**01-overview.md**

text

---

title: "System Overview"

tags: [overview, goals, interactive app, web]

priority: true

language: en

---

# System Overview

This guide explains the main goals when building a system for web-based games or interactive apps—what you can create, who it's for, and how the different parts fit together.

- Build interactive educational, entertainment, or productivity apps.

- Focus on responsive, dynamic user experiences.

// Example: create a simple interactive button  
document.body.innerHTML = "<button>Start Game</button>";

text

undefined

**02-stack-and-tools.md**

text

---

title: "Choosing Technologies and Tools"

tags: [react, javascript, canvas, webpack, vite]

priority: true

language: en

---

# Choosing Technologies and Tools

A quick guide to modern web dev stacks: vanilla JS, React, Canvas, and build tools (Webpack/Vite). These tools make it easy to develop, test, and deploy interactive projects.

- Try React for component-based UI.

- Use the Canvas API for fast 2D rendering.

- Webpack or Vite streamline local development.

// Example: React component rendering a game canvas  
function GameCanvas() {  
return <canvas width={300} height={300}></canvas>;  
}

text

undefined

**03-architecture-ui-patterns.md**

text

---

title: "UI Structure and Architecture Patterns"

tags: [UI, components, state, architecture]

priority: true

language: en

---

# UI Structure and Architecture Patterns

How to split your code into manageable pieces: UI components, state management, and separating business/game logic from rendering.

// Example: Managing state in a React component  
const [gameState, setGameState] = useState("ready");

text

undefined

**04-game-loop-and-timing.md**

text

---

title: "Game Loop and Timing"

tags: [game loop, raf, delta-time, timing]

priority: true

language: en

---

# Game Loop and Timing

Explains the core of interactive games—the update-render loop. Use requestAnimationFrame for smooth animation, and calculate delta-time for consistent movement.

function gameLoop(now) {  
update();  
render();  
requestAnimationFrame(gameLoop);  
}  
requestAnimationFrame(gameLoop);

text

undefined

**05-canvas-and-rendering.md**

text

---

title: "Canvas and Rendering"

tags: [canvas, drawImage, sprites, retina]

priority: true

language: en

---

# Canvas and Rendering

Covers drawing and rendering sprites on the HTML5 Canvas, with tips for HiDPI ("retina") displays.

// Example: Drawing a sprite  
ctx.drawImage(sprite, x, y, width, height);

text

undefined

**06-physics-and-collision.md**

text

---

title: "Physics and Collision Detection"

tags: [collision, AABB, circle collision, physics]

priority: medium

language: en

---

# Physics and Collision Detection

Simple approaches for collision detection: axis-aligned bounding box (AABB) and circular collision checks.

// Example: AABB collision check  
function isColliding(rect1, rect2) {  
return (  
rect1.x < rect2.x + rect2.width &&  
rect1.x + rect1.width > rect2.x &&  
rect1.y < rect2.y + rect2.height &&  
rect1.y + rect1.height > rect2.y  
);  
}

text

undefined

**07-input-and-controls.md**

text

---

title: "Input and Controls"

tags: [input, keyboard, touch, gamepad]

priority: true

language: en

---

# Input and Controls

Covers handling controls from keyboard, mouse, touch, and basic gamepads, including code for de-bouncing and mobile compatibility.

// Keyboard input example  
window.addEventListener('keydown', (e) => {  
if (e.code === 'ArrowLeft') {  
// move left  
}  
});

text

undefined

**08-asset-management-and-loading.md**

text

---

title: "Asset Management and Loading"

tags: [assets, loader, audio, caching]

priority: medium

language: en

---

# Asset Management and Loading

How to preload images, sounds, and other assets; manage caching; and use CDNs for speed.

// Image preloading  
let img = new Image();  
img.src = 'sprite.png';  
img.onload = () => { /\* ready to render \*/ };

text

undefined

**09-audio-and-sound-effects.md**

text

---

title: "Audio and Sound Effects"

tags: [webaudio, sound, sfx, music]

priority: low-medium

language: en

---

# Audio and Sound Effects

Basics of using the Web Audio API for playback of music and sound effects.

