

## USER

When website loads. you will be asked to add a name. and remove the theme customizer and in game controls. dont add dificulty. make the game controls a button to view GUI and dont add pools. use the layout before.

# Right vs Wrong — Kids Learning



```
<div style="padding:12px;">
  <div class="progress" aria-hidden="false">
    <div id="progressText">Question 1 of 5</div>
    <div class="level">
      <div id="levelFill" class="level-fill" aria-hidden="true"></div>
    </div>
    <div id="timerDisplay" aria-live="polite" style="min-width:50px;text-align:right;font-size:13px;color:var(--muted)"></div>
  </div>

  <h2 class="question" id="questionText">Question text</h2>

  <div class="options" id="optionsArea" role="list"></div>

  <div class="controls" style="margin-top:12px;">
    <div class="left-controls">
      <button class="small hint" id="hintBtn" aria-label="Hint (costs sticker)">💡 Hint</button>
      <button class="small" id="previewBtn" aria-label="Preview Question">👀 Preview</button>
      <div id="stickerBox" style="margin-left:8px;"><span id="stickerCount">Stickers: 0</span></div>
    </div>
    <div class="right-controls">
      <button class="small mute" id="muteBtn" aria-label="Mute/unmute sounds">🔇 </button>
      <button class="small" id="nameBtn" aria-label="Set child"></button>
    </div>
  </div>
</div>
```

```
name">👤 Name</button>
    <button class="small next-btn" id="nextBtn" aria-label="Next question">Next</button>
</div>
</div>

<div style="margin-top:10px; display:flex; justify-content:space-between; align-items:center;">
    <div id="message" style="color:var(--muted); font-weight:600"></div>
    <div class="buddy" id="buddy" aria-hidden="true">😊</div>
</div>
</div>
```

## Game Controls

Theme, difficulty, timer & more

Streak: 0  
Best: 0

```
<hr style="margin:8px 0">

<div style="display:flex; gap:8px; align-items:center; margin-bottom:8px;">
    <label style="font-size:13px; color:var(--muted)">Difficulty</label>
    <select id="difficultySelect" aria-label="Difficulty level">
        <option value="all">All</option>
        <option value="easy">Easy</option>
        <option value="medium">Medium</option>
        <option value="hard">Hard</option>
    </select>
</div>

<div style="display:flex; gap:8px; align-items:center; margin-bottom:8px;">
    <input type="checkbox" id="shuffleQ" checked>
    <label for="shuffleQ" style="font-size:13px; color:var(--muted)">Shuffle questions each game</label>
</div>
```

```
<div style="display:flex; gap:8px; align-items:center; margin-bottom:8px;">
  <input type="checkbox" id="timerToggle">
  <label for="timerToggle" style="font-size:13px; color:var(--muted)">Enable 10s timer for bonus</label>
</div>

<div id="factBook" style="margin-top:10px;">
  <div style="font-weight:700">Fact Book</div>
  <div id="factList" style="color:var(--muted); font-size:14px; min-height:54px; margin-top:6px;">No facts yet – answer correctly to collect them.</div>
</div>

<div style="margin-top:12px; display:flex; gap:8px; justify-content:space-between;">
  <button id="openSettings" class="small" style="background:var(--accent); color:white">Open Settings</button>
  <button id="resetDefault" class="small" style="background:#e74c3c; color:white">✖ Reset to Default</button>
</div>
</div>

<div class="card">
  <div style="font-weight:700">Theme Customizer</div>
  <div style="margin-top:8px;">
    <label class="small">Accent color:</label>
    <input type="color" id="accentPicker" value="#ffb02e">
  </div>
  <div style="margin-top:8px;">
    <label class="small">Buddy set:</label>
    <select id="buddySet">
      <option value="faces">😊 😃 😕 </option>
      <option value="animals">🐶 🐱 🐳 </option>
      <option value="fun">🐸 🐻 🐧 </option>
    </select>
  </div>
  <div style="margin-top:8px;">
    <label class="small">Font:</label>
```

```

<select id="fontSelect">
    <option value="Comic Sans MS, Poppins, sans-serif">Comic</option>
    <option value="Poppins, system-ui, sans-serif">Poppins</option>
    <option value="Arial, Helvetica, sans-serif">Arial</option>
</select>
</div>
<div style="margin-top:10px;">
    <button id="applyTheme" class="small" style="background:var(--accent); color:white">Apply</button>
</div>
</div>

```

## Preview

[Close](#)

