Cheat Sheet: Using ES6 Classes in HTML5 Game Development

What is a Class?

A class is a blueprint for creating objects that represent things in your game. It bundles together data (like position, speed) and behavior (like move, draw).

Five Rules for Using Classes in Games

1. If it exists in the game world, make it a class.

- Examples: `Ship`, `Asteroid`, `Bullet`, `Enemy`, `Paddle`, `Ball`

2. If it controls or manages the game, it should be a class too.

- Common example: `Game` class
- Manages game loop, object creation, and interactions

3. Group behavior inside the class that owns it.

- The ship should know how to rotate and thrust
- The asteroid should know how to drift
- The game class checks for collisions

4. Keep classes focused on their job.

- Avoid making one class do everything
- Example: Ship shouldn't check for collisions with asteroids—Game should

5. Systems like input or sound can also be classes.

- `InputHandler`, `SoundManager`, `ScoreBoard` are great class candidates

Who Does What?

Class	Responsibility

`Ship`	Moves, rotates, shoots
`Bullet`	Moves forward, has limited life
`Asteroid`	Drifts, gets destroyed by bullets
`Game`	Manages state, loop, collisions, drawing
`InputHandler`	Tracks keys pressed
`SoundManager`	Plays sounds, music

Analogy: A Game is Like a School Play

- **Actors (Ship, Asteroids)** = Classes that appear on screen
- **Director (Game class)** = Runs the show
- **Stage crew (Input, Sound)** = Handle behind-the-scenes support

Quick Tips for Students

If...	Then...
It appears/moves in the game	Make it a class
It manages or controls other objects	Make it a class
It just does one simple task	Maybe just use a function
It’s hard to describe what it does	Rethink its responsibilities

Class Design Makes Games Easier to Build, Read, and Extend!

- Clear responsibilities = easier debugging
- Reusable components = faster development
- Good structure = more fun to make and share!