Clean Code & Agile Experiences

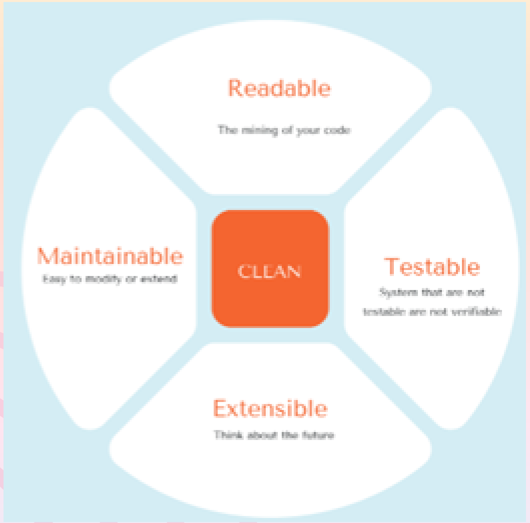
Team 6:

Adrian Bernardino

Jan William Haug

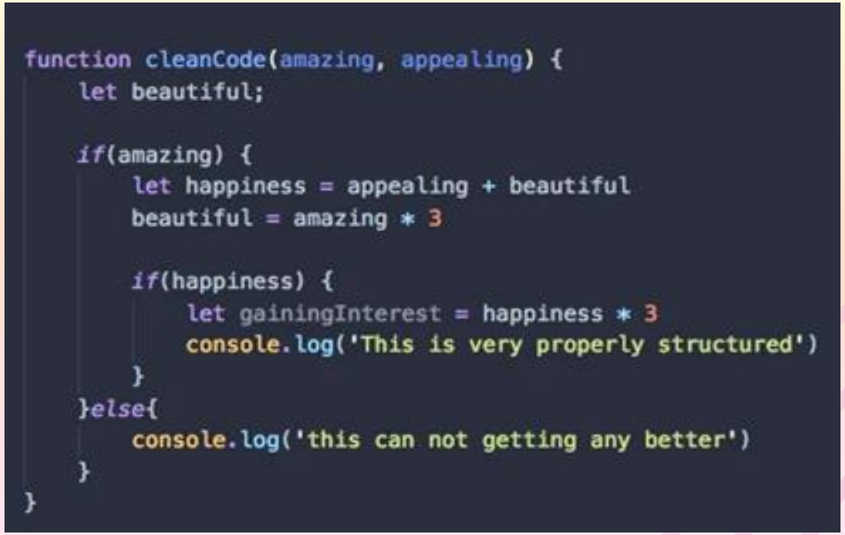
Stephania Rey

Table of contents



✳ 01

Introduction to clean code



```
function cleanCode(amazing, appealing) {
    let beautiful;

    if(amazing) {
        let happiness = appealing + beautiful
        beautiful = amazing * 3

        if(happiness) {
            let gainingInterest = happiness * 3
            console.log('This is very properly structured')
        }
    }else{
        console.log('this can not getting any better')
    }
}
```

What is Clean Code?

- Clean code is an approach to software development that is easy to read and understand.

- Clean code requires a lot of discipline

- Benefits for easier maintenance

- Code is cost effective
  - Clean code leads to less technical debt

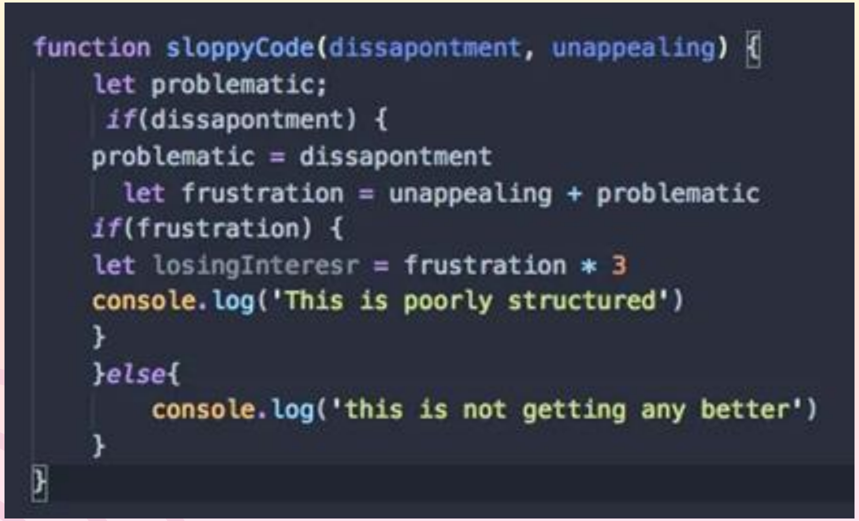
Clean Code

Clean code is focused →

- Each function, class, and module exposes a single minded attitude:
  1. Undistracted
  2. Unpolluted