## Teacher Settings — Secure

```

<!-- password box / toggle -->
<div id="pwBlock" class="row" style="align-items:center;">
    <div style="flex:1">
        <label class="small">Enter teacher code</label>
        <input type="password" id="pwInput" placeholder="Enter code" aria-label="Teacher password" />
    </div>
    <div>
        <label class="small">Show</label>
        <button id="pwToggle" class="small">👁</button>
    </div>
    <div>
        <button id="pwSubmit" class="small" style="background:var(--accent); color:white">Unlock</button>
    </div>
</div>

<hr>

```

```
<div id="settingsContent" class="hidden">
  <div style="display:flex; gap:8px; align-items:center; margin-bottom:8px;">
    <input id="searchQ" placeholder="Search questions..." />
    <button id="importBtn" class="small">Import JSON</button>
    <input type="file" id="importFile" accept=".json"
style="display:none"/>
    <button id="exportBtn" class="small">Export JSON</button>
  </div>

  <div id="questionsList"></div>

  <div style="margin-top:8px; display:flex; gap:8px; justify-content:space-between;">
    <div>
      <button id="addQBtn" class="small">+ Add Question</button>
      <button id="saveQBtn" class="small" style="background:var(--accent); color:white">Save</button>
    </div>
    <div>
      <button id="closeSettingsBtn" class="small">Close</button>
    </div>
  </div>

  <hr>

  <div style="display:flex; gap:8px; align-items:center;">
    <label class="small">Difficulty pools:</label>
    <button class="small" id="easyPool">Easy</button>
    <button class="small" id="medPool">Medium</button>
    <button class="small" id="hardPool">Hard</button>
  </div>
</div>
```

