Arbeidskrav DS3103 Webutvikling

General Information

- The assessment of the arbeidskrav will be Godkjent or Ikke godkjent. One must get Godkjent on the arbeidskrav to be allowed to take the exam. One may have a second chance to deliver if one gets Ikke godkjent on the first attempt.
- Can be solved individually or in groups of up to 3. If you work in a group you will be expected to have more code than if working alone.
- The arbeidskrav consists of 2 main tasks:
    - A technical solution
    - Collaboration task giving and receiving feedback from other groups
- You are to solve the solution in the attached files. In the attached files there are images. You may freely add more images found on the internet to the project. If you want to publish the solution as part of your portfolio (for example on github pages) be aware of picture use. All attached images are royalty free.
- You must zip the project before upload.
- All techniques refer to those taught in the course.
- You may ask the teacher or student assistants for guidance (but not direct solutions) in the lab exercises.
- Have fun!!! 😊

Technologies

- HTML5, CSS3, CSS framework (you may choose Bootstrap or Tailwind), and vanilla JavaScript (ES6+).

Techniques and resources

- Semantic coding
- Diverse CSS code for styling content in combination with CSS framework.
- Grid system. Make use of both in your solution:
    - CSS3 Grid system (display: grid;)
    - CSS framework grid system
- Media Queries
- FontAwesome or icons from CSS framework
- JavaScript (main points):
    - ES6+ techniques such as arrow functions, forEach() and filter()
    - Array with objects
    - localStorage
    - module

Tips / things to reflect over.

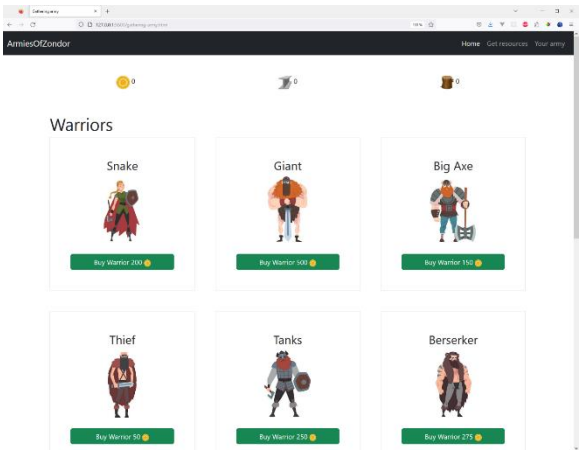
- Is the code tidy and structured?
- Are the variables and functions named properly?
- Is the HTML semantic?
- Do I have unnecessary code repetition?
- Is the code easy to read for other developers?
- Should I add a comment (HTML, CSS, and/or JavaScript) to mark and/or explain certain codes or sections?

Case: Web application for a medieval army

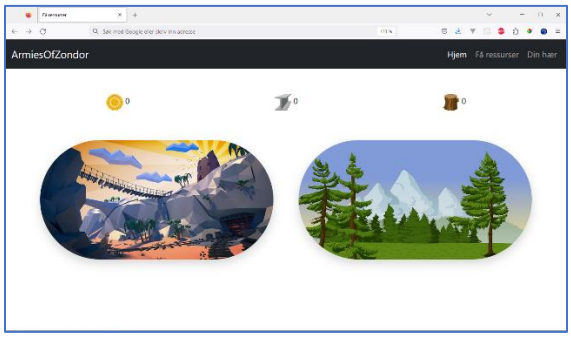
Note that all screen shots below are to exemplify the pages and functionality, and do not represent the complete functionality or design. You can decide freely on design according to what you think will be/look best.

Web pages quick overview

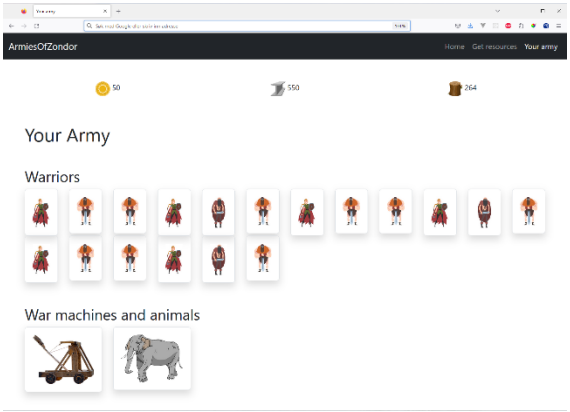
Page 1. Resource shop.