- Should possess a range of desirable characteristics such as:
  1. Minimal Software Dependencies
  2. Robust Testability

What is Clean Code? With “Uncle Bob”



```
function sloppyCode(dissapontment, unappealing) {
    let problematic;
     if(dissapontment) {
    problematic = dissapontment
      let frustration = unappealing + problematic
    if(frustration) {
    let losingInteresr = frustration * 3
    console.log('This is poorly structured')
    }
    }else{
        console.log('this is not getting any better')
    }
}
```

02

Advantages and Disadvantages of clean and bad code

Advantages vs Disadvantages in coding

- Advantages:

-Easier to test
-Lower maintenance burden

- Disadvantages:

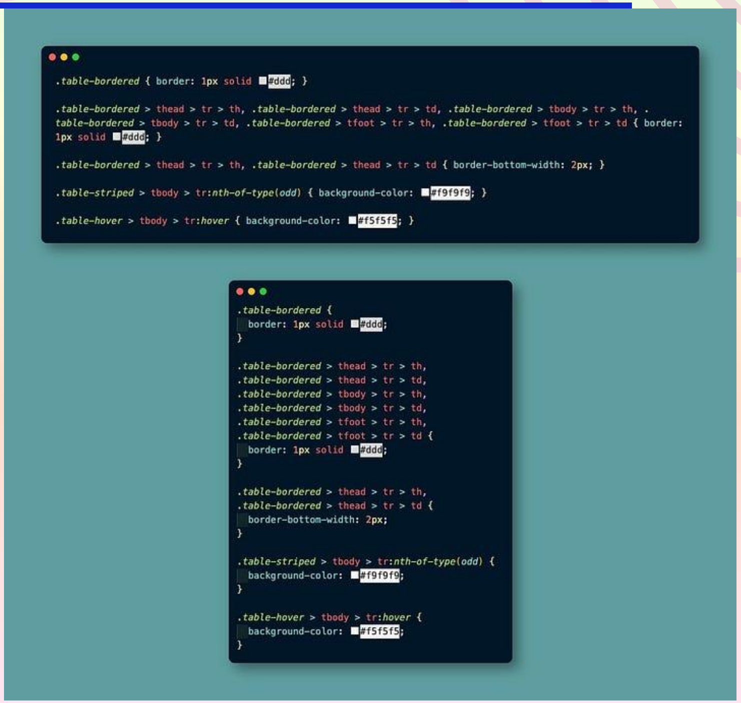
-reduces software quality
-increase in development time
-security vulnerabilities

03

Code Layout

Code Layout

- Pleasing appearance that is readable
- Efficient code
- Error handling should be complete and attention to detail
- Abbreviated error handling
- Memory leaks
- race conditions
- Inconsistent naming



```
.table-bordered { border: 1px solid #ddd; }

.table-bordered > thead > tr > th, .table-bordered > thead > tr > td, .table-bordered > tbody > tr > th, .
table-bordered > tbody > tr > td, .table-bordered > tfoot > tr > th, .table-bordered > tfoot > tr > td { border:
1px solid #ddd; }

.table-bordered > thead > tr > th, .table-bordered > thead > tr > td { border-bottom-width: 2px; }

.table-striped > tbody > tr:nth-of-type(odd) { background-color: #f9f9f9; }

.table-hover > tbody > tr:hover { background-color: #f5f5f5; }
```

```
.table-bordered {
  border: 1px solid #ddd;
}

.table-bordered > thead > tr > th,
.table-bordered > thead > tr > td,
.table-bordered > tbody > tr > th,
.table-bordered > tbody > tr > td,
.table-bordered > tfoot > tr > th,
.table-bordered > tfoot > tr > td {
  border: 1px solid #ddd;
}

.table-bordered > thead > tr > th,
.table-bordered > thead > tr > td {
  border-bottom-width: 2px;
}

.table-striped > tbody > tr:nth-of-type(odd) {
  background-color: #f9f9f9;
}

.table-hover > tbody > tr:hover {
  background-color: #f5f5f5;
}
```