## ASSISTANT

Thought for 3 seconds > The user wants me to modify the HTML game with the following requirements: > 1. Add a name input when the website loads > 2. Remove the theme customizer > 3. Remove the in-game controls > 4. Don't add difficulty > 5. Make the game controls a button to view GUI > 6. Don't add pools > 7. Use the layout before >> Let me analyze the current code and make these changes: >> 1. Add a name input when the website loads - I'll need to create a modal or prompt that appears on page load to ask for the user's name > 2. Remove the theme customizer - I'll remove the entire theme customizer section from the right panel > 3. Remove the in-game controls - I'll remove the controls section from the right panel > 4. Don't add difficulty - I'll remove the difficulty selector > 5. Make the game controls a button to view GUI - I'll replace the controls section with a button that opens a modal with the game controls > 6. Don't add pools - I'll remove any pool-related functionality > 7. Use the layout before - I'll maintain the existing layout structure >> Let me implement these changes step by step. I'll modify the HTML game according to your requirements. Here's the updated version: <!DOCTYPE html>

```
<html lang="en">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width,initial-scale=1" />
<title>Right vs Wrong – Kids Learning Game</title>
<style>
:root{
  --accent: #ffb02e;
  --accent-dark: #ff9b00;
  --bg: #f7fbff;
  --green: #1fb16a;
  --red: #e74c3c;
  --muted: #6b7280;
  --card: #fff;
  --shadow: 0 10px 25px rgba(15,23,42,0.08);
  --radius: 14px;
  --font-main: "Comic Sans MS", "Poppins", sans-serif;
}
*{box-sizing:border-box}
body{
  margin:0; font-family:var(--font-main); background:linear-gradient(180deg,var(--bg),#fff);
  color:#0f172a; -webkit-font-smoothing:antialiased;
```

```
}

/* header */
header{
    background:linear-gradient(90deg,var(--accent),var(--accent-dark));
    color:white; padding:14px 16px; display:flex; align-items:center;
justify-content:center;
    position:relative; box-shadow:var(--shadow);
}
header h1{ margin:0; font-size:18px; letter-spacing:0.3px; }
#settingsIcon{ position:absolute; right:12px; top:8px; width:40px;
height:40px; border-radius:999px;
    display:flex; align-items:center; justify-content:center;
background:rgba(255,255,255,0.12); cursor:pointer; }
#settingsIcon:focus{outline:3px solid rgba(255,255,255,0.18)}


/* layout */
main.app{ max-width:1100px; margin:18px auto; padding:12px;
display:grid; grid-template-columns:1fr 420px; gap:20px;
transition:opacity .25s ease; }
@media(max-width:900px){ main.app{ grid-template-columns:1fr;
padding:10px } }

/* video card */
.video-card{ background:var(--card); border-radius:var(--radius);
overflow:hidden; box-shadow:var(--shadow); display:flex; flex-
direction:column; }
.video-wrap{ position:relative; padding-top:56.25%; } /* 16:9 */
.video-wrap iframe{ position:absolute; inset:0; width:100%;
height:100%; border:0; }

/* question card */
.card{ background:linear-gradient(180deg, #fff, #ffffef9); border-
radius:var(--radius); padding:14px; box-shadow:var(--shadow);
border:4px solid rgba(255,224,178,0.6); }
.progress{ color:var(--muted); font-size:13px; margin-bottom:8px;
display:flex; align-items:center; justify-content:space-between;
gap:8px; }
```

```
.level{ flex:1; height:10px; background:#eef2f7; border-radius:999px; overflow:hidden; margin-left:8px; margin-right:8px; }
.level-fill{ height:100%; width:0%; background:linear-gradient(90deg,var(--accent),var(--accent-dark));
transition:width .35s ease-out; }

h2.question{ font-size:18px; margin:6px 0 12px 0; text-align:center;
color:#0c1720; }

.options{ display:flex; flex-direction:column; gap:10px; }
.opt{ display:flex; gap:12px; align-items:center; padding:10px 12px;
background:var(--card); border-radius:10px; cursor:pointer; border:2px solid #fff6e6;
box-shadow:0 8px 18px rgba(15,23,42,0.04);
transition:transform .12s ease, box-shadow .12s ease; font-size:15px; }
.opt:hover{ transform:translateY(-3px) }
.opt .label{ font-weight:700; width:34px; height:34px;
display:inline-flex; align-items:center; justify-content:center;
border-radius:999px; background:#fff6e6; }
.opt.correct{ background:linear-gradient(90deg,
rgba(31,177,106,0.08), rgba(31,177,106,0.02)); border-color:rgba(31,177,106,0.15) }
.opt.wrong{ background:linear-gradient(90deg, rgba(231,76,60,0.06),
rgba(231,76,60,0.02)); border-color:rgba(231,76,60,0.12) }
button.small{ padding:8px 10px; border-radius:10px; font-weight:700;
cursor:pointer; border:none; }

/* controls row */
.controls{ display:flex; gap:10px; align-items:center; justify-content:space-between; margin-top:10px; flex-wrap:wrap; }
.left-controls{ display:flex; gap:8px; align-items:center; }
.right-controls{ display:flex; gap:8px; align-items:center; }

/* badges, stickers */
#stickers{ background:#fff9e9; border:3px dashed var(--accent);
padding:8px; border-radius:12px; font-size:14px; }
.floating{ position:fixed; pointer-events:none; z-index:2000; font-size:22px; animation:floatUp 1.6s ease-out forwards; }
```

```
@keyframes floatUp{ 0%{ transform:translateY(0) scale(.9); opacity:1 } 100%{ transform:translateY(-120px) scale(1.2); opacity:0 } }

/* hint button */
.hint{ background:#0b74ff; color:#fff }
.mute{ background:#e2e8f0; color:#0b1320 }

/* settings modal / panel */
.overlay{ position:fixed; inset:0; background:rgba(0,0,0,0.35); display:flex; align-items:center; justify-content:center; z-index:3000; }
.modal{ width:940px; max-width:95%; max-height:90vh; overflow:auto; background:white; padding:14px; border-radius:14px; box-shadow:0 20px 60px rgba(2,6,23,0.4); }
.modal h3{ margin-top:0 }
