

# Project Documentation: Canvas Platformer Game

## Overview:

This project is a simple platformer game implemented using JavaScript and the HTML5 <canvas> element. The game features a player character that can move left, right, and jump on various platforms. The background and platforms are designed to create a scrolling effect as the player navigates the game world.

## Features

- **Player Movement:** The player can move left, right, and jump using the A, D, and W keys, respectively.
- **Scrolling World:** The game world scrolls as the player moves, creating an immersive experience.
- **Platform Collision:** The player can land on platforms and stand on them.
- **Background Elements:** The game includes background images to enhance the visual appeal.
- **Win and Lose Conditions:** The game checks for win conditions when the player reaches a certain point and for lose conditions when the player falls off the screen.

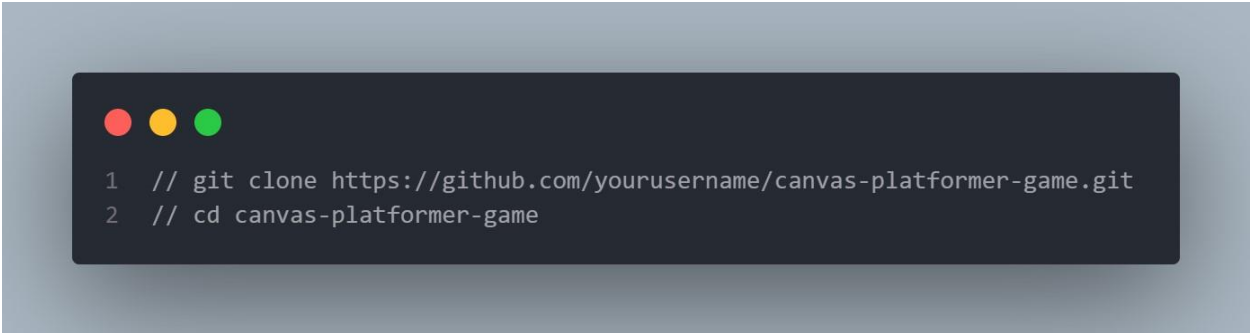
## File Structure

- `index.html`: The main HTML file that includes the canvas element.
- `style.css`: The CSS file for styling the game (if needed).
- `main.js`: The JavaScript file containing the game logic and classes.
- `img/`: Directory containing image assets for the game.

## Installation and Setup

To run this project locally, you need to have Node.js and npm installed. Follow the steps below to set up and run the game:

### 1. Clone the Repository

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. It contains two lines of text: a comment followed by a git clone command, and a comment followed by a cd command.

```
1 // git clone https://github.com/yourusername/canvas-platformer-game.git
2 // cd canvas-platformer-game
```

### 2. Install Dependencies

This project uses a simple server to serve the files. You can use http-server for this purpose

```
npm install http-server -g
```

### **3. Run the Server**

Navigate to the project directory and start the server:

```
http-server
```

### **4. Open the Game in Browser**

Open your browser and navigate to <http://localhost:8080> (or another port if specified by http-server).

# Game Logic

## Classes

### Player

Properties:

- **speed**: Movement speed of the player.
- **position**: Current position of the player.
- **velocity**: Current velocity of the player.
- **width, height**: Dimensions of the player.
- **image**: Current sprite image of the player.
- **frames**: Frame counter for sprite animation.
- **sprites**: Object containing all sprite states and their properties.
- **currentSprite**: Currently active sprite.
- **currentCropWidth**: Width of the current sprite's frame.

## **Methods:**

- `draw ()`: Draws the player sprite on the canvas.
- `update ()`: Updates the player's position, velocity, and handles animation frames.

## **Platform**

### **Properties:**

- `position`: Position of the platform.
- `image`: Image of the platform.
- `width, height`: Dimensions of the platform.

### **Methods:**

- `draw ()`: Draws the platform on the canvas.

## **GenericObject:**

### Properties:

- **position:** Position of the object.
- **image:** Image of the object.
- **width, height:** Dimensions of the object.

### **Methods**

- **draw ():** Draws the object on the canvas.

## **Functions**

- **createImage(imageSrc):** Creates and returns a new Image object with the specified source.
- **Init ():** Initializes the game state, including player position, platforms, and background objects.
- **animate ():** The main animation loop that handles drawing and updating all game elements. It uses requestAnimationFrame for smooth animation.



## **Event Listeners:**

- **keydown:** Listens for keydown events to control the player's movement and jumping.
- **keyup:** Listens for keyup events to stop the player's movement when keys are released.

## **Game Mechanics:**

- **Gravity:** A constant force that pulls the player down, simulating gravity.
- **Collision Detection:** Checks for collisions between the player and platforms to handle landing and standing.
- **Scrolling:** Adjusts the position of platforms and background elements to create a scrolling effect when the player moves.

## **Future Improvements:**

- **Enhanced Collision Detection:** Improve collision handling for more complex interactions.
- **More Levels:** Add more levels with different platform arrangements and challenges.
- **Sound Effects:** Integrate sound effects for player actions and events.
- **Responsive Design:** Make the game responsive to different screen sizes and orientations.

## **Conclusion**

This project is a foundational example of a platformer game using the HTML5 `<canvas>` element and JavaScript. It demonstrates basic game development concepts such as sprite animation, collision detection, and game loops. The structure allows for easy expansion and improvement, making it a great starting point for more advanced game development projects.