressively enhanced frontend components
- You can build a web application with semantic HTML/CSS/JS
- You can write docs and explain your code and application structure
- You are able to research sources and read documentation

A1

deliverables

- **Progressive enhancement (code):** working **interactive enhancement** for matching application
- **Process book (wiki):** that provides insight into the weekly iterative process and documents your research



Andy Bell - The power of enhancement

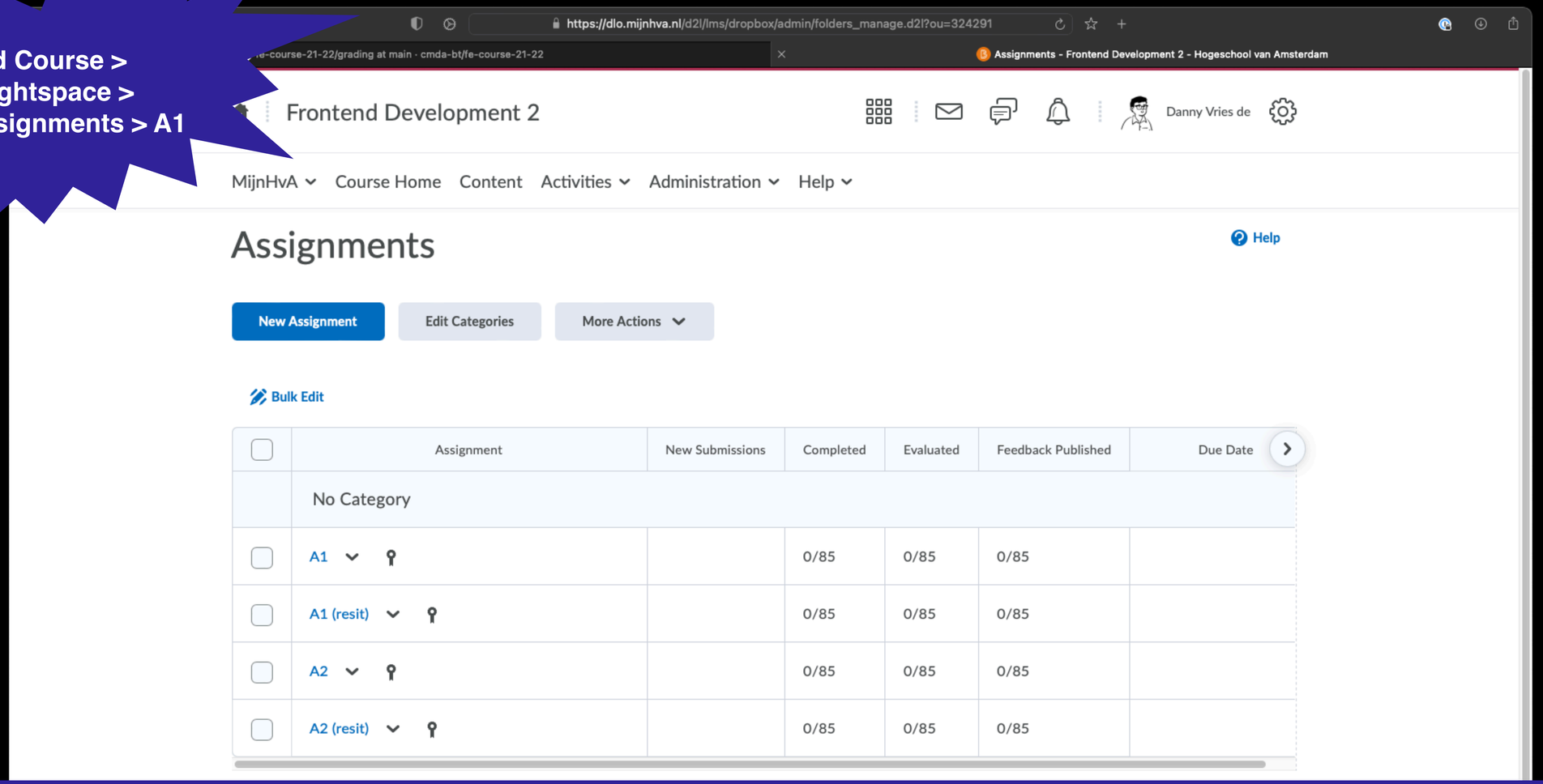
fe() - a1	1-2	3-4	5-6	7-8	9-10	feedback
Progressive Enhancement (PE)	There is no form of PE	There is some PE in your functionality, but it is not working / not significant	Your feature uses JavaScript to enhance the experience of the user	You've used one or more Web API's and/or external sources to provide a layered cake of PE	You've gone above and beyond to give the user the best possible experience when possible	
Application	The feature doesn't work; there are errors and warnings	The feature partially works but is not complete; incomplete flow	The feature completely works and is usable from a user experience point of view	The feature is advanced and is technically more complex	The user experience is fantastic and the feature is complex. You took special care of your interface and your user	
Quality	The project is handed in documented, on time, working without technical problems, and on GitHub	The code is readable, consistent and the code, project, and process are partially documented	Code adheres to standards; docs cover the process and what the project is and does	Code quality is good and enforced; docs are more than useful and professional	Code and docs both read like great books and the project is structured logically	
Front-end	Your HTML, CSS and JavaScript contain errors and is messy (divs!???)	Your HTML, CSS and JavaScript display some form of semantic meaning, but are still all over the place	Your HTML is semantic and appropriate elements have been chosen. Your CSS is using some sort of structure and uses CSS variables. Your JavaScript is mostly consistent and structured	Your HTML, CSS and JavaScript display high levels of skill and learning. Other developers would love to build on your work	HTML, CSS and JavaScript are exemplary. You may have used preprocessors properly, worked modular and/or provide in-code documentation like a professional developer	

You'll need a > 5.5 for each row to pass: you can't compensate between rows.
Each of this rubric's rows is cumulative: for example, to get a 5-6 on application, you also need to have a 1-2 and 3-4.

student name	student number	lecturer	date (first chance)	grade		

/grading

Fed Course >
Brightspace >
Assignments > A1



Deadline: xxxxx

Q&A

exit;

see you in a1!