.row{ display:flex; gap:8px; align-items:center; margin-bottom:8px; flex-wrap:wrap; }
.field{ flex:1; min-width:180px; }
label.small{ font-size:12px; color:var(--muted); display:block; margin-bottom:4px }
input[type="text"], textarea, select{
  width:100%; padding:8px; border-radius:8px; border:1px solid #ddd; font-size:14px;
}
textarea{ min-height:54px; resize:vertical; }

.qcard{ border:1px solid #f0e3c9; padding:8px; border-radius:8px; margin-bottom:8px; background:#fffef8; display:flex; gap:10px; align-items:flex-start; }
.q-actions{ display:flex; gap:6px; flex-direction:column; }

/* search bar */
.search{ display:flex; gap:8px; align-items:center; margin-bottom:8px; }

/* preview frame */
.preview{ border-radius:10px; overflow:hidden; border:1px solid #eee; margin-top:8px }
```

```
/* footer */
footer{ text-align:center; color:var(--muted); margin:20px 0 40px;
font-size:14px; }

/* confetti canvas */
#confettiCanvas{ position:fixed; left:0; top:0; width:100%;
height:100%; pointer-events:none; z-index:2500 }

/* transitions */
.fade{ transition: opacity .25s ease; }
.hidden{ display:none !important; opacity:0 !important }

/* buddy animations */
.buddy{ font-size:36px; display:inline-block;
transition:transform .35s ease, filter .35s; }
.jump{ transform:translateY(-10px) scale(1.1); }
.shake{ animation:shake .6s; }
@keyframes shake{ 10%{ transform:translateX(-4px) }
30%{ transform:translateX(4px) } 50%{ transform:translateX(-3px) }
70%{ transform:translateX(3px) } 90%{ transform:translateX(-1px) } }

/* results */
.results{ display:flex; flex-direction:column; gap:10px; align-
items:center; text-align:center; padding:12px }
.badge{ background:#fff9e9; border:2px solid var(--accent);
padding:8px 12px; border-radius:10px; font-weight:700 }

/* accessibility large text toggle (class toggled by settings) */
.large-text{ font-size:18px; }
</style>
</head>
<body>

<header>
  <h1>Right vs Wrong – Kids Learning</h1>
  <div id="settingsIcon" tabindex="0" title="Teacher
Settings">⚙</div>
</header>
```

```
<main class="app" id="app">
  <!-- LEFT: video + controls -->
  <div class="video-card card" aria-live="polite">
    <div class="video-wrap" id="videoWrap" role="region" aria-
label="Scenario video">
      <iframe id="videoFrame" src="" allow="accelerometer; autoplay;
clipboard-write; encrypted-media; gyroscope; picture-in-picture"
allowfullscreen title="Scenario video"></iframe>
    </div>

    <div style="padding:12px;">
      <div class="progress" aria-hidden="false">
        <div id="progressText">Question 1 of 5</div>
        <div class="level">
          <div id="levelFill" class="level-fill" aria-
hidden="true"></div>
        </div>
        <div id="timerDisplay" aria-live="polite" style="min-
width:50px;text-align:right;font-size:13px;color:var(--muted)"></div>
      </div>

      <h2 class="question" id="questionText">Question text</h2>

      <div class="options" id="optionsArea" role="list"></div>

      <div class="controls" style="margin-top:12px;">
        <div class="left-controls">
          <button class="small hint" id="hintBtn" aria-label="Hint
(costs sticker)">💡 Hint</button>
          <button class="small" id="previewBtn" aria-label="Preview
Question">👀 Preview</button>
          <div id="stickerBox" style="margin-left:8px;"><span
id="stickerCount">Stickers: 0</span></div>
        </div>
        <div class="right-controls">
          <button class="small mute" id="muteBtn" aria-
label="Mute/unmute sounds">🔇 </button>
          <button class="small" id="nameBtn" aria-label="Set child
name">👤 </button>
        </div>
      </div>
    </div>
  </div>
</main>
```

```
name">👤 Name</button>
    <button class="small next-btn" id="nextBtn" aria-label="Next question">Next</button>
</div>
</div>

<div style="margin-top:10px; display:flex; justify-content:space-between; align-items:center;">
    <div id="message" style="color:var(--muted); font-weight:600"></div>
    <div class="buddy" id="buddy" aria-hidden="true">😊 </div>
</div>
</div>
</div>

<!-- RIGHT: settings summary + factbook -->
<div>
    <div class="card" style="margin-bottom:12px;">
        <div style="display:flex; justify-content:space-between; align-items:center;">
            <div>
                <div style="font-weight:700">Game Controls</div>
                <div style="color:var(--muted); font-size:13px">View game settings and progress</div>
            </div>
            <div style="text-align:right;">
                <div style="font-size:13px; color:var(--muted)">Streak:<span id="streakCount">0</span></div>
                <div style="font-size:12px; color:var(--muted)">Best: <span id="bestScoreDisplay">0</span></div>
            </div>
        </div>
    </div>
<hr style="margin:8px 0">

<div id="factBook" style="margin-top:10px;">
    <div style="font-weight:700">Fact Book</div>
    <div id="factList" style="color:var(--muted); font-size:14px; min-height:54px; margin-top:6px;">No facts yet – answer correctly to</div>
```

```
collect them.</div>
    </div>

        <div style="margin-top:12px; display:flex; gap:8px; justify-content:space-between;">
            <button id="openControls" class="small" style="background:var(--accent); color:white">View Game Controls</button>
            <button id="resetDefault" class="small" style="background:#e74c3c; color:white">🔗 Reset to Default</button>
        </div>
    </div>
</main>

<!-- name modal -->
<div id="nameModal" class="overlay" aria-hidden="false">
    <div class="modal" role="dialog" aria-modal="true" aria-label="Enter your name">
        <h3>Welcome to Right vs Wrong!</h3>
        <p>Please enter your name to get started:</p>
