1. API Design Specification (Scenarioer og Klient-kode)
2. The resulting API from the design spec
3. Principles of the design (Liten rework)
4. Result of User Testing: Description of setup, the code, and the feedback (Check)
5. The Revised API (Kun endringene I koden)
6. Examples from the API (Nye/Finere eksempler )

# API Design Specification

## Scenarios

test

## Client-Code

test

# Principles of the Design

(Nevn om hvordan høynivå api skal fungere for HTML og CSS, få frem forskjellen mellom høynivå/lavnivå?)

The framework consists of two major APIs one for HTML and one for CSS. In the low level HTML API each major HTML tag is represented as classes,

The goal of this web framework is to let users easily create general HTML structures and HTML pages, with emphasis on simplicity. The framework enables people lacking knowledge of HTML and CSS to generate webpages from a range of functions. The framework is easy to understand and effective for both new and recurring web designers, which allows them to make simple websites quickly, but also modify specific details such as class names and meta-data.

The idea is that the user makes html-tags as objects in Java, and appends them orderly to their page objects, which compiles into HTML-files. This lets them engage in the same design principals as before, without worrying about HTML syntax. The framework simplifies the process by providing pre-built html templates and overloaded methods, meaning significantly less code and a more intuitive API. Yet, the user can go deeper and modify details such as the ID of a tag. For example, generating an HTML form is very quick and easy, and changing the label is only a matter of adding a new property to the form. To save the newly added changes, the framework translates the page object into HTML-code and writes it as a local file on the machine.

# User Testing

We were given the opportunity to cooperate with Group 5 when doing user testing, where they tested our framework, and later we tested theirs. The following section covers the results from our session.

## Description of the setup

We sent a jar file containing the framework to Group 5 and started with some simple instructions. Firstly, they needed to start a new project and add our framework/library to their project from the project structure menu. Once this was done, they were ready to start coding inside the main method.

We provided them with two sample files, one called index.html and one called style.css, their task was to simply reproduce the structure and the code inside these files using our Java framework.

|  |  |
| --- | --- |
| index.html | style.css |
|  |  |
|  |  |

At first, we let them figure out the process themselves, but once they met issues and got stuck, we gave explanations to what each code block represented and guided them through the process. The next section will cover the resulting code and feedback from our testing.

## The Code

The following is the client code created by Group 5 using our framework to make a simple HTML and CSS file:

import CSS.LowLevel.Style.RuleSet;

import CSS.LowLevel.StyleSheet;

import HTML.LowLevel.Component.HTML;

import HTML.LowLevel.Element.Anchor;

import HTML.LowLevel.Element.Container;

import HTML.LowLevel.Element.Heading;

import HTML.LowLevel.Element.Paragraph;

public class Main {

    public static *void* main(String[] *args*) {

    // write your code here

        HTML html = new HTML();

        html.setLinkType("stylesheet","style.css" ,"css" );

        Container head = new Container("head");

        Container body = new Container("body");

        html.append(head);

        Container header = new Container("header");

        Container main = new Container("main");

        Container footer = new Container("footer");

        Heading heading = new Heading("1", "asdasdsada");

        heading.applyClass("h1 class name");

        header.addElement(heading);

        body.addElement(header);

        body.addElement(main);

        body.addElement(footer);

        Container nav = new Container("nav");

        header.addElement(nav);

        Anchor anchor = new Anchor("google","google.com");

        nav.addElement(anchor);

        Container article = new Container("article");

        Paragraph p = new Paragraph("YESYEYES");

        article.addElement(p);

        p.applyId("paragraph");

        main.addElement(article);

        html.append(body);

        html.initialize();

        StyleSheet css = new StyleSheet();

        RuleSet rs = new RuleSet();

        rs.setSelector("body");

        rs.addRule("background-color", "red");

        rs.setDisplay("grid");

        RuleSet rsNav = new RuleSet();

        rsNav.setSelector("nav");

        rsNav.setDisplay("flex");

        RuleSet rsfont = new RuleSet();

        rsfont.setClass("header");

        rsfont.addRule("font-size", "12px");

        RuleSet pC = new RuleSet();

        pC.setId("paragraph");

        pC.addRule("background-color", "blue");

    }

}

(Note that this code section excludes all errors)

## Feedback

Group 5:

Right after they had tested our framework and completed the setup, we asked them to give their initial reaction and opinion. They said, “It was pretty good once we got the hang of the class names” which was a relief to hear. It seemed to us that they found the APIs to be intuitive, as they quickly grew familiar with the main classes. We then followed up with a few questions regarding their experience, and they told us they could easily expand further on their website with enough practice.

Group 5 later sent us a short report on our program, starting with the setup: “The setup was pretty good, since we were ordered to reproduce a given webpage along with its styling, which is pretty straightforward. The setup structure was good”. And to the actual framework they said: “The code was semantic and well structured. It is also easily readable and applicable to making websites with java.”. They thought our framework took a good direction for forming webpages, as they found Java to be challenging for that use.

“The only nit-pick is that some parts of the framework produced errors, or they wouldn’t work like they did in the example code.”, meaning that some classes didn’t extend correctly, and didn’t have access to certain methods. “For example, when we tried to add a paragraph to an article element, java didn’t recognize it. Other than that, the code structure was clean”.

### Notes taken during the testing

During our session with Group 5, we took some keynotes while they were playing with the framework. Here’s a list of the main issues we discovered:

* setLinkType method must be accessible within the Container class.
* setLinkType method should have an overload with two parameters.
* The Heading class should take an integer as the first parameter (for level).
* The Heading class must extend correctly, as it didn’t have access to the applyClass method.
* addElements method could have an overload that uses varargs in parameter.
* The Paragraph class must extend correctly, as it didn’t have access to addElement and applyId/applyClass.

# The Revised API

test

# Examples from the API

test