Globant Piscine

2048

Summary: Merge your way to glory! A simple grid, some JS, and the endless pursuit of 2048.

Version: 2

Contents

Chapter I

A word about this Project

Game development is an exciting avenue of coding because of the creative challenges and the opportunities it offers to build interactive experiences from the ground up. In this project, we'll focus on using pure JavaScript and CSS to recreate the classic puzzle game 2048, emphasizing simplicity and foundational skills in web development.

Here, the focus shifts to implementing game mechanics and UI design without external libraries, ensuring a deeper understanding of how core web technologies can work together to deliver a dynamic and engaging user experience.

Chapter II

Introduction

What this Project will show you:

- Implementation of game logic using pure JavaScript.
- Creation of a dynamic and responsive grid-based UI with CSS.
- Handling real-time updates and user interactions without external libraries.
- Developing modular, maintainable, and reusable code.
- Exploring game mechanics like tile merging and win/loss detection.
- Building a foundational understanding of web development principles.

Chapter III

General instructions

Unless explicitely specified, the following rules will apply for every project of this Piscine.

- This subject is the one and only trustable source. Don't trust any rumor.
- This subject can be updated up to one hour before the turn-in deadline.
- The assignments in a subject must be done in the given order. Later assignments won't be rated unless all the previous ones are perfectly executed.
- Be careful about the access rights of your files and folders.
- Your assignments WON'T be evaluated by your Piscine peers.
- You must not leave in your turn-in your workspace any file other than the ones explicitly requested By the assignments. If the assignment don't precise them, put only the necessary ones to run your Project.
- Using some API Key or Token? Keep them for you! Do not push them on your repository.
- You have a question? Ask your left neighbor. Otherwise, try your luck with your right neighbor.
- Every technical answer you might need is available in the `man` or on the Internet.
- You must read the examples thoroughly. They can reveal requirements that are not obvious in the assignment's description.
- By Loki, by Freya! Use your brain!!!

Chapter IV

Glossary

This glossary is designed to help you quickly understand the main technologies and concepts you will work with in **2048**.

- **HTML (HyperText Markup Language)**: The standard language used to structure the content of a webpage. It defines the elements you see in the browser, such as buttons, grids, or text...
- **CSS (Cascading Style Sheets)**: Language used to style and visually design web pages — controls colors, fonts, layout, and animations.
- **JavaScript (JS)**: Programming language that allows you to add logic and interactivity to your webpage.
- **DOM (Document Object Model)**: A tree-like structure representing the HTML of a page. You'll use JavaScript to select, create, and modify DOM elements.
- **Event Listener**: A JavaScript function that "listens" for user actions like pressing arrow keys or clicking the restart button, triggering corresponding game logic.
- **Animation**: Visual effect used to make transitions smooth. Can be implemented using CSS transitions or JavaScript.

Chapter V

Mandatory part

Globant	Exercise 00
2048	
Turn-in directory : $ex00/$	
Files to turn in : `All needed files to run your Project and nothing else`	
Allowed functions : `None`	

- You must include a **README.md** file in your repository explaining briefly the project and how to run it.
- The game is played on a **4x4 grid**, where each cell can contain a tile with a value that is a power of 2 (e.g., 2, 4, 8, up to 2048).
- **Initial Setup**: The game starts with two tiles randomly placed on the grid, each with a value of either 2 or 4.
- **User Input**: Players use the keyboard arrow keys to move tiles in the following directions:
  - **Up Arrow**: Move tiles up.
  - **Down Arrow**: Move tiles down.
  - **Left Arrow**: Move tiles left.
  - **Right Arrow**: Move tiles right.
- **Tile Movement**: Tiles slide as far as possible in the chosen direction until they hit the edge of the grid or another tile.
- **Merging Rules**:
  - When two tiles with the same value collide, they merge into one tile with a value equal to their sum.
  - Merging increases the player's score by the value of the merged tile.

- Merging happens only once per move per tile.

- **Score Tracking**: The player's current score is displayed and updates with every merge.
- **End Game Condition**:
  - The game ends when the grid is full and no valid moves are possible (i.e., no adjacent tiles can merge).
  - A "Game Over" message notifies the player, with an option to restart the game.
- **Winning Condition**:
  - Players win the game by creating a tile with the value of 2048.
- **Restart Functionality**: A "Restart" button allows players to reset the game to its initial state.
- **Visual Feedback**:
  - Provide animations for tile movement and merging.
  - Display messages for winning and game over scenarios using alerts or modals.