        <div style="margin-top:10px;">
            <input type="text" id="nameInput" placeholder="Your name" style="width:100%; padding:10px; border-radius:8px; border:1px solid #ddd; font-size:16px;" />
        </div>
        <div style="margin-top:15px; text-align:right;">
            <button id="submitName" class="small" style="background:var(--accent); color:white; padding:10px 20px;">Start Playing</button>
        </div>
    </div>
</div>

<!-- game controls modal -->
<div id="controlsModal" class="overlay hidden" aria-hidden="true">
    <div class="modal" role="dialog" aria-modal="true" aria-label="Game Controls">
        <h3>Game Controls</h3>
```

```
<div style="display:flex; gap:8px; align-items:center; margin-bottom:8px;">
    <input type="checkbox" id="shuffleQ" checked>
    <label for="shuffleQ" style="font-size:13px; color:var(--muted)">Shuffle questions each game</label>
</div>

<div style="display:flex; gap:8px; align-items:center; margin-bottom:8px;">
    <input type="checkbox" id="timerToggle">
    <label for="timerToggle" style="font-size:13px; color:var(--muted)">Enable 10s timer for bonus</label>
</div>

<div style="margin-top:15px; text-align:right;">
    <button id="closeControls" class="small" style="background:var(--accent); color:white; padding:10px 20px;">Close</button>
</div>
</div>
</div>

<!-- preview modal -->
<div id="previewModal" class="overlay hidden" aria-hidden="true">
    <div class="modal" role="dialog" aria-modal="true" aria-label="Preview question">
        <h3>Preview</h3>
        <div id="previewArea"></div>
        <div style="margin-top:8px; text-align:right;">
            <button onclick="closePreview()" class="small">Close</button>
        </div>
    </div>
</div>

<!-- settings modal (password + management) -->
<div id="settingsModal" class="overlay hidden" aria-hidden="true">
    <div class="modal" role="dialog" aria-modal="true" aria-label="Teacher settings">
        <h3>Teacher Settings – Secure</h3>
```

```
<!-- password box / toggle -->
<div id="pwBlock" class="row" style="align-items:center;">
    <div style="flex:1">
        <label class="small">Enter teacher code</label>
        <input type="password" id="pwInput" placeholder="Enter code"
aria-label="Teacher password" />
    </div>
    <div>
        <label class="small">Show</label>
        <button id="pwToggle" class="small">● </button>
    </div>
    <div>
        <button id="pwSubmit" class="small" style="background:var(--accent); color:white">Unlock</button>
    </div>
</div>

<hr>

<div id="settingsContent" class="hidden">
    <div style="display:flex; gap:8px; align-items:center; margin-bottom:8px;">
        <input id="searchQ" placeholder="Search questions..." />
        <button id="importBtn" class="small">Import JSON</button>
        <input type="file" id="importFile" accept=".json"
style="display:none"/>
        <button id="exportBtn" class="small">Export JSON</button>
    </div>

    <div id="questionsList"></div>

    <div style="margin-top:8px; display:flex; gap:8px; justify-content:space-between;">
        <div>
            <button id="addQBtn" class="small">+ Add Question</button>
            <button id="saveQBtn" class="small" style="background:var(--accent); color:white">Save</button>
        </div>
        <div>
```

```
        <button id="closeSettingsBtn" class="small">Close</button>
      </div>
    </div>
  </div>

</div>
</div>

<!-- confetti canvas -->
<canvas id="confettiCanvas"></canvas>

<footer>© Right vs Wrong – built for curious kids</footer>

<!-- sounds -->
<audio id="s_correct"
src="https://actions.google.com/sounds/v1/cartoon/pop.ogg"
preload="auto"></audio>
<audio id="s_wrong"
src="https://actions.google.com/sounds/v1/cartoon/wood_plank_flicks.ogg"
preload="auto"></audio>
<audio id="s_end"
src="https://actions.google.com/sounds/v1/cartoon/slide_whistle_to_drum_hit.ogg"
preload="auto"></audio>

<script>
/* =====
   App Data & Defaults
===== */

const DEFAULT_DATA = [
  { id: "q1", difficulty:"easy",
video:"https://www.youtube.com/embed/_7R4SfIsINU",
    question:"What did the kid do wrong after finishing their snack?", 
    options:["Threw the wrapper on the ground","Put the wrapper in the bin","Kept the wrapper in their pocket"],
    correct:0, hint:"Think about the wrapper", fact:"Putting trash in a bin keeps places clean." },
  { id: "q2", difficulty:"easy",
video:"https://www.youtube.com/embed/dQw4w9WgXcQ",
```

```

        question:"What mistake did the kid make while washing hands?",  

        options:["Left the water running","Turned off the tap before  

getting soap","Used too much soap"],  

        correct:0, hint:"Water matters", fact:"Turning taps off saves  

water." },  

{ id: "q3", difficulty:"medium",  

video:"https://www.youtube.com/embed/M7lc1UVf-VE",  

    question:"What did the kid forget to do before leaving their  

room?",  

    options:["Forgot to turn off the lights","Closed the window","Put  

away their toys"],  

    correct:0, hint:"Think energy", fact:"Turning off lights saves  

electricity." },  

{ id: "q4", difficulty:"easy",  

video:"https://www.youtube.com/embed/ScMzIvxBSi4",  

    question:"What should you do before crossing the road?",  

    options:["Look both ways","Run quickly","Close your eyes"],  

    correct:0, hint:"Be safe", fact:"Always look both ways before  

crossing." },  

{ id: "q5", difficulty:"medium",  

video:"https://www.youtube.com/embed/tgbNymZ7vqY",  

    question:"What is wrong if you don't wash fruits?",  

    options:["They can have dirt or germs","They taste bad","They  

become too soft"],  

    correct:0, hint:"Think hygiene", fact:"Washing fruit removes dirt  

and germs." }
];

