Project #1 Guidelines: The Game

PROJECT

Generate a quiz about this unit | Summarize this unit | Have any doubt about this content? Ask!

Introduction

Congratulations on your progress in Module 1! You have been learning and applying different programming concepts, and you're well on your way to completing the module.

For your Module 1 project, you will create a browser-based game using HTML, CSS, JavaScript, DOM manipulation, and Object-Oriented Programming (OOP).

The purpose of this project is to consolidate and practice all the knowledge you have acquired in this module. You may choose any game idea you like, but it must meet the technical requirements in this lesson.

Examples

Below are examples of the past projects created by the students:

- Kitty Clicker
- Paper Droid
- Manny-Ski
- Oprah Game
- Changing Keys
- Galactic Wars
- The Interview

Technical Requirements

Your game must meet the following technical requirements:

1. **Render a game in the browser**.
2. **Have logic for winning and/or losing** and show feedback to the player in either case:

   Your game must have logic that allows the player to win or lose.
3. Your game code must be organized in **separate files for HTML, CSS, and JavaScript**.
4. Use **plain JavaScript** for DOM manipulation.
5. Your game entities and elements must be organized using **classes** and **OOP**.
6. **Have a repo on GitHub**.
7. Have at least **one (1) commit per day** that you worked on.
8. You must **deploy your game online using GitHub Pages** so anyone can play it.
9. Your code should follow the principles of **KISS (Keep It Simple Stupid)** and **DRY (Don't Repeat Yourself)**.

Deliverables

- A **working game, built by you**, that runs in the browser.
- Your game **deployed** on GitHub Pages **and available online**.
- The URL of the **deployed game available online**.
- The URL of the **GitHub repository** for your game.
- The URL of the **slides for your project presentation**.
- You must **present your game during Module 1 project presentations** (last day of Module 1 project).

Rubric

To assess your project and ensure all requirements are met, a **rubric** will be used. This rubric is used to **evaluate your project** by your teaching staff and to **communicate** what constitutes incomplete, acceptable and excellent performance across each learning outcome for the final project. Take some time to review the rubric and ask your lead teacher any questions about it if necessary.

Outcomes	0 - Incomplete	1 - Fair	2 - Good	3 - Excellent
Develop a browser game using HTML, CSS, JavaScript, DOM manipulation, and browser events	Student didn't implement any game views, game logic or user interactions, or only partially implemented some of them without making them fully functional.	Student created a basic game including, at a minimum, a game screen with basic logic and interactions, enabling the user to play the game.	Student implemented a fully functional game with win/lose conditions and ability to restart the game including the following game views: - start game screen, - game screen, - game over screen.	Student implemented all criteria in "Good", with the addition of at least one aditional feature, such as high score tracking, level progression, or other that demonstrated a deeper understanding of the material.
Organize application data and behavior using classes and OOP	Student did not use classes or OOP principles to organize their code.	Student used classes to organize and encapsulate game entities and elements.	Student used classes to organize and encapsulate game entities and elements. Student effectively used sub-classes (extends) to organizate the code and achieve inheritance of properties and methods from parent classes.	Student implemented all criteria in "Fair" and "Good", with the addition of researching and implementing one extra feature from the following: Unit Tests, Typescript, or private class fields (#).
Organize and abstract application logic using functions	Student has not made any effort to organize and abstract application logic using functions. The code is unstructured and lacks organization.	The code is partially organized using files and classes, but contains functions and methods that are considered too large or perform multiple tasks.	The code is broken down into functions or methods with clear responsibilities. However, there are still functions and methods that could be broken down and made smaller with clearer separation of concerns.	The entirety of the code is modular and broken down into functions or methods with clear responsibilities and minimal coupling.
Write clean code following best practices	The code indentation, spacing, syntax and variable and function names are inconsistent and don't follow best practices. Overall, the code is difficult to read and understand.	The code is readable, but there are still areas where formatting (indentation, spacing, syntax) or variable and function names could be improved to better follow best practices.	The code has consistent formatting. Variables and functions are named well and adhering to best practices. The code is readable and easy to understand, with some minor areas of improvement.	The code is exceptionally well-formatted, with clear and descriptive variable and function names that convey purpose and usage. The code is easy to read and understand, with an exemplary level of clarity.
Layout and style web pages using and the game's UI using CSS	Student has not made any effort to layout and style the web pages or the game's user interface using CSS.	Student partially styled the web pages and the game's UI using CSS.	Student adequately styled all the web pages and the game's UI using CSS. Student made an obvious effort to create a pleasant and appealing user interface.	Student implemented all criteria in "Good", with the addition of researching and implementing an extra feature such as animated sprites or CSS animations and transitions. The end result is a highly polished and professional-looking game UI.
Version Control	Student has not made any effort to save and track changes in the source code using Git and Github.	Student has created a GitHub repository and have made commits to track changes in the source code.	Student has created a GitHub repository and had made regular commits to track changes in the source code. The commit messages are only partially clear and could be improved for clarity.	Student has created a GitHub repository and had made regular commits to track changes in the source code. The commit messages are clear and informative, making it easy to understand the changes that were made.
Client-side development	Student has not made any effort to deploy the game using Github Pages or a similar solution.	Student deployed the game using Github Pages or a similar solution, but some or all of the deployed game features are not working.	Student has deployed the game using Github Pages or a similar solution. However, there may be some minor issues with the functionality of the game when deployed that need to be addressed.	Student has successfully deployed the game using Github Pages or a similar solution. The deployed game is full functional with no issues or errors present.
Career	Student did not make time or effort to communicate with others during standups or other activities. Student depended on others	Student communicated at times with others during standups but could be more clear and concise in their delivery. Student remained in their comfort zone by	Student demonstrated clear communication skills and attended all standups. Student adhered to a level of professionalism by	Student impressively demonstrated clear communication skills during standups and contributed to groups discussions, going out of their way to help other students in the group