- **Possible HTML Boilerplate Starting Point**:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>2048 Game</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="game-container">
        <h1>2048</h1>
        <div class="score-container">
            <span>Score: <span id="score">0</span></span>
            <button id="restart-btn">Restart</button>
        </div>
        <div class="grid-container">
            <!-- The grid will be filled by JavaScript -->
        </div>
    </div>
    <script src="script.js"></script>
</body>
</html>
```

For this project you will only need pure JavaScript, HTML and CSS. No external libraries are allowed, unless specified.

You can still use some CSS frameworks to help you with the layout. You have the choice between Bootstrap, Tailwind, Bulma, Materialized or Pure.css.

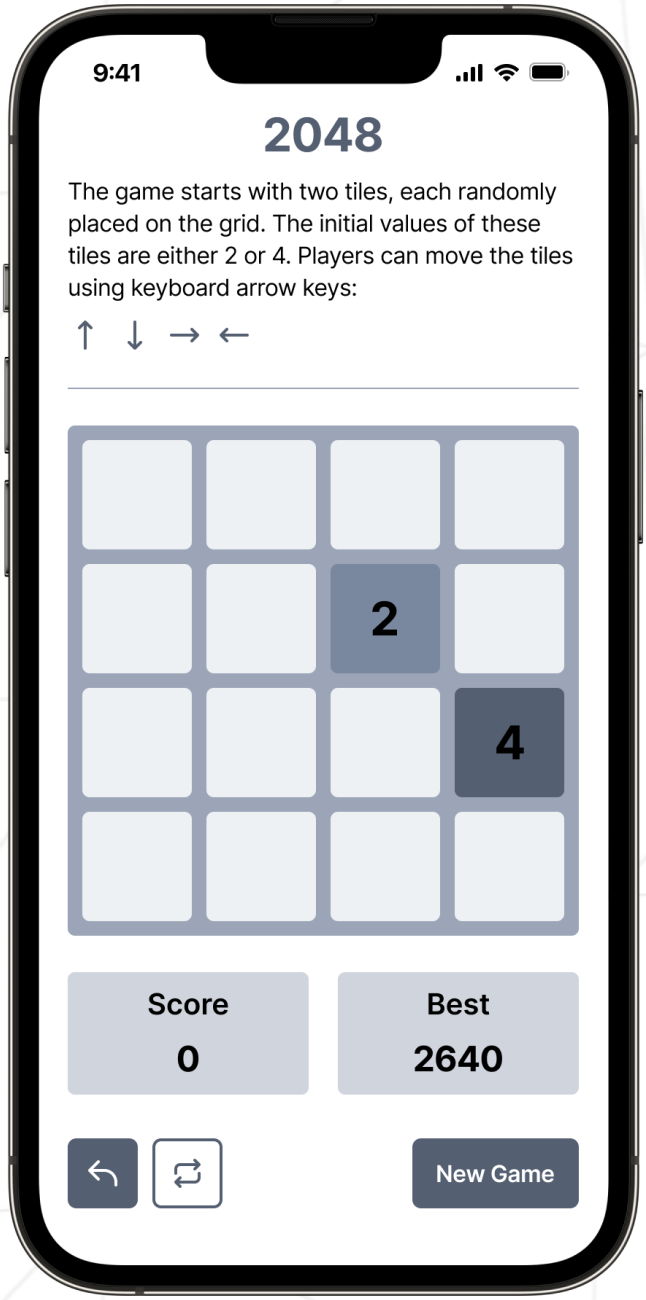

Figure V.1: Example of the main game interface

Chapter VI

Bonus Part

Once mandatory requirements are implemented in the application, you can add any additional features you see fit. The only limit is your imagination, but be respectful of your time.

Chapter VII

Submission

- Create a git repo (Github, Gitlab, Bitbucket, etc) and add your project files to it.
- Copy the link to your repository and paste it in the project submission form.
- Project submission form: TYPEFORM

Please note, no modifications made on the repo after the form is sent will be taken into account for the evaluation.

No Peer evaluation for this Piscine, but we strongly recommend reviewing and using the evaluation sheet we give you to check if your project meets all the requirements.