Page 2. Getting resources.

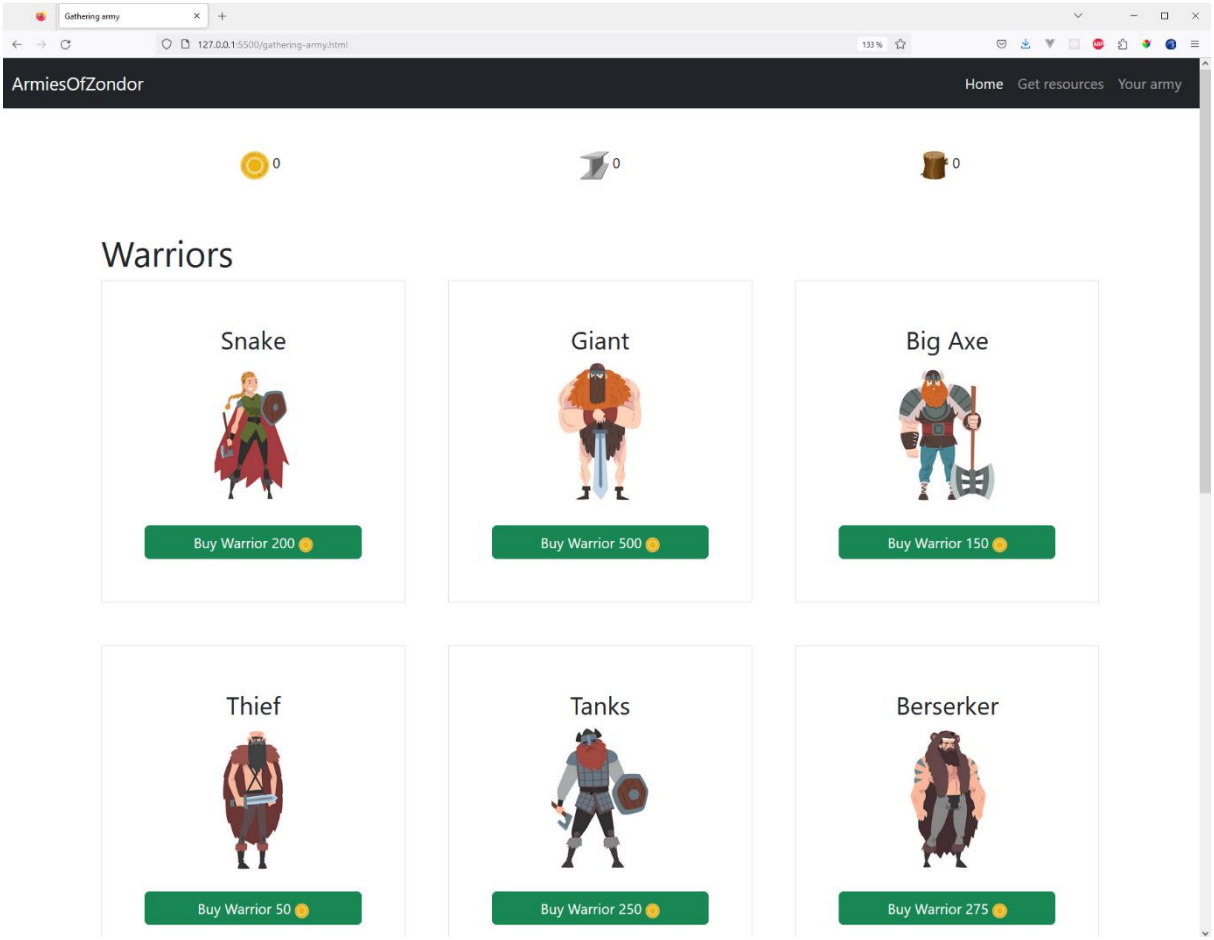


Page 3. Your army.



