

# Program & Career Orientation

---

**CU75001V3 - PCO**

July 2025

HZ University of Applied Sciences



# PCO - Your Showcase Website

---

## Introduction

## Course content

Preparation Assignment

Set up your working environment

Good sites

Code Review

Visual Design

Motivation & Presentation

Online Company Safari

Poster presentations

Assessments

## Assignment

Assignment Specification

## Resources and Further reading

String Case Styles: Camel, Pascal, Snake, and Kebab Case

8 of the Most Important HTML Tags for SEO

What Is Code Review?

# Introduction

---

In the Program and Career Orientation (PCO) course you will get to know each other, the teachers, the program and the career opportunities. Using HTML and CSS you will take your first steps as software engineer and build your own website. The course follows a two week pattern with lectures, workshops and self study. Based on the knowledge you gain in the lectures and workshops you execute the assignment, building your website step by step. The subject of the website is you and your IT education supported by examples and/or reflections. You answer questions like 'who am I?', 'why did I choose IT?' and 'what does the IT program at the HZ look like?'. The course will have a final assessment with a minimum grade to pass of 5.5.

This course reader functions as a workbook for the course. It contains both knowledge and exercises. The Learn page for the PCO page guides you through the timing of the lectures and which sections of this course reader are discussed on which days. In the schedule you also see selfstudy time, indicated as 'guided self study'. During guided self study you work on the exercises on your own or with fellow students. On your schedule, you can find a room with one or more teachers, you can go to this room for additional help and guidance from teachers and teacher assistants. They can help you on things like:

- The code is not behaving as expected
- Your working environment does some strange things
- You can't get your code pushed to GitHub
- You don't understand something from the workshops, reading material or other
- You don't know what to do
- You want to try something that we didn't discussed (yet), and want some leads to start
- You just want some advice about your code or working environment
- You want some inspiration on what to add to your project

# Preparation Assignment

---

The first two weeks of the ICT program consists of an assignment in which you "apply" for the ICT study programme. Convince the teachers of your ICT talent. In order to do so, you create a *showcase* website that displays what you are able to achieve in this period. It will also explain how you study and apply the necessary knowledge in order to achieve your goals. At the end of the two weeks, you will present the result to a panel of teachers and students.

To prepare for this assignment, you need to get (re-)acquainted with the HTML language and collect some data from the SKC environment. The instructions below will help you to do this properly.

**!** IMPORTANT Before the course starts, you have prepare yourself by following the instructions in the chapters below. It should take you 6 to 7 hours if you are entirely new to coding.

## Retrieve your WHO AM I document

In the past period, you entered a digital environment called **SKC HBO-ICT**. In this environment you had the opportunity to let us know who you are in the **WHO AM I** assignment. You filled out a questionnaire and you can use this as a base for the content of some of your web pages.

Go back to the **submission status page** on that environment and:

1. Retrieve your original answers.
2. Retrieve the feedback you got from the teacher(s) about the assignment

💡 If the assignment hasn't been graded, you wouldn't have received any feedback. If this is the case, don't worry. Move on to the next step and ask your teachers next class.

## Prepare your IDE

HTML documents are plain text based files. These files can easily been created and edited using any text editor that can edit plain text. The following site lists some notable editors:

[List of text editors](#)

Using a normal text editor can be annoying, so we prefer to use an Integrated Development Environment (IDE)

We prefer Microsoft **Visual Studio Code** (or VSCode for short) as the editor for different types of plain text files. During the program, you will see us using this editor. We advise you to use this tool as well, but you are free to use any other editor you find suitable.

Download and install VS Code from the link below:

## Study HTML

✓ If you did all the SKC assignments, including the HTML, you can skip this section.

In order to create websites you have to use HTML. During your pre-education and/or just out of interest, you might have already have studied the HTML language. Refresh your knowledge about this language if needed.

When you did not study any HTML yet, you can use the online courses of FreeCodeCamp now.

[freeCodeCamp.org](https://freeCodeCamp.org)

Before you start, make an account and log on. All courses can be found under the heading "/learn" on top.

The course you need is: **(New) Responsive Web Design -> Learn HTML by building a Cat Photo App:**

[freeCodeCamp.org](https://freeCodeCamp.org)

○ Learn HTML by Building a Cat Photo App

HTML tags give a webpage its structure. You can use HTML tags to add photos, buttons, and other elements to your webpage.  
In this course, you'll learn the most common HTML tags by building your own cat photo app.  

Start project

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68		

⚠ This assignment (all 68 steps) should take you 3-4 hours.

## Create your first HTML page

Use the editor to create a new file called `profile.html`. Note the extension `.html`. This is required for all web pages. Build a basic web page HTML structure using the `<html>`, `<head>` and `<body>` tags like:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```


```
<meta http-equiv="X-UA-Compatible" content="ie=edge">
<title>your-title-here</title>
</head>
<body>
  <h1>Hello World!</h1>
</body>
</html>
```

Feel free to copy and paste the code snippet in your file. If you do, replace the content of the `<title>` tag from `your-title-here` into `My profile`.

Next, use at least `<h1>` and `<p>` tags to copy and paste the following sections from your WHO AM I assignment into the `<body>` of the document:

- Personal information
- Extra-curricular activities

Use the feedback you retrieved to check whether some things might require some changes.

 You can already show off your HTML skills by using other tags like `<ul>` or `<img>` to add more content or make the content more structured.

Test the document by opening the file in your browser.

## Install and learn Git

We are going to use an app called Git a lot in the future. So, install it now. We recommend installing Git on your computer by downloading the command line version of git.

### Git - Downloads

Git is a piece of software that has been specifically created to exchange code (like HTML pages) between software development teams. It also keeps track of the history of coding. You can compare it to apps like Dropbox, OneDrive, iCloud or Google Drive. There is a big difference however: you need to tell Git you want to save something everytime. Dropbox, Onedrive and other services usually automatically upload your changes every couple of minutes. This makes Git somewhat unintuitive to learn, but will save you many, many, many times later on. You will probably experience this in the second quartile of this year. For now, you'll have to accept that Git is something you just need to get used to.

Watch the explanation in the video below and type along to practice with Git. Please take note of the following points while watching the video. **Read these first before you watch the video**

1. You can skip the following chapters:
  - SSH Keys
  - Git branching
  - Undoing in git
  - Forking in git
2. When creating a GitHub account, please use your HZ email address. When signed up, request student benefits from [https://education.github.com/discount\\_requests/student\\_application](https://education.github.com/discount_requests/student_application). It can take a couple of weeks to clear your request and you need the benefits in Quarter 3, so please request them now.
3. Don't use *SSH* if you don't know what you're doing. Use the *HTTPS* option if you want to clone a repository. This usually is the default in GitHub. At some point your terminal will direct you to the

GitHub website and ask for permission. You can safely accept.

<https://www.youtube.com/watch?v=RGOj5yH7evk>

## Prepare for class

Bring the `profile.html` document, your WHO AM I assignment and feedback to class.

# Set up your working environment

A **Software Development Working Environment** is typically a computer where software is installed that supports software development. From code editing tools, usually referred to as *Integrated Development Environments* or *IDEs* to tools to test the code like web- and database servers. A good working environment setup is such that it helps the developer create, test and share code as efficient as possible.

📌 Some important links:

- [Preparation assignment](#)
- [Assignment specification](#)

## Outcomes

When you are done with this assignment, your final result should look like this:

1. When you enter the URL: `http://127.0.0.1:5500/profile.html` in your browser, you should see your profile page.
2. When you make a change in the file with your editor (IDE), you should see that change in the browser after you refresh (F5)
3. When you enter the URL: `https://yourusername.github.io/profile.html` where *yourusername* is your GitHub username, you should see the same profile page.
4. When you make a change to the file with your editor, *commit* the change and *push* it to your remote repository on GitHub, you should see that change in the browser after you refresh (F5).