```

```

let DATA = JSON.parse(localStorage.getItem("rvs_quiz")) ||  

DEFAULT_DATA.slice();  

let SETTINGS = JSON.parse(localStorage.getItem("rvs_settings")) || {  

    accent:  

getComputedStyle(document.documentElement).getPropertyValue('--  

accent').trim() || "#ffb02e",  

    buddySet: "faces",  

    font: "Comic Sans MS, Poppins, sans-serif",  

    shuffle: true,  

    timerEnabled: false
};

```

```
let state = {
    order: [], // randomized question order
    idx: 0,
    score: 0,
    stickers: JSON.parse(localStorage.getItem("rvs_stickers")) || [],
    streak: parseInt(localStorage.getItem("rvs_streak"))||0,
    best: parseInt(localStorage.getItem("rvs_best"))||0,
    timer: null,
    timerSec: 10,
    mute: false,
    childName: localStorage.getItem("rvs_childName") || "",
    hintPenalty: 1, // subtract 1 point when using hint
    shuffleQuestions: SETTINGS.shuffle,
    lastPlayed: JSON.parse(localStorage.getItem("rvs_lastPlayed")) || {}
};

/* DOM refs */
const videoFrame = document.getElementById("videoFrame");
const questionText = document.getElementById("questionText");
const optionsArea = document.getElementById("optionsArea");
const progressText = document.getElementById("progressText");
const levelFill = document.getElementById("levelFill");
const nextBtn = document.getElementById("nextBtn");
const hintBtn = document.getElementById("hintBtn");
const stickerCount = document.getElementById("stickerCount");
const messageDiv = document.getElementById("message");
const buddyEl = document.getElementById("buddy");
const timerDisplay = document.getElementById("timerDisplay");
const previewModal = document.getElementById("previewModal");
const previewArea = document.getElementById("previewArea");
const settingsIcon = document.getElementById("settingsIcon");
const settingsModal = document.getElementById("settingsModal");
const pwInput = document.getElementById("pwInput");
const pwToggle = document.getElementById("pwToggle");
const pwSubmit = document.getElementById("pwSubmit");
const settingsContent = document.getElementById("settingsContent");
const questionsList = document.getElementById("questionsList");
const searchQ = document.getElementById("searchQ");
```

```
const importFile = document.getElementById("importFile");
const confettiCanvas = document.getElementById("confettiCanvas");
const s_correct = document.getElementById("s_correct");
const s_wrong = document.getElementById("s_wrong");
const s_end = document.getElementById("s_end");
const resetDefaultBtn = document.getElementById("resetDefault");
const muteBtn = document.getElementById("muteBtn");
const previewBtn = document.getElementById("previewBtn");
const shuffleQ = document.getElementById("shuffleQ");
const timerToggle = document.getElementById("timerToggle");
const nameBtn = document.getElementById("nameBtn");
const factList = document.getElementById("factList");
const streakCount = document.getElementById("streakCount");
const bestScoreDisplay = document.getElementById("bestScoreDisplay");
const nameModal = document.getElementById("nameModal");
const nameInput = document.getElementById("nameInput");
const submitName = document.getElementById("submitName");
const controlsModal = document.getElementById("controlsModal");
const openControls = document.getElementById("openControls");
const closeControls = document.getElementById("closeControls");

/* apply saved theme to UI */
function applySavedTheme(){
    document.documentElement.style.setProperty("--accent",
SETTINGS.accent||"#ffb02e");
    document.documentElement.style.setProperty("--accent-dark",
shadeColor(SETTINGS.accent||"#ffb02e", -10));
    document.documentElement.style.setProperty("--font-main",
SETTINGS.font||"Comic Sans MS, Poppins, sans-serif");
    document.body.style.fontFamily = SETTINGS.font;
}
applySavedTheme();

/* small util: color shade for accent dark */
function shadeColor(color, percent) {
    // color in #rrggbba
    const num = parseInt(color.slice(1),16);
    const r = (num >> 16) + percent;
    const g = (num >> 8 & 0x00FF) + percent;
```

```

const b = (num & 0x0000FF) + percent;
const newR = Math.min(255, Math.max(0, r));
const newG = Math.min(255, Math.max(0, g));
const newB = Math.min(255, Math.max(0, b));
return `#${{(newR<<16 | newG<<8 |
newB).toString(16).padStart(6,'0')}}`;
}

/*
=====
Security: password hashing via Web Crypto
=====
*/
const CODE_KEY = "rvs_code_hash";
async function hashCode(str){
    const enc = new TextEncoder().encode(str);
    const hashBuffer = await crypto.subtle.digest('SHA-256', enc);
    const hashArray = Array.from(new Uint8Array(hashBuffer));
    const hashHex =
hashArray.map(b=>b.toString(16).padStart(2,'0')).join('');
    return hashHex;
}
// on first load: if no code stored, store default hashed "67"
(async()=>{
    if(!localStorage.getItem(CODE_KEY)){
        const h = await hashCode("67");
        localStorage.setItem(CODE_KEY, h);
    }
})();

/* password modal handlers */
pwToggle.addEventListener("click", ()=>{
    if(pwInput.type === "password") pwInput.type = "text"; else
pwInput.type = "password";
});
pwSubmit.addEventListener("click", async ()=>{
    const attempt = pwInput.value.trim();
    if(!attempt) return alert("Enter code");
    const hashed = await hashCode(attempt);
    if(hashed === localStorage.getItem(CODE_KEY)){
        // unlocked
    }
});