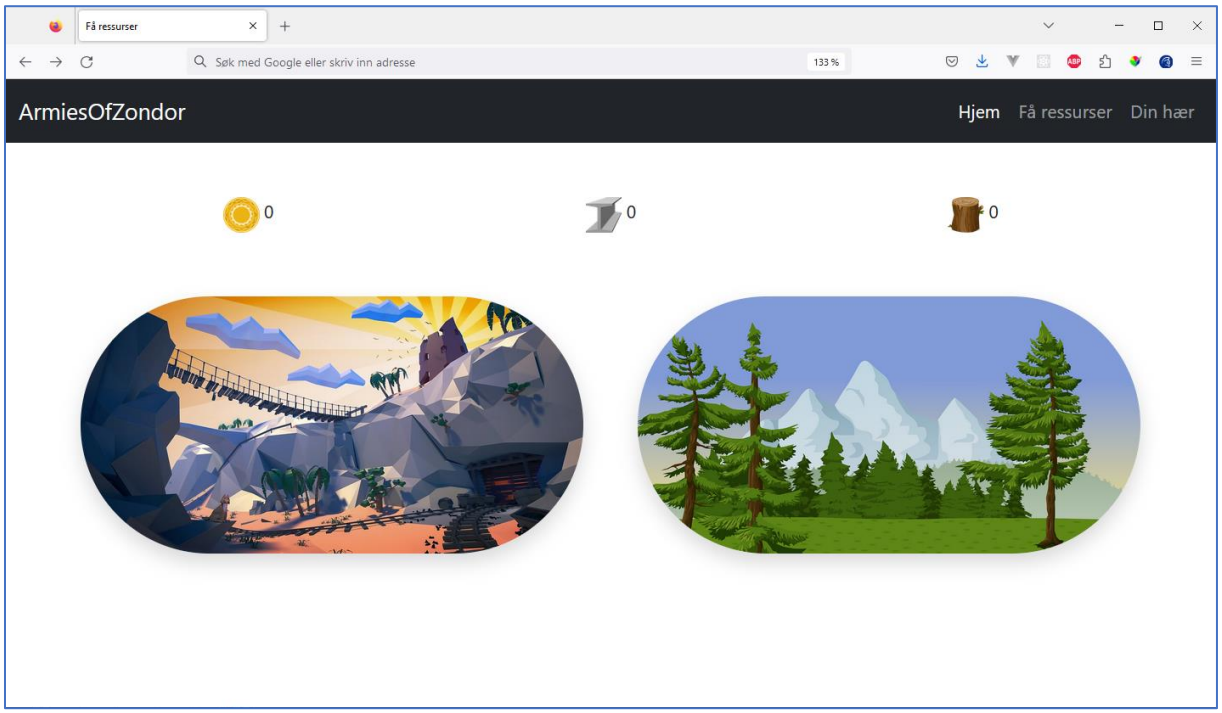
Page 1. Resource shop.

Page where you buy army resources (warriors, animals, and war machines).



- The warriors should have the upper part of the page, while the animals and war machines have the lower part. They are all shown with category name (for example a “Snake” type warrior, image, and price.
- You should make 2 (or 3) separate Modules for the information:
  - Warriors
  - Other (animals and war machines)
- Each warrior to buy has (at least):
  - categoryName
  - priceGold
  - image
- A war machine costs gold, metal, and wood to buy.

Page 2. Getting resources.