## Preparation assignment

Before the course started, you were asked to prepare yourself. With this preparation, you:

- Have a basic knowledge on *HTML*.
- Created your first HTML page: `profile.html`. The content is based on the information from the **Who am I** assignment from the SKC page
- Created an account on **GitHub** using your HZ email address, Installed **Git** on your computer and have a basic understanding of what Git is

## Lecture

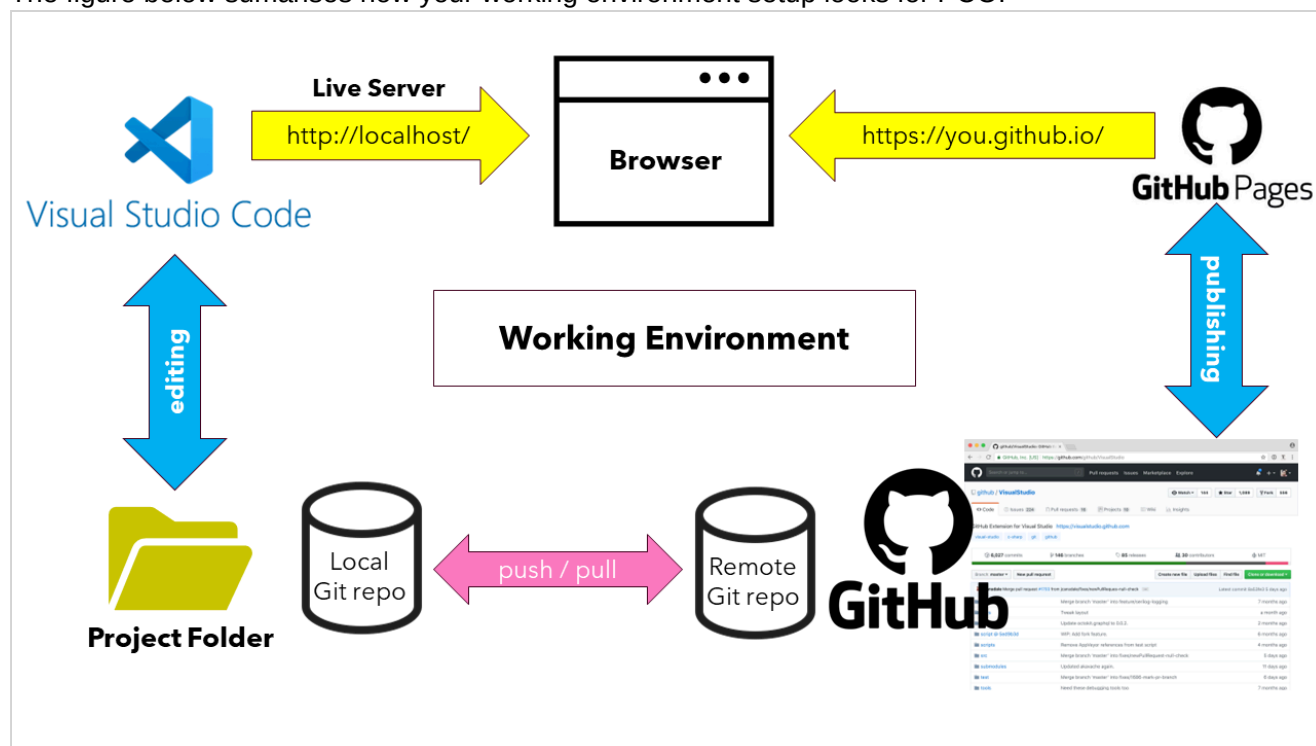
You attended the workshop on your working environment. Not only for this assignment, but partly also for your entire school career. You:

- Used your account to connect your laptop to the **Eduroam wifi network**
- Installed your first IDE ([Visual Studio Code](#))
- Hosted your website locally using the **Live Server** extension on VS Code.

! If you did not attend the workshop and/or preparation you need to do these as soon as possible. You need this for the rest of the assignment, and later.



The figure below summarises how your working environment setup looks for PCO:



## Git and GitHub

🚧 Have you never used to command line before? Have a look at this video first:

### Command Prompt Basics: How to use CMD

⚠️ The following exercises are based on the fact that you already installed Git and have a basic knowledge about Git and GitHub. If not, go back to the Preparation assignment and work through the section on that subject.

👉 On Windows, if you get an error that 'git' is not recognized as an internal or external command,

type the following command

`winget install --id Git.Git` to install the command line version of git.

## Create your Repository on GitHub

The following exercises will guide you through the process that sets up Git version control to your project and connects it to your GitHub Pages Repository.

✂️ Exercise 1.1: Browse to GitHub and log in with your account. Create a new repository.

Top repositories



Name the new public repository `yourusername.github.io` where `yourusername` is your username on GitHub.

Tick the `Add a README file` checkbox.

## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

*Required fields are marked with an asterisk (\*).*

### Repository template

No template ▾

Start your repository with a template repository's contents.

---

Owner \*

Repository name \*

YourUsername ▾

 / 

yourusername.github.io

🔒 yourusername.github.io is available.

Description (optional)

---

☒ Public

Anyone on the internet can see this repository. You choose who can commit.

☐ Private

You choose who can see and commit to this repository.

---

Initialize this repository with:

☒ Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: None ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: None ▾

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

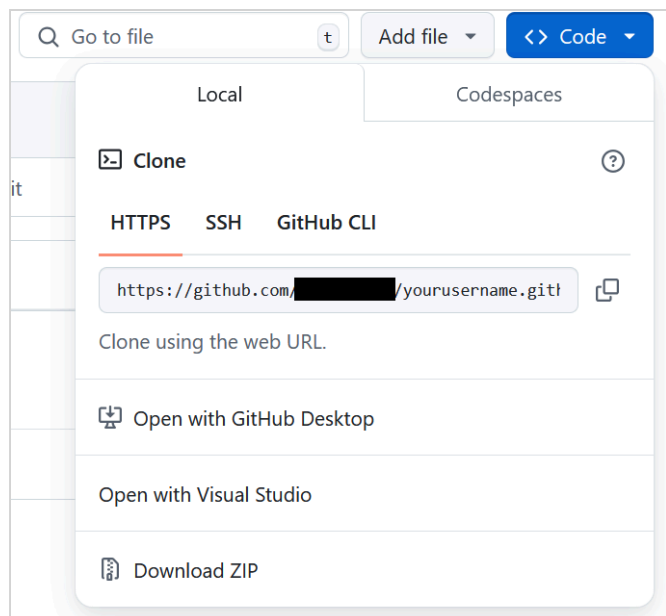
This will set `main` as the default branch. Change the default name in your [settings](#).

ⓘ

 You are creating a public repository in your personal account.

Create repository

You now have a repository that only contains one file, namely `README.md`. You must now clone a copy of the repository on your local machine. First you will need to find the Remote URL. When the repository is created you can find the Web URL under the Code button:



✂ Exercise 1.2: Open your command line and navigate to a directory where you want to clone your repository. Use the `git clone web*url*` command to clone the remote repository onto your computer. Use the `*weburl*` you retrieved from your repository. If it is the first time you use the clone command, git will ask for your GitHub credentials. Feel free to enter it.

⚠ DO NOT use the Download ZIP feature. This will only create a temporary instance of your repository and it will not be linked to the remote repository.

## Commit the Files Locally

Once you have cloned the repository, you should see a new directory called `yourusername.github.io` on your computer and it should contain a `README.md` file. On your command line, navigate to that directory.

✂ Exercise 1.3: Move your `profile.html` into the the directory containing the `README.md` file

The next step is to commit the new project file(s) into git.

✂ Exercise 1.4: Add the file(s) to a commit and commit this to Git by following the steps:

1. Add the `profile.html` file with the command: `git add profile.html`
2. If needed add other files with the same `git add` command
3. Check the status ( `git status` ) to see if all the files are mentioned under "Changes to be committed"
4. Commit those changes to the repository's history with a short (m) message describing the updates like: `git commit -m "Created profile.html"`

## Commit them to GitHub

And finally, you can **push** (send) everything you've done locally to your remote repository on GitHub. This is something you'll do often so that your remote version is up to date and matching the state of your local

version.