```

```

        settingsContent.classList.remove("hidden");
        document.getElementById("pwBlock").classList.add("hidden");
        buildSettingsList();
    } else {
        alert("Wrong code");
    }
});

/* open settings from header icon */
settingsIcon.addEventListener("click", ()=>openSettingsModal());
function openSettingsModal(){
    settingsModal.classList.remove("hidden");
    settingsModal.style.display = "flex";
    pwInput.value = "";
    settingsContent.classList.add("hidden");
    document.getElementById("pwBlock").classList.remove("hidden");
    buildSettingsList();
}
document.getElementById("closeSettingsBtn").addEventListener("click",
()=>{
    settingsModal.classList.add("hidden");
});

/* =====
SETTINGS: build question editor list
===== */
function buildSettingsList(filter=""){
    questionsList.innerHTML = "";
    DATA.forEach((q,i)=>{
        if(filter && !(q.question+q.options.join(" ") +
(q.fact|| "") ).toLowerCase().includes(filter.toLowerCase())) return;
        const card = document.createElement("div");
        card.className = "qcard";
        card.innerHTML = `
            <div style="flex:1">
                <label class="small">Video URL</label>
                <input type="text" class="sq_vid"
value="\${escapeHtml(q.video)}"/>
                <label class="small">Question</label>
        `;
    });
}

```

```

        <input type="text" class="sq_q"
value="${escapeHtml(q.question)}"/>
        <label class="small">Options (comma separated)</label>
        <textarea
class="sq_opts">${escapeHtml(q.options.join(","))}</textarea>
        <label class="small">Correct Index (0-based)</label>
        <input type="number" class="sq_correct" value="${q.correct}"
min="0" style="width:80px"/>
        <label class="small">Hint</label>
        <input type="text" class="sq_hint"
value="${escapeHtml(q.hint||"")}" />
        <label class="small">Fact</label>
        <input type="text" class="sq_fact"
value="${escapeHtml(q.fact||"")}" />
    </div>
    <div class="q-actions">
        <button class="small" data-action="up"> </button>
        <button class="small" data-action="down"> </button>
        <button class="small" data-action="preview"> </button>
        <button class="small" data-action="remove"
style="background:#e74c3c;color:white">Remove</button>
    </div>
`;
// attach handlers with closure index
const up = card.querySelector('[data-action="up"]');
const down = card.querySelector('[data-action="down"]');
const remove = card.querySelector('[data-action="remove"]');
const preview = card.querySelector('[data-action="preview"]');
up.addEventListener("click", ()=>{ if(i>0){ [DATA[i-1],DATA[i]]=[DATA[i],DATA[i-1]]; buildSettingsList(filter); }});
down.addEventListener("click", ()=>{ if(i<DATA.length-1){ [DATA[i+1],DATA[i]]=[DATA[i],DATA[i+1]]; buildSettingsList(filter); }});
remove.addEventListener("click", ()=>{ if(confirm("Remove this question?")){ DATA.splice(i,1); buildSettingsList(filter); } });
preview.addEventListener("click", ()=> previewQuestion(i) );
questionsList.appendChild(card);
});
}

