

Neon Dodger: Browser Arcade Game

A neon arcade game where you steer a ship between falling asteroids. It speeds up over time and keeps your high score. It runs in any browser, including phones.

How to use this kit: read the spec, set up the stack, then paste the prompts into your AI coding tool one at a time, in order. Test after every prompt before moving on.

1. One-page spec

Problem

Wants a quick, shareable game to play with friends without installing anything.

Target user

Casual players on desktop and mobile, shared via a link on Discord.

Features

- Player ship with keyboard and touch control
- Falling asteroids that speed up
- Score over time
- Collision and game over
- Local high score
- Neon glow visuals and simple sounds

Screens

Screen	Purpose
Title	Game name, Play button, high score.
Game	Canvas play area with score HUD.
Game over	Final score, best score, Play again.

MVP: build this first

- ✓ One endless level
- ✓ Keyboard + touch controls
- ✓ Score, game over, restart
- ✓ localStorage high score

Later, not in v1

- Online leaderboard
- Power-ups
- Multiple ships/skins
- Music track

2. Beginner-friendly tech stack

Layer	Choice	Why
Rendering	HTML5 Canvas + vanilla JS	Built into every browser, so there's no engine to learn.
Game loop	requestAnimationFrame with delta time	Smooth movement at the same speed on any screen.
Save	localStorage	Keeps the high score with no server.
Hosting	Netlify, Cloudflare Pages, or itch.io (free)	Shareable link in minutes.

Setup before prompt 1

1. Install Cursor.
2. Create a folder called neon-dodger and open it in Cursor.
3. Use Live Server or any local preview to run index.html.

3. Build prompts (13 steps, in order)

Copy each block into your AI tool's chat/agent panel. Accept the changes, run the test, then continue.

Prompt 1: Canvas skeleton

Create `index.html`, `style.css`, `game.js`. `index.html` has a full-screen `<canvas id="game">`, black background, no scrollbars. `game.js` resizes the canvas to the window (handle `devicePixelRatio`). No game logic yet.

Test it: See a black full-screen canvas that resizes.

Prompt 2: Game loop

Add a `requestAnimationFrame` loop with `update(dt)` and `draw()` functions, where `dt` is in seconds and capped at `0.05`. Draw an FPS counter in the corner for now.

Test it: FPS counter shows around 60.

Prompt 3: Player ship

Add a player ship at the bottom of the canvas. It moves left and right with the arrow keys and A/D, and follows the finger on touch screens. Keep it inside the canvas edges. Don't add enemies yet.

Test it: Move with keys and touch; it can't leave the screen.

Prompt 4: Asteroids

Spawn asteroids at random x positions above the screen that fall down. Store them in an array, remove them once they're off-screen. Spawn rate is a variable.

Test it: Asteroids rain down; check that the array length stays small.

Prompt 5: Collision

Add circle-vs-circle collision between the ship and asteroids, using slightly forgiving hitboxes. On hit, set state to `'gameover'`.

Test it: Crash into a rock and the game stops.

Prompt 6: Game states

Add states: `'title'`, `'playing'`, `'gameover'`. Title shows `'NEON DODGER'` and `'Tap or press Space'`. Game over shows score and `'Play again'`. Reset everything properly on restart.

Test it: Play, die, restart 3 times with no leftover asteroids.

Prompt 7: Score and difficulty

Score increases with time survived. Every 10 seconds, increase fall speed and spawn rate a little, with a sensible cap.

Test it: It feels harder after 30 seconds.

Prompt 8: High score

Save the best score to `localStorage` key `'neondodger.best'`. Show it on the title and game-over screens, plus `'NEW BEST!'` when beaten.

Test it: Beat your score, reload, best is kept.

Prompt 9: Neon visuals

Make it neon: glowing strokes (`shadowBlur`), a cyan ship, magenta asteroids, a scrolling starfield background. Keep it fast on phones, and turn the glow off if FPS drops below 45.

Test it: Looks good and stays smooth on your phone.

Prompt 10: Juice

Add a small screen shake and particle burst on crash, plus a thruster flicker. Keep the effects short.

Test it: Crash feels punchy, not laggy.

Prompt 11: Sound

Add simple sound effects using the Web Audio API (no files): a blip on start, a boom on crash. Add a mute toggle (M key or an icon) saved to localStorage. Audio starts only after the first user input.

Test it: Sounds play; mute persists after reload.

Prompt 12: Mobile polish

Prevent page scroll/zoom on touch, add a pause when the tab is hidden, and remove the FPS counter unless ?debug is in the URL.

Test it: Play on phone; switching apps pauses.

Prompt 13: Deploy prep

Add title, meta description, Open Graph tags, and a favicon. Write a short README with controls. Confirm no console errors.

Test it: Upload to Netlify Drop or itch.io (HTML game) and share the link.

4. Common pitfalls

Speed depends on monitor refresh rate. Multiply all movement by dt.

Blurry canvas on phones. Scale the canvas by devicePixelRatio.

Restart bugs. Put all reset logic in one resetGame() function.

Audio blocked by the browser. Create the AudioContext only after the first click or tap.

Glow kills performance. shadowBlur is expensive, so limit it or turn it off on slow devices.

Feature creep. Ship the endless mode first; power-ups go in v2.

5. Launch checklist

- [] Playable with keyboard and touch
- [] Stable 60fps on a mid-range phone
- [] Restart works with no leftover state
- [] High score persists
- [] Mute works
- [] No console errors
- [] OG image/description so the link previews nicely
- [] Deployed and link tested in incognito
- [] Shared on Discord/X for feedback

Questions or problems with this kit? Email thetenderventure@gmail.com. Kit generated by AI agents from the Charging4ward template; double-check anything involving money, security, or personal data.