✂ Exercise 1.5: push your local changes to you remote repository on GitHub with the command: ``git push``. If the response doesn't contain some error, you should be fine. Check your GitHub repository with a browser. You should find your files in the code.

## Finalising

Now, let's check if everything is as it should be for developing your website and according the rules in section **Final Result** above.

✂ Exercise 1.6: create one screenshot where you can see:

- A browser window with the content of your `profile.html` and the url: `https://username.github.io/profile.html` (where `username` is your username on GitHub)
- A browser window with the content of your `profile.html` and the url: `http://127.0.0.1:5500/profile.html` (or `http://localhost:5500/profile.html` )
- An (Windows) explorer view that clearly shows the **location** (i.e. `C:\projects\...`) and the **content** (files and folders) of the portfolio project folder

💡 If you feel that you need more help on this subject, you can install Git-it; a special desktop app that teaches you how to use Git and GitHub on the Command line:

[GitHub - jlord/git-it-electron: Git-it is a \(Mac, Win, Linux\) Desktop App for Learning Git and GitHub](#)

The first five chapters will basically do the same as following exercises except for a few settings:

- **Get Git:** you should already have done this (Preparation Assignment)
- **Repository:** Instead of a new folder, use your project folder instead (see Exercise 1.2). **Warning:** Exercise 1.3 is not mentioned in Git-it, but you must do it now
- **Commit to it:** Instead of the readme.txt file, commit your project file(s) as in Exercise 1.4
- **GitHub:** you should already have done this (Preparation Assignment). You can use the username of that account.
- **Remote Control:** Use the repository name as described in Exercise 1.6

# Good sites

---

## Workshop

In this workshop you learn that good sites are about:

- **Maintainable code**
  - Your project folder is *structured* with folders like `resources` , `css` and `js` .
  - File and folder names should always be in *lower case*
  - Always choose *meaningful names* for files, folders, classes and ids. Whatever casing you use (all caps, all lower case, camel, pascal, snake or kebab casing) choose one that seems appropriate and use this *consistently* throughout your project for that type of 'thing' you want to name
  - Use a consistent *indentation style* for your code
  - Prevent too much *blank lines* in your code. Place them only when appropriate
  - Prevent *code duplication* whenever possible
- **Usable site structure**
  - Your content is divided into *multiple pages*. Each page should show a *coherent* set of content and design
  - There should be a *landing page* ( `index.html` ) that helps new arrived users to understand what your site is about and make a first impression
  - Help your users to navigate throughout your site with a *navigation menu*
- **Findable site**
  - Choose appropriate `<title>` s for each page
  - Use HTML5 semantic tags as they should be used
- **Purposeful appearance**
  - The appearance should be internally consistent throughout your site
  - Use CSS to manipulate the site layout, use of color and typography

You added the following to your website:

- You added an `index.html` file
- You added a rudimentary navigation menu to both pages
- You added some CSS to style the navigation menu

## Self study

During the self study you apply and practice with what you have learned so far and learn some new things by yourself.



The exercises build on what was covered in class. If you did not attend class, you should do that first before you continue with the exercises.

## Part 1: skeleton site

In the [Assignment Specification](#) you can find all requirements the site must meet. Among these there is a list of required pages. This means that your site **MUST** contain all these pages.

✂ Exercise 2.1: Check the Assignment Specification and create the rest of the required pages, but only add the basic html structure and not any content yet. This way you create the entire sites basic structure.

Because you added new pages to your site, you now must update the navigation menu

✂ Exercise 2.2: Extend the navigation menu in both `profile.html` and `index.html` so your users can navigate through all the pages of your site. Also, add the same menu to all the new pages.

⚠ When done, commit the resulting changes to your Git repository and and push them to GitHub.

## Part 2: learn CSS

Before you can start designing the layout of your website, you should learn some CSS first.

✂ Exercise 2.3: Walk through the entire "Learn Basic CSS by Building a Cafe Menu" course in FreeCodeCamp.

[freeCodeCamp.org](https://freeCodeCamp.org)

💡 For those of you who are new to CSS, you can find out a bit more about it in the ["What is CSS" page on MDN Web Docs](#).

## Part 3: layout

So far, you have not really focused on the differences between a website and a webpage. These exercises combine all previous results into one coherent website with an eye-catching layout.

💡 Are the assignments too hard for you (don't feel ashamed, it's a lot! Especially if you're new to programming)? Then do the first 3 chapters of the CodeCademy course: [Make a Website](#). This course lasts about 3 hours, so take your time. A big advantage is that you can copy/paste most of the code that you come across to your own portfolio website and therefore gain some time.

Make sure to understand how to:

- manipulate the size, position and behavior of each HTML element by means of CSS classes and IDs in order to change the layout;
- consider a website as a collection of pages that are in tune in terms of content and design.

✂ Exercise 2.4: Use the layout concepts from the workshop, FreeCodeCamp and maybe CodeCademy to design the layout of your webpages. Start by sketching where you want to place the different sections like the header, navigation, aside and the main content of your pages. Then create the html structure in one of the empty pages and develop CSS to layout these sections. When done, copy and paste the structure in the other pages, leaving the content, if present, visible of course.

You could choose the layout from the CodeCademy course or create a visually attractive one yourself. A middle way could be choosing the existing layout and adjusting it. That option is probably the most instructive one.

💡 Tip: use a distinguishable `background-color` for each section to make them visible and distinguishable on your page. This helps you to develop the layout. When you're happy with the layout, change the colors back to whatever you want.

💡 Tip: A lot of sites use the so called *\*holy grail layout\** as the basic structure. Search the internet on that term to see loads of examples.

In addition, you could also use Bootstrap or another CSS framework (but no template!) to design your website. However, in case you decide to use some CSS framework, pay attention to the fact that you design the website content like tables, images and lists accordingly.

⚠ Don't forget to push the result to GitHub.

## Part 4: navigation menu

Now it's time to style your navigation menu from an ugly list of links into a real nav bar. Nicely styled nav bars should have features like:

- Change the background color of the links when the user moves the mouse ('hovers') over a menu item
- Change the style for the current link to let the user know which page he/she is on
- A full-height, "sticky" side navigation for vertical bars
- Inline or Floating List Items for horizontal bars
- Make the horizontal navigation bar stay at the top or the bottom of the page, even when the user scrolls the page
- Or, make the horizontal bar sticky

[W3schools.com](https://www.w3schools.com) has an article that demonstrates how to create either a horizontal or vertical navigation bar with these features and more.

### CSS Navigation Bar

✂ Exercise 2.5: Read the article on W3Schools (3 pages). Decide on which type of nav bar you need and try to understand what needs to be done in order to incorporate that into your website.

💡 You might want to consider a horizontal nav bar for the main navigation and a vertical for your aside section (see part 5 below).

Look up the difference between *absolute* and *relative* URLs. If you use absolute URLs, you might just find out that your website doesn't work on GitHub Pages. And when you finally get it to work there, it suddenly doesn't work on localhost anymore.... The sensible option is to use relative URLs.

✂ Exercise 2.6: Develop the navigation bar(s) in the layout of your website. Use `index.html` first and copy the changes to the other pages.

⚠ Again, this seems to be a good moment to commit and push your code to GitHub.

## Part 5: aside menu

The Assignment Specification states that the aside menu should contain a collection of links that are relevant to your study. It specifies exactly which links should at least be present

✂ Exercise 2.7: Create the collection using a `<ul>`. For each required link, find the appropriate URL and add this to the list. And, of course, style the collection as a menu.

💡 Tip: Let each link open the page in another tab in the browser. This way your user does not really leave your site when they click on that link. For more info on how to do this, see: <https://www.freecodecamp.org/news/how-to-use-html-to-open-link-in-new-tab/>

⚠ And as always, don't forget to push your website to GitHub!

## Part 6: The other pages

### The Profile Page

Refactor the profile page so it meets the Assignment Specification.