Career Development	(including LTs and TAs) to do the heavy thinking and most of the work. Student did not make the effort to assist others or collaborate in any way.	excluding themselves from conversations with classmates, LTs and TAs that would require team-work skills and idea exchange during the development of the project.	acting emotionally mature, and engaged in team-work skills by listening to others' (including LTs and TAs) ideas while contributing their own.	students in the group). Student adhered to a level of professionalism by acting emotionally mature, and engaged in team-work skills by listening to others' (including LTs & TAs) ideas while contributing their own.
Career Development	Student did not present or demo their application or talk through their slides during the project presentations.	Student presented and demoed their work but struggled with the timing and the rhythm of the presentation. Student had to skip or rush through certain slides to keep up with the time. Student struggled delivering a clear message, leaving the audience confused.	Student presented their work effectively. The presentation and demo appeared prepared and structured, conveying a clear message to the audience. Student used the allocated time effectively.	Student presented their work impressively. The presentation and demo appeared very well prepared and structured, conveying a clear message to the audience. Student used the allocated time effectively. Student's tone was well aligned with the subject of the project. Student kept the audience in awe for the duration of the presentation and demo.

Schedule

Full-time:

- **Stand ups***
- **Coding Kata**
- **Project development**

Part-time:

- **Week Nights**
  - **Stand ups***
  - **Project development**
- **Saturdays**
  - **Stand ups***
  - **Coding Kata**
  - **Project development**

** You will start with this activity every day during project weeks. Take a minute to update your colleagues with your progress: what did you do so far, do you feel on the track, behind or ahead, what's your plan for today and do you think you'll have any blockers.*

Project Presentation day

On the last day of the module, you will have the opportunity to present your project to the staff and your fellow students. This is your chance to showcase what you've learned and demonstrate the skills you've developed throughout the module. Remember to dress at least one degree nicer than usual. For tips on how to prepare and deliver an effective presentation, see the Demo Tips section.

Presentations

For each of the 3 projects you make at Ironhack, you will also have to make a presentation about it. Communication (including public speaking) is an important skill to practice for finding a job after Ironhack.

Format

- Talking with Slides: **3 minutes**
- Demo: **2 minutes**
- Total: **5 minutes**

Attire

- Dress nicely for this and all final project presentations (last day of each project's time).
- Dress at least **one degree more elegantly** than you usually dress for class.
- *Examples*:

- If you wear T-shirts every day, wear a button-down shirt.
- If you wear jeans every day, wear some slacks.
- If you wear sneakers every day, wear nicer shoes.

Online Presentation Tools

- For your project presentation, you **must use an online presentation application** to create your presentation slides.
- PowerPoint files, Keynote files or files of any kind **will not be accepted**.
- We recommend using one of the following online presentation applications:
  - Slides
  - Prezi
  - Google Slides

Presentation Structure

Your presentation should consist of 7-10 slides that cover the following topics:

1. **Title Slide** (1 slide): your project's name & your name
2. **About Me** (1-2 slides):
   - Where are you from?
   - What are some interesting facts about you? (hobbies, travels, etc.)
3. **Project Elevator Pitch** (1-2 slides):
   - What is your project?
   - How does it work?
   - Why did you choose it?
4. **Technical Challenge** (1 slide):
   - What was **the most important** technical challenge you faced?
   - How did you overcome that challenge?
5. **Learnings** (1 slide):
   - What was **the biggest** mistake you made during this project?
   - What did you learn from it?
6. **Demo Slide** (1 slide): Include a link to your project to open it for the demo easily.
7. **Closing Slide** (1 slide): your project's name, your name & a *"Thank You"*

Demo Tips

1. Plan what you are going to demo and **practice it on the live site**. That way, you won't be surprised if something breaks on the live version.
2. **Deploy early** so you can squash bugs. There are *always* bugs on the live site at first.
3. Add a **link to your live project** to your DEMO slide so you can start it smoothly.
4. Your app's colors and sizing **might look different on the projector**. If you think it might be a problem, ask to test it beforehand.

Mark as completed

PREVIOUS
← LAB | DOM Race Car

NEXT
JS OOP (202402) →