04

Groups Agile experiences

Project management

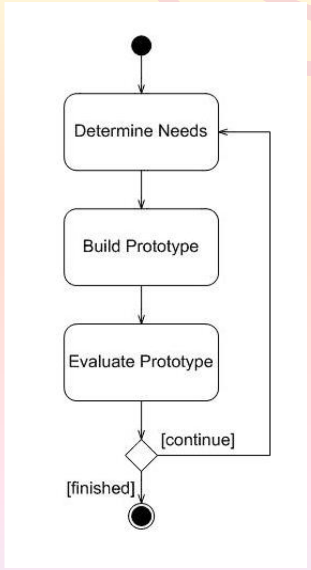
- **Project planning:** Coordinating times and dates,meetings schedule, and the team's tasks.
- **Task management:** Assigning specific tasks for each Sprint and complete them
- **Risk management:** Reviewing project initiatives & requirements to ensure productivity stays on track and minimize potential risks
- **Performance management:** Monitoring and measuring the performance of the project team. Identifying areas for improvement towards project goals.

User Interface prototyping

- **We started by creating a user interface prototype**

- UI prototypes can have several purposes:
- Enabled us to explore the problem with our stakeholders.
- To initially envision the system.
- Enabled us to explore the solution of our system.
- A way to communicate the possible UI design(s) of our system.
- A potential foundation for our system

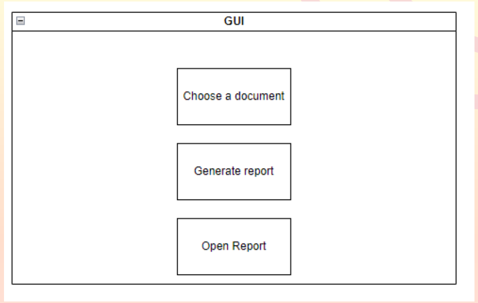
UI prototyping process



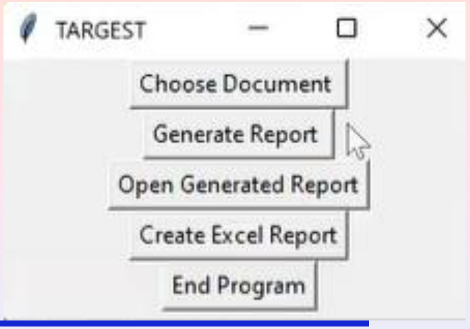
User Interface (continued)

- We prototyped a portion of the user interface
- Then moved on to implementing it.
- We don't need to define everything up front before moving on
- After a version of the UI prototype is built, it needed to be evaluated by our mentor to verify that it meets the requirements.

Prototype



Our Gui last semester

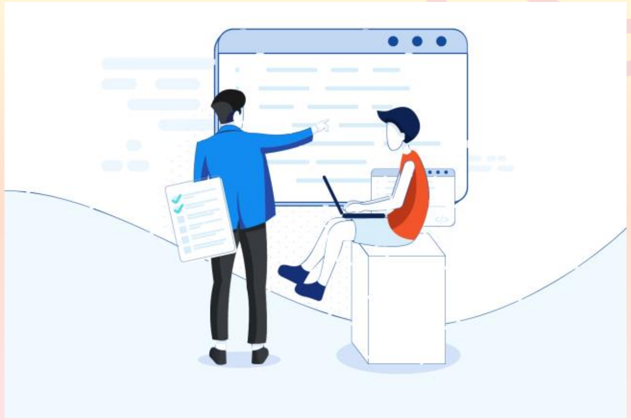


Code and design review

Code Review: Our team used code review tools like GitHub to hold a central repository to review each commit being made and highlight our continuous improvement to our code base.

Design Review: As a team, we constantly updated our design architecture to meet the standards of our code.

Collaborative Review: We always discussed about alternative design patterns and potential issues we might face if we weren't updating the architecture.

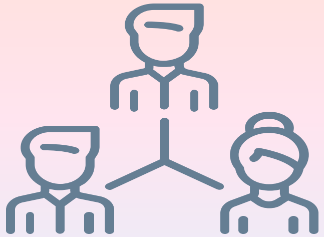


Small release cycles

- Delivering working software in small increments allowed for us to have consistent feedback
- Frequent feedback helped us identify issues earlier in the development process.
- Small release cycles are a great way to build momentum over time.

Scrum meetings

- All team members attend
- Sprint planning meeting
- Daily scrum
- Sprint review meeting
- Sprint retrospective meeting
- Point out issues



Kahoot!!

when you win at kahoot in class, but fail all your exams
It ain't much, but it's honest work

Me and the boys waiting for our funny kahoot names to show up

https://create.kahoot.it/share/cs-351-quiz/cfc89ffa-f9e3-406f-b7fb-38ab16e1fb27

Thanks!

Do you have any questions?