✂ Exercise 2.8: Check your current profile page. It should reflect at least what is specified in chapter 4 of the preparation assignment [here](#). Turn your personal information into an ordered or unordered list of characteristics of yourself using an `<ol>` or `<ul>`.

✂ Exercise 2.9: Add images, videos and/or audio that supports the content. At least one of the images should be downloaded and stored in a separate folder (named `img`, `resources` or similar) within your project. Use a relative URL to link the image in your HTML structure.

⚠ Don't forget to push your website to GitHub! You now may have figured out that it is good to push your code every time you finished a specific piece of functionality in a project. This is what professional developers also do.

### The Blog

The content of this page should be a *list* of blog posts or better: blog post excerpts with links to a page that shows the entire post. When using semantic tags like `<article>` and `<section>`, this leads to a discussion where people still not agree upon: should you nest `<article>`s inside a `<section>` or the other way around? According to W3Schools:

Can we use the definitions to decide how to nest those elements? No, we cannot! So, you will find HTML pages with `<section>` elements containing `<article>` elements, and `<article>`



elements containing `<section>` elements.  
[https://www.w3schools.com/html/html5\\_semantic\\_elements.asp](https://www.w3schools.com/html/html5_semantic_elements.asp))

✂ Exercise 2.10: Search for at least 3 articles on this discussion, and form your opinion about this subject. Use this to create the skeleton of the Blog page.

✂ Exercise 2.11: Use the WHO AM I assignment, the programming experience and the feedback of the teachers to create the required blog posts to meet the Assignment Specification, except the last one. This is about ICT companies, and you will have an opportunity to gather some information about that later this week.

💡 The `<summary>` element has some nice functionality out of the box. Experiment with this element to see what's happening.

## The FAQ

The FAQ page is also, like the Blog, a list of content parts. If you are happy about the Blog page, you can use the same structure to create the FAQ page.

✂ Exercise 2.12: Use all the information you remember, the HZ website and/or your fellow students/teachers to find the answers of the required FAQ questions. And, of course, you may add your own as well.

## The (personal) Dashboard

An important goal of the entire project is that you get acquainted with the school and the HBO-ICT programme. The following exercise lets you discover what courses and exams lie ahead of you.

✂ Exercise 2.13: Use the Implementation Regulations (IR) of the HBO-ICT programme and your study progress in MyHZ to discover the courses and exams you will attend this year. Create an HTML `<table>` and add a row for each exam. Add columns so the the names of the Quartile, course and exam can be noted and the amount of credits that you can earn for that exam. Also, add a column where you can add grades for the exams.

An important issue in the propaedeutic year year is the so called NBSA boundary. NBSA stands for Negative Binding Study Advice. See the The HZ Course and Examination Regulations (CER) for more information about this.

✂ Exercise 2.14: Find out what the \*NBSA boundary\* means. Design a visualization of your study progress with respect to this boundary that helps you maintain an overview whether or not you are in danger for an NBSA at the end of the year. Incorporate the design into your dashboard.

💡 A \*burn down chart\* might be a good idea for this.

## The home page

The home page will play an important role during your poster presentations next week. You might already want to think on the first impression you would like to give to your audience.

✂ Exercise 2.15: document your first thoughts about the home page in the home page. So, create a basic version of the home page.

✓ Remember that you can always update the content of your website at any time during this project. You can add, remove or change content if you think it is needed or if you like. But keep in mind that you always must meet the requirements in the Assignment Specification.

# Code Review

💡 Code Review, or Peer Code Review, is the act of consciously and systematically convening with one's fellow programmers to check each other's code for mistakes, and has been repeatedly shown to accelerate and streamline the process of software development like few other practices can.  
<https://smartbear.com/learn/code-review/what-is-code-review/>

## Lecture

In this lecture you will learn why and how to give and review your code. You will learn that reviewing code is common practice and there are several techniques to review code: face-to-face, online or even tool-based.

## Skills Lab

In the period between lecture and checkpoint you collect as much feedback on your website and working environment as possible using the *over the shoulder* technique. You will need your fellow students for this, so we advise you to do this on location if possible.

## Over-the-shoulder code review with fellow students

This step must be executed in groups.

✂ Exercise 4.1: Form a group of 2-4 students. Arrange a time and place where you will meet: one of the available classrooms, the restaurant or other rooms or book a project room. It would be nice if you have a large screen available to show the code/website when reviewing.

✂ Exercise 4.2: Review each other's website one by one (about 20-30 minutes per website). The website of each student is reviewed by the rest of the group. Use the following instructions: [Website Code Review Instructions](#)

✂ Exercise 4.3: Discuss the results within your group. List the defects and errors you found within your group. Were there any defects and errors that have not been mentioned in the instructions? Make a list of these as well.

## Checkpoint

☑ Bring all your notes from Exercise 4.3 to the checkpoint.

This will be discussed with the rest of the class and we might add them to the instructions to help next years' students.

# After the Checkpoint

## Experiment with tool-based reviews

This step can be done individually.

First, you try an easy way to collect generated feedback on HTML documents (not CSS).

✂ Exercise 4.4: Use the [FreeFormatter.com](#) (see the link below) website to collect automatically generated feedback based on best practices. You must upload each page at a time, collect the feedback and add it to the feedback you got from the previous exercise.

[Free Online HTML Validator - FreeFormatter.com](#)

FreeFormatter is just an example a static analysis tool. It is also possible to get the same kind of feedback in your IDE as you type. Most IDEs already have some basic functionality built-in. But, they generally can be extended with more language specific and configurable tools.

✂ Exercise 4.5: install HTMLHint which integrates the HTMLHint static analysis tool into Visual Studio Code. If you use another IDE, look for an extension/plugin that does the same. If successful, look at each HTML file and see the feedback it generated.

[HTMLHint - Visual Studio Marketplace](#)

HTMLHint checks only HTML code. In order to check CSS, you need another extension.

✂ Exercise 4.6: stylelint is an extension that does the same with CSS code. Install this extension into VSCode (or a similar if you use another IDE). Check your CSS files and, again, look at the feedback.

[stylelint - Visual Studio Marketplace](#)

## Process the feedback

You have collected all kinds of feedback from the code reviews and linting tools. Now it's time to see if there is anything to learn from it (fallacies) or it was just carelessness.

A *carelessness* is generally something you forgot just once or twice. For instance, it might occur once that you used capitals in a file name. But if it happened more, like using capitals and other special characters in file names all the time, you might encounter a *fallacy*: something you did not understand well enough to do it correctly. In the example, maybe you did not understand the fact that different OS (Windows, Linux, MacOS, etc.) have different rules on the use of capitals and special characters, and might lead to errors if you deploy your site.

Besides this, there are other perspectives to think about these carelessnesses and fallacies. Do you have significant more or less carelessnesses with respect to others in your group? Are there common factors between carelessnesses you can find? Something you do in a certain way that causes some of these defects and errors? A reason why you keep forgetting certain things? Something you still do not know about websites, HTML/CSS or how to manage your computer? Why am I getting almost no real feedback?

✂ Exercise 4.7: think about all the feedback you got from the previous exercises. List all the feedback and classify them as carelessness or fallacy. Describe at least 3 of these fallacies in more detail. Add something that shows what you have learned from it to prevent it next time. Add your observations about when you think about the feedback from other perspectives. Describe at least one.

⚠ Please note: these descriptions will be used for another course later in the quartile and can impact your grade for said course if you do not have (good) descriptions.

🔧 Exercise 4.8: Now, use this feedback to improve your code, preferably before next class. If there is a lot to improve, try to prioritize. Are there easy/quick fixes? Is there something structural to change that is best to be done first before other changes makes this more complex?

# Website Code Review Instructions

---

## Introduction

These instructions will guide you through a structured code review of your portfolio website.