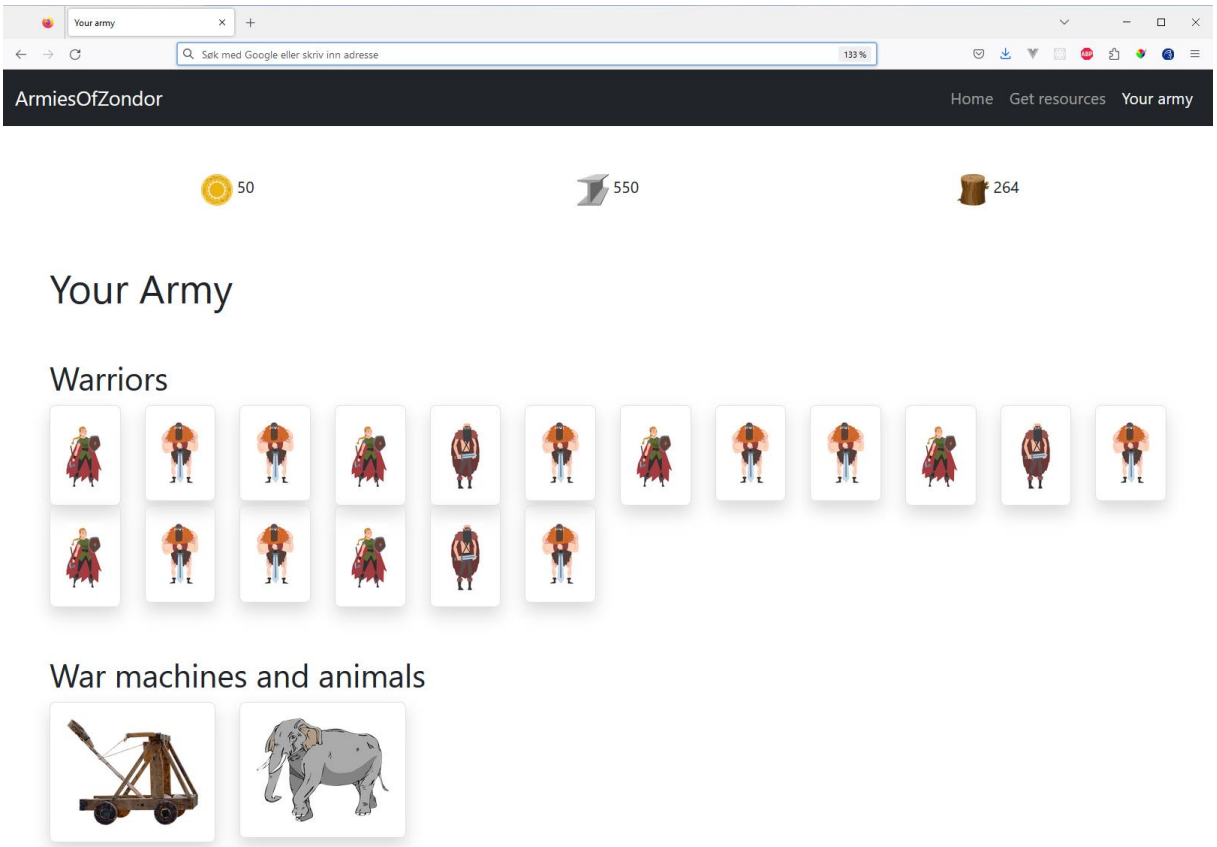


Page where you have to cut wood and mine metal and gold.

- When you click on the mine picture you will get a random amount of either gold or metal. There is a 75% chance of getting metal; thus 25% chance of getting gold.
- When you click on the woods picture you will get a random amount of wood.
- The numbers beside the gold, metal and wood pictures get updated as you get more gold, metal and wood. These resources need to be saved in localStorage.
- When you hover over the mines you should get the cursor of pickaxe-cursor.png
- When you hover over the woods you should get the cursor of axe-cursor.png

Page 3. Your army

Page where you see your bought army resources.



Other requirements and specification (for the entire solution)

- When something is bought it will have to be stored to localStorage in some way so that it can be shown in the "your army" page.
- Note all warriors, war machines, and animals on the pages are generated with JS!
- The solution is to be responsive
- All 3 pages will need access to localStorage regarding how much gold, metal, and wood you have.
- You are expected to add as much as possible different CSS properties (manually or with CSS framework) to get as much experience with coding CSS / CSS framework
- Should make use of favicon on the pages.
- Add search functionality with filter()

Additional things (not mandatory – ikke obligatorisk å ta med)

- Add functionality (on a new page for example) for summing up how much you have of each thing, how much you have spent on warriors, animals, war machines, etc.
- Other things you think may be nice/fun in the application.
- Write 200-300 words about Usability and Universal Design in context of the solution.

Collaboration task (mandatory)

- Give part of your code to 1-2 other groups in your class, and get the code from 1-2 other groups. Then write and 3-5 points about what is good, and 3-5 points about what can be improved to the other group(s) – see template below.
- "Part of your code" can for example be "gathering-army.html", "gathering-army.js", and "WarriorModule.js".
- You will deliver with your solution both the feedback you have given, and the feedback you have gotten.
- This task should/can be done in one of the lab exercises before the delivery date.

Template for feedback

3-5 good things in the solution code

1. X
2. Y
3. Z
4. A
5. B

3-5 things that can be improved, and/or tips etc.

1. X
2. Y
3. Z
4. A
5. B

Examples of things that one can give feedback on

- Variable, and function names
- Tidyness
- Structure
- Code repetition
- Code improvements
- …other things