// Web Audio API example  
const ctx = new (window.AudioContext || window.webkitAudioContext)();  
const audio = new Audio('effect.mp3');  
const source = ctx.createMediaElementSource(audio);  
source.connect(ctx.destination);  
audio.play();

text

undefined

**10-performance-and-profiling.md**

text

---

title: "Performance and Profiling"

tags: [performance, fps, memory, profiling]

priority: true

language: en

---

# Performance and Profiling

Learn how to use browser devtools for FPS tracking, memory leaks, and how to batch draw calls for canvas performance.

Open Chrome DevTools → Performance tab to record, analyze slow frames, and spot memory problems.

text

undefined

**11-testing-and-debugging.md**

text

---

title: "Testing and Debugging"

tags: [test, debug, playwright, jest]

priority: medium

language: en

---

# Testing and Debugging

How to write unit and end-to-end tests and use browser debugging tools. Recommended libraries: Jest (unit), Playwright (e2e).

// Simple Jest unit test example  
test('moves player left', () => {  
expect(movePlayer('left')).toBeLessThan(startPos);  
});

text

undefined

**12-tutorial-tetris.md**

text

---

title: "Tetris Game Tutorial"

tags: [tetris, game, rotation, lines, canvas]

priority: true

language: en

---

# Tetris Game Tutorial

Step-by-step guide for building Tetris with JavaScript and Canvas—from shape logic to rotation and scoring.

// Example: Rotate tetromino matrix  
function rotate(matrix) {  
return matrix.map((\_, i) => matrix.map(row => row[i])).reverse();  
}

text

undefined

**13-example-react-game.md**

text

---

title: "React Mini Game Project"

tags: [react, hooks, canvas, example]

priority: true

language: en

---

# Example: React Mini Game Project

Includes a sample project combining React components and Canvas for rendering game logic.

import React, { useRef, useEffect } from "react";  
function Game() {  
const ref = useRef();  
useEffect(() => {  
const ctx = ref.current.getContext('2d');  
// Game rendering here  
}, []);  
return <canvas ref={ref} width={300} height={300}/>;  
}

text

undefined

**14-code-snippets.md**

text

---

title: "Code Snippet Collection"

tags: [snippet, example, code, runnable]

priority: true

language: en

---

# Code Snippets

Ready-to-use snippets: game loop, collision detection, input handlers, preloaders. Each block is standalone and ready to run.

// Game loop pattern  
function loop() { update(); draw(); requestAnimationFrame(loop); }

text

undefined

**15-fewshot-examples.json**

json

{

"examples": [

{"prompt": "Move character right", "output": "player.x += 5;"},

{"prompt": "Test collision", "output": "isColliding(a, b)"}

*// ...add more prompt-output pairs here*

]

}

**16-security-and-deployment.md**

text

---

title: "Security and Deployment"

tags: [deploy, security, cors, csp]

priority: medium

language: en

---

# Security and Deployment

How to deploy on Netlify/Vercel, use CORS and CSP headers, and sanitize user input.

Set the following CSP: "default-src 'self'; script-src 'self'; object-src 'none';"

text

undefined

**17-accessibility-i18n.md**

text

---

title: "Accessibility and i18n"

tags: [a11y, rtl, i18n]

priority: medium

language: en

---

# Accessibility and Internationalization

Implement keyboard navigation (ARIA roles), support for RTL languages, and localized text.

<button aria-label="Start Game">Start</button>

text

undefined

**18-ocr-and-scan-guidelines.md**

text

---

title: "OCR and Scan Guidelines"

tags: [ocr, tesseract, scans]

priority: low-medium

language: en

---

# OCR and Scan Guidelines

Tips for using Tesseract.js to run OCR on scanned PDFs and improve text parsing results.

import Tesseract from 'tesseract.js';  
Tesseract.recognize('scan.png').then(({ data }) => console.log(data.text));

text

undefined