- Method: Over-the-shoulder
- Duration: 20-30 minutes per website (or 30-45 minutes when online)
- Assignment specification:

Assignment Specification

## Roles and responsibilities

According to this site (<https://www.geeksforgeeks.org/roles-and-responsibilities-in-review/>) there are many different roles when doing a more formal review. Especially two of them are relevant for now:

- **Author** : As the writer of 'document under review' (the code in this case), author's main goal must be to learn and know as much as possible with regard to improving and increasing the quality of the code.
- **Reviewer** : Reviewers only have to check all of the defects and errors and then further improvements also in accordance to business specifications, standards, and domain knowledge. In other words: reviewers check for defects and errors with respect to the assignment specification and general quality rules

## Setting

The author shares his screen to the rest of the group. Start with step 1. The IDE, project folder location and other relevant items are demonstrated by the author. The reviewers look at the relevant section of the assignment specification and the frequently made mistakes. When needed, reviewers ask for specific items to show.

The reviewers mention everything they consider possible defects and errors. The group discusses this for at most one or two minutes and reaches a consensus on how to improve it. The author takes note of this in a TODO like manner (see the **Taking notes** section) so that he can incorporate it later. If no consensus can be reached, ask for a teacher or student assistant (park the subject for now and continue the review if you need to wait for a teacher/student assistant).



Besides mentioning defects and errors, reviewers might also provide tips & tricks on how to solve some of these issues.

Try to timebox each step at a max of 5-7.5 minutes. Then continue with steps 2, 3 and 4. When the review is finished the next student will demonstrate his project. Continue until each student is reviewed.

## Taking notes

As mentioned, the author takes notes about the feedback (changing the code immediately is not advised, because it will be more time consuming and may lead to other errors). This should be in a TODO like

manner. Try to describe what you need to do here in order to process what has been discussed. This might include things like “decide on how to improve ...” or “Ask teacher if ... is correct or not / how ... can be improved or should be solved”

You might want to do this with comments in the code (like `<!-- TODO: your comment here --!>` in HTML documents or `/* TODO: your comment here */` in CSS).

## STEP 1 - Development environment

Demonstrate and discuss if the working environment meets the requirements in section **Hosting and working environment** of the assignment specification. Check at least the:

- Project folder location
- IDE (e.g. Visual Studio Code)
- Webserver (e.g. Live Server)
- Hosted environment (GitHub pages and the GitHub repository)

Frequently found defects and errors here are:

- IDE is NOT used in project mode (files are opened by the OS file explorer and not via the IDE project explorer view)
- Files are not served by the webserver (you see links like `file:///...`)
- Need to copy files from project folder before they can be committed to the GitHub repository
- Folder and filenames contain uppercase characters, spaces and/or other special characters
- The GitHub repository root holds the project folder instead of the project files themselves


## STEP 2 - Design and functionality

Open the Home page and discuss the **Design** and **Functionality** sections of the assignment specification. Discuss on:

- Does the design fit the requirements (remember that *taste* is just an opinion but you can discuss whether or not a certain design communicates the required message)?
- Does the navigation menu function as required?
- Does the aside menu function as required?
- Are the SEO requirements implemented?

Frequently found defects and errors here are:

- The URLs in the navigation menu are not relative and/or contain `file:///...` paths
- Inconsistencies in the design like bad alignment of elements or abnormal scrolling behaviour

 Steps 2 and 3 can be combined. Just make sure you cover every topic and frequently found defects and errors.

## STEP 3 - Content Structure

Go through all the pages and check if they meet all the requirements in the **Content Structure** section of the assignment specification. Discuss on:

- Are all the required pages that are specified in the site map present?
- Whether each page has all the required content and if it meets its purpose

Frequently found defects and errors here are:

- Not enough of the required HTML elements present, i.e. 2 images in the home page
- Everything in one large page
- Structure and content of the table in the dashboard page
- NBSA boundary in the dashboard page
- Blog posts are not in separate pages

## STEP 4 - Code Quality

Go through the code of each page and discuss the code quality.

Frequently found defects and errors here are:

- Indentation errors - misalignment of opening and closing tags
- Inconsistent use of uppercasing; i.e. `<DIV>` and `<div>` or even `<Div>` used throughout the project
- Improper naming; especially CSS classes and IDs; kebab-casing is often used for classes
- Too long lines - horizontal scrolling should not be needed
- Code duplication - a CSS file for each page where lots of the CSS code is duplicated



# Visual Design

## Workshop: Visual Design

During the workshop on Visual Design, you:

- Refactored the study monitor table using an HTML `<table>` with *merged cells* (see: [https://www.w3schools.com/html/tryit.asp?filename=tryhtml\\_table3](https://www.w3schools.com/html/tryit.asp?filename=tryhtml_table3)).
- During this, we also discussed the details about the study program. Especially about the *EC's* you can to get for each *quarter* of the first year and pass the *NBSA boundary*. But, also about the *semesters* of the following years until your *graduation internship*.
- Next, we introduced some alternative layout design techniques like flexbox and grid that you might want to consider as a more modern alternative for `<table>`.
- Finally, we inspired you with more advanced CSS tricks that you might want to incorporate in your website

## Skills lab

### Building the table

During the lecture we discussed how to structure the dashboard with a `<table>` and merged cells using `rowspan` and/or `colspan`. We also discussed briefly the entire structure of the first year of this program.

✂ Exercise 6.1: Use this information to create a complete HTML structure in `dashboard.html` that includes all first years test components. As discussed, they should be grouped by course which should be grouped by quarter.

### Designing the dashboard

The following exercises can be done individually or in small groups.

#### Styling the table (visual design)

According to the requirements specification, a clear distinction must be made between parts that have not yet been completed, sufficient parts (i.e. parts that have been obtained) and insufficient parts (still to be retaken). For example, by using colors.

✂ Exercise 6.2: Think about how you want to visualize this distinction. Would you like to use colors, a different font setting (like bold or italic), a combination or something completely different? Draw your ideas on paper. Visualize at least:

- A fully completed course
- A course that has no grades yet
- A course, consisting of two exams, where only one exam is passed. The other exam still has no grade
- A failed exam

- A course, consisting of two exams, where only one exam is passed. The other exam failed

## The completeness visualization

According the requirements, the completeness of your first year (the amount of obtained credits relative to the required amount of 60EC) must be clearly visible. This should also include the “NBSA boundary” (of 45EC).

The visualization of completeness is something that a lot of different application also need. And most of them have solved it in different ways. This problem (visualizing process completeness) and possible solutions are documented as *UI design patterns*. There is a website that has documented a lot of these patterns: [ui-patterns.com](https://ui-patterns.com). Here is the link to the *Completeness meter* pattern:

### Completeness meter design pattern

✂ Exercise 6.3: Check this link. See the examples for inspiration on how you want to visualize the completeness, including the NBSA boundary. When possible, discuss these with fellow students. Then draw your ideas on paper. Draw different situations like:

- All exams up to now were successful, but there are still some exams left to do. You are still below the NBSA boundary
- Same as before, but you are now above the NBSA boundary
- All available 60ECs are obtained
- You passed some exams, but failed others. It is still possible with the resits and other exams left to pass the NBSA boundary
- Same as above, but with the resits and exams left, it is impossible to ever pass the NBSA boundary

## Handing in before Checkpoint

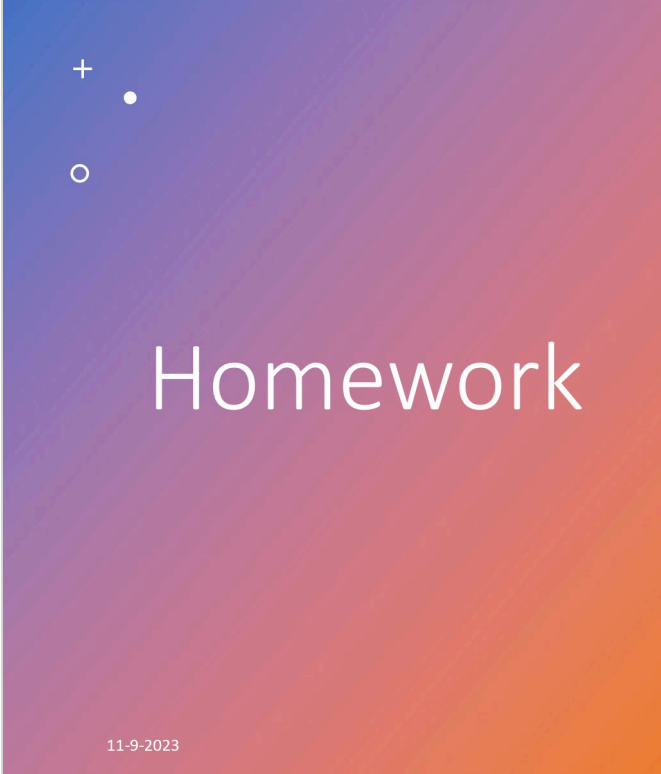
During Checkpoint we like to discuss different designs.

