📄 Project Documentation – Retro Games

🎯 Project Goal

The goal of the project is to create a browser-based retro gaming platform featuring classic games. The main focus is on the game Tetris, fully implemented using HTML5, CSS3, and modern JavaScript (ES6 modules, Canvas API, etc.). The website is playable offline using Service Workers and supports dynamic navigation with the History API.

🛠️ Development Process

Structure Design:

Defined the overall page layout using semantic HTML5 tags (<main>, <section>, <header>, <footer>). Each game has its own structure and canvas area.

Tetris Implementation:

Split game logic into multiple ES6 modules: Arena, Player, Game, TetrisRenderer, Utils, etc. Used Canvas API to render the game board and blocks. Implemented keyboard controls (WASD / arrow keys). Music playback integrated using <audio> and Media API.

Progressive Features:

- Offline support through Service Worker, caching all necessary files.
- Dynamic JavaScript loading depending on selected game.
- Used the History API to support seamless navigation and back button functionality.
- jQuery used for simpler DOM interaction.
- Responsive Design with CSS3:
    - Used media queries to support mobile devices.
    - Included animations, pseudo-classes, and CSS transitions for visual polish.

⚙️ Functionality Overview

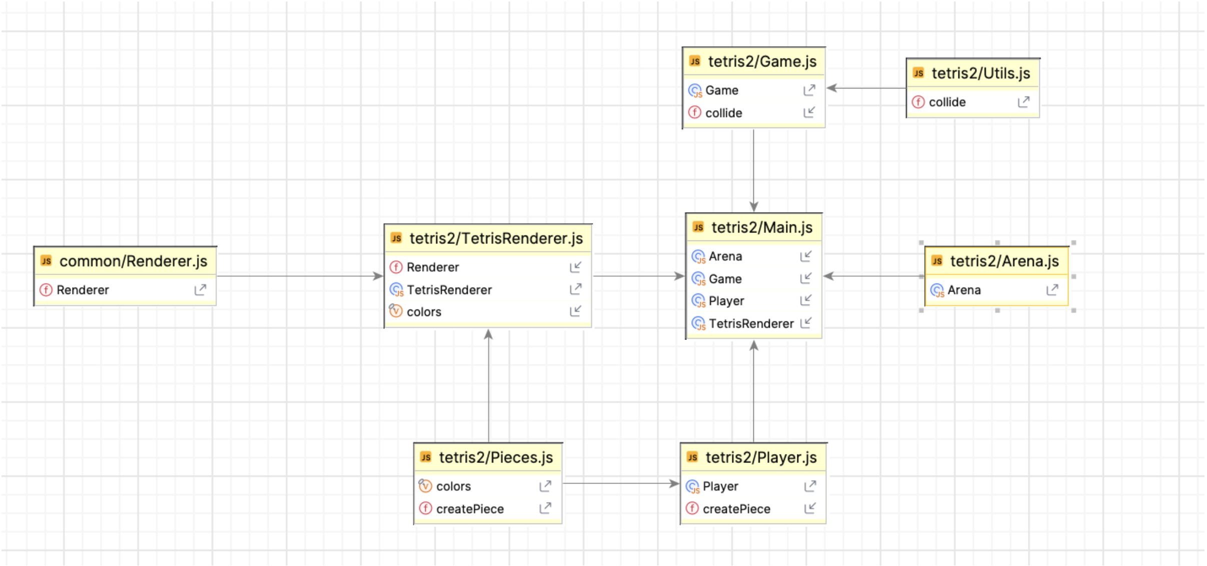
- **Home page (index.html):**
    - Shows a selection of retro games.
    - Buttons for each game: available ones load the game, others show "SOON".
- **Tetris:**
    - Real-time block falling and collision detection.
    - Score, level, and clear line tracking.
    - Pause, restart, and music toggle controls.
    - "Next block" preview shown on side panel.
- **Browser History API:**
    - Clicking "PLAY" updates the URL (e.g., tetris.html) without full page reload.
    - Pressing the browser back button returns to the home screen dynamically.
- **Offline Support:**
    - Once loaded, the entire site works offline via Service Worker.
    - Includes fallback mechanism for missing pages.
- **UI Interaction:**
    - Button hover/disabled states.
    - "Coming Soon" games are non-playable and display a preview message.

💭 Code Comments and Documentation

The code is commented with explanations for:

- Class and module purpose (e.g., Game, Player, TetrisRenderer)
- Method descriptions and parameters
- Complex logic like collision detection, block rotation, etc.

Object Oriented Design of Tetris


tetris2/Game.js
Game
collide
tetris2/Utils.js
collide
common/Renderer.js
Renderer
tetris2/TetrisRenderer.js
Renderer
TetrisRenderer
colors
tetris2/Main.js
Arena
Game
Player
TetrisRenderer
tetris2/Arena.js
Arena
tetris2/Pieces.js
colors
createPiece
tetris2/Player.js
Player
createPiece