✂ Exercise 6.4: Upload your designs on learn in assignment Exercise 6.2 and 6.3 - visual dashboard designs. You may upload one set of designs per group or individually

## Self-study

In this part of the course you can choose between the FreeCodeCamp subjects to add more CSS to your website.

✂ Exercise 6.5: On [FreeCodeCamp](https://www.freecodecamp.org) (under Responsive Web Design Certification) you'll find the following subjects, pick one of the subjects and follow the lessons



# Homework

11-9-2023

- Browse to [FreeCodeCamp's New Responsive Web Design](https://www.freecodecamp.org/learn/2022/responsive-web-design/) course
- <https://www.freecodecamp.org/learn/2022/responsive-web-design/>
- Scroll down until you see **Learn CSS Flexbox by Building a Photo Gallery**
- Now, pick any combination of **two** of the projects, starting from Flexbox.  
For example: **Typography** and **Animation**
- Work through your chosen paths and add learned elements to each page of your own website

10

✂ Exercise 6.6: Apply what you've learned to your own website. For instance, if you learned Flexbox, Rework your Dashboard page so the study monitor is constructed with a Flexbox instead of a `<table>`.

✂ Exercise 6.7: Apply other CSS tricks anywhere you want in your site. Use the inspiration you got from this workshop and/or other resources you find. Be creative and experiment. This exercise is also about learning and trying new things. Also, have fun trying but always keep in mind the general usability rules (like coherent design). Spend at least three hours applying your new-found tricks to make your website look as good as possible.

💡 Tip: Don't worry if your experiments fail. Because you use Git, you can always reset your project to the latest commit, using the `git reset --hard` command.

# Motivation & Presentation

Motivation is an important part to help you through your studies. Are you participating in the right study for you and why? You have seen and done a lot these past few days. What's your impression? We would like to know and so the following exercises will help you with your HOME page.

✂ Exercise 7.1: Update your home page according the following:

1. Describe first what your motivation for ICT is, use your who am I form
2. Then argue as to why ICT is the right study for you. What is to be concluded regarding your motivation?
3. Support this with proof. For instance, you have done preliminary coding work or a former education, projects, hobby or your showcase website speaks for itself, information gathered the past few days, etc.

✂ Exercise 7.2: Watch the TED talk below. Make notes about Dan Pink's views and ideas. According to him, what is motivation? What determines someone's motivation? How can you influence and manage your own motivation? Look over you motivation page and update your home page if needed.

[https://www.ted.com/talks/dan\\_pink\\_the\\_puzzle\\_of\\_motivation?utm\\_campaign=tedsread&utm\\_medium=referral&utm\\_source=tedcomshare](https://www.ted.com/talks/dan_pink_the_puzzle_of_motivation?utm_campaign=tedsread&utm_medium=referral&utm_source=tedcomshare)

✂ Exercise 7.3: Make your page visually more attractive by adding images, etc. and the layout of the text blocks.

## Poster Preparation

The poster presentation is done by presenting your website on a A3 poster and by showcasing your website on your laptop.

Your job is to *convince the teachers and fellow students of your creativity, professionalism and motivation*. You can become creative here in creating a poster however the combination of your laptop with the poster needs to give a good view of your portfolio website and your development environment. To be prepared you:

1. Know *what* you will be evaluated on - You study the feedback forms that are used during the evaluation.
2. Know *how* you will be evaluated - you study the presentation procedure
3. Finish your portfolio website so that it complies with the assignment specification and supports your poster presentation in the best possible way.
4. Prepare and practice your presentation. You find some general tips here even though it's about a research:

<https://www.youtube.com/watch?v=vMSaFUrK-FA>

✂ Exercise 7.4: Download the feedback form of your language from learn [here](#). The forms are located in a folder on the main assignment page. Open the form and find out where the teachers and fellow students

will pay attention to during your presentation.

The form consists of numbered *rubrics*. A rubric is a row where each cell represents a higher level of complexity (reading from left to right). Sometimes it's just a number (scale) and sometimes each cell contains a description. Each rubric will be scored by marking one of these cells. If a cell is marked it means that the teacher is convinced that you have met the entire description in that cell (mostly including all cells on the left).

By studying the form you have an idea of what to do to get a good result. Next, you need to know what you need to do during the poster presentation in order to enable the teachers to evaluate you and keep the procedure efficient so you don't run out of time.

✂ Exercise 7.5: Read the presentation procedure on the poster presentations section . Make sure you know how to present and what type of feedback you need to gather.

✂ Exercise 7.6: Finish your portfolio website so that it complies with the assignment specification and supports your presentation in the best possible way. While doing, already try to figure out for yourself what your presentation should look like (what to tell when and how to use your website).

✂ Exercise 7.7: Prepare and practice your presentation (make a concept print of your poster to see if the text and pictures are visible/readable). Use the HZ printer A3 size is OK

# Online Company Safari

If you want to really dive into finding your motivation and go the extra mile then make this additional assignment to get more insight into your future potential job. In November there will be a "We are in IT Together" conference giving you more insight into the professional IT environment. But you will first try and find a "role model" company which peaks your interest and helps with investigating your study motivation. **The results of the research done will be processed in a blog post on your website.**

As you may already know, there is no such thing as the IT professional job. As an undergraduate IT professional you have many options to choose from in the IT field. Those options can be categorized into:

1. Choice of organization: area of business, size, and mission.
2. Choice of function: general and/or specialist in a specific role as developer or security expert.
3. Choice of technical field: web, apps, enterprise applications, data science, etc.

Almost every organization has to deal with some form of IT. That is why the job market for an IT-professional is huge. You can work in many different areas after graduating. For mid-term decisions it is important to get a more detailed understanding of the IT field and possibilities. It will help you to make the right choices for study specialization, your minor and internship. For short term, this study year, it is important to find out as much as you can to help you decide if this is the right study for you and stay motivated.

## Assignments on Research

✂ Exercise 5.1: First recall your written who am I form motivation. It might give you some idea where to start your search for IT companies. For instance if you like gaming, you might want to find out about companies in that area. If you like program languages look for software engineering companies. **Write down** some search words that you can type in your favorite search engine to find IT companies. Examples are: Software engineering, Data Science, Business and IT, Gaming, Cyber security, C#, JavaScript etc. Combine these search words to find better results and combine them with the word "Company" and an area where you would like to work (Netherlands, UK etc.) see what results the search engine presents you. Of Course when searching Dutch companies translating these into Dutch words will also help.

✂ Exercise 5.2: Select from the found results **three** companies of interest. In other words companies you would want to work for. Maybe you already have a role model company in mind. Like wanting to work for Microsoft or Google.

✂ Exercise 5.3 Now with those **three** selected companies gather more detailed information about them. Visit their websites and read the information. What you want to look out for is the following:

- What type of projects / work do they do?
- What IT technology do they work with?
- How big are they in numbers of staff? (Small business, multinational etc.)
- What do they tell you about there company culture?
- Do they operate international,
- what different job descriptions do they have.

You can use different sources to find you information. Youtube channels to get an impression of culture and technology or projects they work on. [linkedin.com](https://www.linkedin.com) for the vacancies and jobs they have or through a job offer page of the website (Dutch intermediair) or <https://www.stepstone.com/en/solutions/jobseeker/>

✂ Exercise 5.4: If you have gathered enough information on the company next see if you can find out about job specifics. What would for instance a data scientist do? or a software engineer versus a Business and IT consultant. This will help you for choosing a track in the future

## Analyzing your information

You have gathered a lot of information now on what we call doing Desk research. Now it's time to process that information into a summary for your blog. Keep it short but detailed enough.

✂ Exercise 5.5: Gather the following information and place your findings on your portfolio website by doing the following:

For all the three companies make a paragraph per company where you shortly state something about your findings. To help you find the checklist below to see if your text includes the following items. (if you could not find on or two that's fine)

- Primary task also referred to as core business. In short what do they do
- Different functions or roles you can work in
- The specific technologies or services they work with / perform
- What is the company culture (dusty, business like, casual, professional etc)
- Put in the links to where you found the information (referencing is important)
- Add a sentence or two on if you see yourself working here and why, what speaks to you?

✂ Exercise 5.6: Create a blog article named "Profession" on your website. Do not place the above findings in a bullet point manner but try to create an attractive coherent visual design that combines your findings in visuals, and text.

💡 TIP: Browse websites, commercials, and job application websites to get a feel for roles, techniques, and culture. Take a good look at the used colors, visuals used by the companies, videos, and scan through articles published. You might also find more information by taking a look at their customer reviews.

# Poster Presentations

---

## Before Presenting

- Prepare your website. Make sure all your pages are finished (or as close to as you can)
- Prepare and print your poster

## Presenting

The poster presentation a very casual way of presenting where the assessors and you yourself walk around the room to look at each others work. You must also give and receive feedback during the session.

You will present both your website and your poster. During the presentation, the following will be looked at:

1. **Your Poster:** The front end, design, and presenting the information on the pages
  1. Is your site showing your eagerness for this studies by everything you have done
2. **Laptop:** Back-end coding and technical skills
  1. Does your code and working environment show your ICT talent?

## Give and Gather Feedback

During the presentations, you must give and gather feedback from lecturers and fellow students.

- Give feedback by filling in the feedback forms ([Dutch](#) and [English](#) version) for three fellow students websites
- Additionally, gather feedback from at least one assistant or teacher

💡 Feedback should be about improvement of your website to make it even better to be able to finalise it for the assessment. It could also be about applying a technical solution somebody offered you.

## After the Presentation

### Do this after the presentation is over

- Process the feedback into one document
- Scan the feedback forms to add them to your upload later.
- Make a pdf of your poster and merge with feedback forms
- Upload on Learn

### How to process the feedback into one document

In a Word document you:

1. Create a heading called **Compliments**. Add all the feedback items that you consider as compliments. Order them so the items you feel most important are on top.
2. Create a heading called **Improvements**. Add all the feedback items that you consider as things to improve. Describe each item as carelessness or fallacy (see exercise 4.6).



3. Create a heading called **Code Review Feedback**. Copy and past your code review feedback (see also exercise 4.6) here and describe which of the items you think are already improved and wich items you also see in the presentation feedback.
4. Create a heading called **Tasks**. Add at least 5 tasks, based on the feedback, that you want to perform in the following weeks to improve your website and/or presentation.

! Attention: these tasks are an important input for the final assessment at the end of this quarter.

### How can you scan

You can easily scan the documents at the HZ using one of the printers in the building. You should have the scan sent to your email address as a .pdf document. This does require you to have an active HZ account.

### Save your poster as a pdf

Search for a practical way to make a screenshot of your site and merge it with the scanned form and your analysis word document into 1 PDF document. If you really don't know how to do that:

1. Print one page of your website on paper
2. Print out your analysis on paper
3. Then scan again all forms together with your analysis using a HZ printer

Of course, there is a smarter way for you to find out...

### Submit the PDF document in learn

See the link **Hand in feedback and repository**

### Add the following text when submitting:

1. The URL to your site.

# Assessments

---

## Prepare

During the assessment, you will show your portfolio website according to the specifications but will also show what you have changed according to feedback received. Please make sure to check the [assignment specification](#) to see if you have everything you need.

- Book your assessment slot. On Learn you can find a scheduler where you can book an assessment time and room. First come, first served
- Make sure you know where you need to be and be there 5 minutes beforehand. The Learn scheduler shows the room.
- You have to present the website on screen, make sure you tested this before the assessment and have the appropriate converters for your laptop connection
- Have your website started and ready to show before entering the project room

## During the assessment

- The purpose of the assessment is to be able to prove to yourself and us by means of your website and gained knowledge that this is the right study and environment for you and you have applied some or all of the feedback received.
- We will give you some tips, ask you questions or give advice which you need to process in your final feedback document.

## After the assessment

Take some time to process the received feedback, the information and experiences gathered in the last couple of weeks which should have given you an impression about the ICT program. Write a summarized piece of text (no more than an A4 sized page) and conclude if or if not this is the right program for you.

**!** Exercise: Make a PDF file of your conclusion. Also make a ZIP file of your Git repository (your website) and upload both files together with your poster and feedback that you received at the poster presentation at the appropriate hand-in assignment on Learn. After your assessment, you have **one week** to complete this exercise and hand in your work. Your grade can't be finalised without your work handed in.

## Resit

If you obtained an insufficient grade or missed the first opportunity of the assessment, you will have to take the resit. Resits will be scheduled in the exam weeks of the first quarter. If your website is not ready at the first assessment opportunity, you can use this as a feedback moment. Just notify the assessor at the beginning of your assessment. When you resit make sure you have your website finished. When you do not obtain a sufficient grade on your resit or do not show up you must redo the assessment in the next study year.

# Assignment Specification

---

## Overview

*This section will give a basic overview of the project and the organisation behind it.*

The first two weeks of the ICT program consists of an assignment in which you convince the teachers that you are really motivated for the ICT study program by building a *showcase portfolio website*.

At the end of the two weeks you will present the result to a panel of teachers and students. How do you convince the teachers of your ICT talent and your motivation for this study programme?

ICT talent is foremost shown by your capability to develop your technical skills and being able to apply technical knowledge. It is about the willingness to learn, to try (and fail often), to seek (for help, for information). Whether you are an absolute beginner or someone who has already experienced in programming, ICT talent is shown by an attitude and development.

Motivation is foremost shown by communicating a message. Your showcase website contains a Personal Profile, where you present yourself. As an upcoming IT specialist, you must be able to convey your message towards a certain group (in this case your teachers). This is your opportunity to start developing that skill. What will I present of myself? What tone or words do I use to be convincing? What word font or colors do I use? Is my code well structured and readable? What other examples will I present that reflect the best who I am and why I am really motivated for this study program? And all the time you are aware of the target group, the teachers, you are sending your message to.

After this course, you can use this website to showcase your project outcome and other learning experiences. In the third quarter of this year you will use the site to learn about developing web applications.

## Goals

*Briefly description of the goals of the project. This will give you as a developer an idea of what you are trying to achieve, which will enable you to suggest the most appropriate solutions.*

So, your assignment is to decide on how to communicate that message with the website, select and learn the skills you think you need to show your willingness to learn. Develop the website and prepare yourself for that presentation. And, finally, do the presentation.

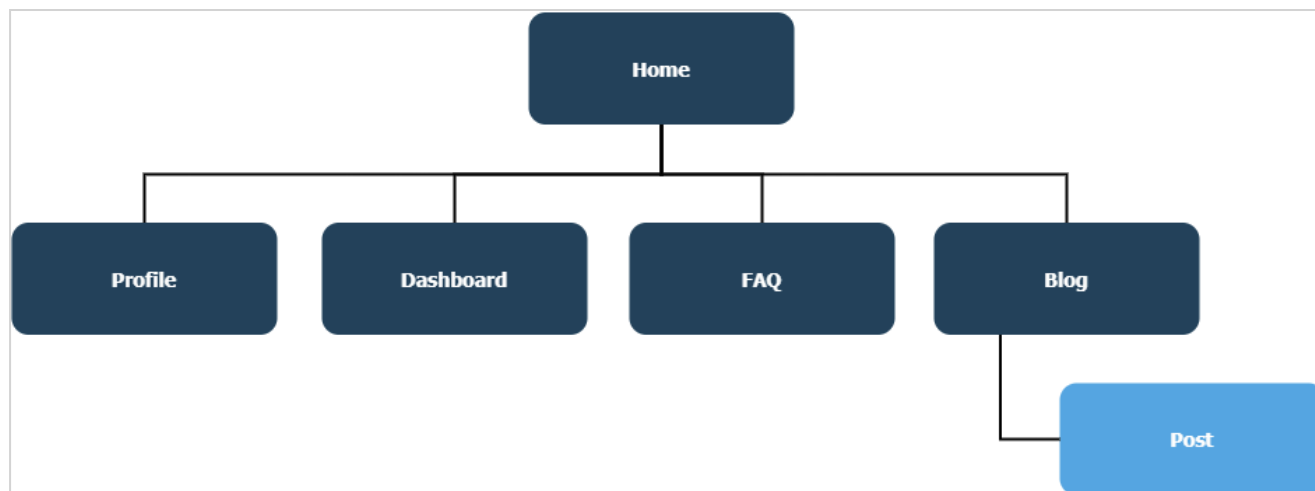
Below you will find all the requirements that your website must comply to. Use this regularly to check whether or not your project is complete.

## Content structure

*Content structure, or Information Architecture (IA), is comprised of various parts and will depend on the complexity and size of your website content.*

## Site map

*This is usually provided as a diagram which shows the 'tree' type, hierarchical structure of the website pages.*



## The home page (landing page)

*The landing page of the site (index) is the page that the visitor arrives at when he enters the URL of your website (without a specific .html file). This is often also referred to as the home or index page.*

The purpose of this page is to give the audience of your presentation your opinion why you think the HBO-ICT programme and/or the ICT field suits you. As this page is only your own opinion, the content can never be wrong. At worst it can be incomplete. You are free in the way you want to demonstrate this. The challenge is to do it in an original way and use the available technology (HTML/CSS) to show what you have learned in the past period.

The page must contain at least:

- 2 images
- 2 links
- 2 paragraphs of text
- 1 list (ordered or unordered) or table.

## The Profile Page

The purpose of this page is to give a brief idea of who you are. You could use this as an introduction of your presentation. The page should tell a member in the audience who you are. Where are you from? What hobbies do you have? Is there a specific movie you like to watch over and over again?

The profile page contains at least:

- 1 image or video
- 2 paragraphs describing yourself.
- 1 ordered or unordered list of characteristics of yourself.
- Link(s) to your GitHub profile and optionally social profiles (Instagram, X, TikTok, etc.) that open in a new tab/window.

## The (Personal) Dashboard

The purpose of this page is to create and maintain a clear overview of your study. This also shows to your student career coach that you have some detailed knowledge where you can find relevant information and what your activities in the coming year entail. The page contains at least the following parts:

The Study Monitor that shows the current status and progress of your study.

- It is based on an html `<table>` (or better, for example a CSS grid).

- All first-year test components ('things' for which you received a grade, such as exams and portfolios) must be included.
- The test items must be grouped by course and quarter in chronological order.
- A clear distinction must be made between parts that have not yet been completed, sufficient parts (i.e. parts that have been obtained) and insufficient parts (still to be retaken). For example, by using colors.
- The completeness of your first year (the amount of obtained credits relative to the required amount of 60EC) must be clearly visible. This should also include the "NBSA boundary".
- It must be easy to add grades and credits obtained (in the future).

## Frequently Asked Questions (FAQ)

The purpose of this page is to find and catalogue answers to questions you didn't even know you had. You must answer at least the following questions, but feel free to add more. For example, you might have struggled a bit with the coffee machine but you finally found out how it works. Catalogue it!

1. How can you print a document from your laptop at HZ?
2. How can you scan a document and send it to your laptop at HZ?
3. How can I buy something (like when I sign up for the IT introduction event) on the HZ web shop?
4. How can you book a project space?
5. What are the instructions if you want to park your car at the HZ parking lot?

## Blog/Post feed

The purpose of the blog is to create a chronological journal about your program and career orientation experiences. The page should contain a feed of excerpts of the available blog posts in chronological order (latest first). These excerpts act as links (i.e. 'read more...') to the actual blog post pages containing the full article.

It must **at least** contain articles on the following:

1. Study choice - a combination of the sections Study choice activities, Motivate your study choice and After completing the study... \*\*\*\*from you SKC document
2. Personal SWOT analysis - taken from your SKC document
3. Programming experience - taken from the SKC questionnaire
4. First feedback - The feedback you had from the SKC assignment and what your thoughts were about that feedback.
5. At least one article about the ICT field of work.

## Design

*This section contains guidance on the constraints and desired stylistic direction.*

You are free to choose the appearance of your site. However, the appearance of the site must be:

- Purposeful - it fits the **Goal** (communication the message specified in the **Overview** section)
- Uniform across all pages (the site is one entity).

The only specific design requirement you must fulfill is that the **HZ-logo** must be on at least one of the pages. You can download the HZ logos in PNG and SVG below.

## Functionality

*Functionality is how your site actually works. This could be anything about specific parts of the website that need additional explanation.*

## Navigation menu

The site must include a **navigation menu** that allows the user to easily navigate between the pages.

The navigation menu must highlight the current link to let the user know which page he/she is on.

## Aside menu

The site must include an **aside section** that acts as a menu to useful other sites. The `<aside>` element contains content that is only indirectly related to the main content of the website.

👉 Take note this is a not “a side” menu. This menu is aside. Therefore, it does not have to be on the side.

The aside menu contains a collection of relevant links (minimally 5). At least the following links should be present:

- The HZ HBO-ICT Course and Examination Regulations (CER) (see [hz.nl> About HZ> Regulations & Documents> Course and Examination Regulations](#))
- The Implementation Regulations (IR) of the HBO-ICT program (last year's is also allowed)
- This Learn environment
- The page in MyHZ with your study progress
- The GitHub environment of the study program (<https://github.com/HZ-HBO-ICT>)

## SEO

*Search engine optimization (SEO) is the art and science of getting pages to rank higher in search engines such as Google.*

At least the following SEO requirements from <https://www.socialmediatoday.com/news/8-of-the-most-important-html-tags-for-seo/574987/> must be fulfilled:

- Title tag
- Meta description tag
- Heading tags
- Image alt text
- HTML5 semantic tags

## Browser and device support

*Websites can be viewed on a wide range of devices and browsers. It is important to know which of these browsers and devices need to be supported, as their technical requirements can vary.*

*In particular, if you require support for older browsers (typically Internet Explorer) this can add to the overall project cost.*

The website must support the device and browser of your choice (typically your laptop). Support for mobile designs (*responsive design*) is not required, but allowed.

## Hosting and working environment

The site code should be developed using an adequate *IDE* (VS Code or similar) in which all relevant files can be opened as one project instead of single files.

The site can be demonstrated on your localhost webserver. So, not the url: `file:///...`, but `http://localhost...`.

The working environment (own laptop) is designed in such a way that editing, testing, and pushing the code can be carried out with as few actions as possible. In any case, files do not have to be copied.

The site must be demonstrated in a hosted environment using *GitHub pages*, *Bitbucket pages* or similar.

## Additional Technical Requirements

### Code quality

*Naming* files, variables, and other parts where you can choose a name is always meaningful and consistent in capitalization and case style. Specifically for file and folder names: file names must always be in lowercase and special characters like spaces are not allowed.

All code must be *styled and formatted* consistently throughout the entire project. This means that all HTML files must be formatted the same. Same for CSS and other file types if needed. At least indentation, capitalization, white spaces and lines must be formatted consistently. You may follow a more generic style guide like:

[Google HTML/CSS Style Guide](#)

### Frameworks, Libraries and other reusable code

The reuse of existing code like CSS frameworks (e.g. Bootstrap, W3.css or Bulma) is allowed. However, you are not allowed to use page or site generators or to copy entire sample pages from the framework.

This specification structure is based on <https://highrise.digital/blog/web-specification-guide/>