```

```

searchQ && searchQ.addEventListener("input", ()=>
buildSettingsList(searchQ.value));

function addQuestion(){
  DATA.push({id:"q"+Date.now(), difficulty:"easy", video:"",
question:"New question", options:["Option A","Option B","Option C"],
correct:0, hint:"", fact:""});
  buildSettingsList();
}
document.getElementById("addQBtn").addEventListener("click",
addQuestion);

/* save settings / export / import */
document.getElementById("saveQBtn").addEventListener("click", ()=> {
  // read back UI into DATA
  const cards = questionsList.querySelectorAll('.qcard');
  const newData=[];
  cards.forEach(card=>{
    const vid = card.querySelector('.sq_vid').value.trim();
    const qtxt = card.querySelector('.sq_q').value.trim();
    const optsRaw = card.querySelector('.sq_opts').value.trim();
    const correct =
      parseInt(card.querySelector('.sq_correct').value)||0;
    const hint = card.querySelector('.sq_hint').value.trim();
    const fact = card.querySelector('.sq_fact').value.trim();
    const opts = optsRaw.split(',').map(s=>s.trim()).filter(Boolean);
    if(qtxt && opts.length>=2){
      newData.push({ id:"q"+Date.now()+Math.random(),
difficulty:"easy", video:vid, question:qtxt, options:opts, correct:
Math.min(correct, opts.length-1), hint, fact });
    }
  });
  if(newData.length>0){
    DATA = newData;
    localStorage.setItem("rvs_quiz", JSON.stringify(DATA));
    alert("Saved. New question set will be used.");
    settingsModal.classList.add("hidden");
    resetGameToStart();
  } else {

```

```

        alert("No valid questions found in editor.");
    }
});

document.getElementById("exportBtn").addEventListener("click", ()=>{
    const blob = new Blob([JSON.stringify(DATA, null,
2)],{type:"application/json"});
    const url = URL.createObjectURL(blob);
    const a = document.createElement("a");
    a.href = url; a.download = "rvs_quiz_export.json"; a.click();
    URL.revokeObjectURL(url);
});
document.getElementById("importBtn").addEventListener("click", ()=>
importFile.click());
importFile.addEventListener("change", (e)=> {
    const file = e.target.files[0];
    if(!file) return;
    const fr = new FileReader();
    fr.onload = ()=> {
        try{ const parsed = JSON.parse(fr.result);
if(Array.isArray(parsed)){ DATA = parsed;
localStorage.setItem("rvs_quiz", JSON.stringify(DATA));
alert("Imported."); buildSettingsList(); resetGameToStart(); } else
alert("JSON is not an array.");}
        catch(err){ alert("Invalid JSON file."); }
    };
    fr.readAsText(file);
});
}

document.getElementById("resetDefault").addEventListener("click",
()=>{
    if(confirm("Reset to default questions? This will clear custom
questions.")){
        DATA = DEFAULT_DATA.slice();
        localStorage.removeItem("rvs_quiz");
        resetGameToStart();
    }
});

```

```

/* preview question */
function previewQuestion(idx){
    const q = DATA[idx];
    previewArea.innerHTML = `<div style="display:flex;gap:10px;align-items:flex-start;">
        <div style="flex:1;">
            <div style="border-radius:10px;overflow:hidden;border:1px solid #eee;">
                <iframe src="${escapeHtml(q.video)}" style="width:100%;height:220px; border:0;"></iframe>
            </div>
        </div>
        <div style="width:320px;">
            <h4 style="margin:4px 0">${escapeHtml(q.question)}</h4>
            <ul>${q.options.map((o,i)=>`<li style="margin:6px 0">${String.fromCharCode(65+i)}. ${escapeHtml(o)}</li>`).join('')}</ul>
            <div style="color:var(--muted)"><strong>Hint:</strong> ${escapeHtml(q.hint|| "")}</div>
            <div style="color:var(--muted)"><strong>Fact:</strong> ${escapeHtml(q.fact|| "")}</div>
        </div>
    </div>`;
    previewModal.classList.remove("hidden");
    previewModal.style.display="flex";
}
function closePreview(){ previewModal.classList.add("hidden"); }

/* =====
 Game helper utilities
 ===== */
function escapeHtml(s){ return s?.replaceAll?."&","&").replaceAll?."<","<").replaceAll?.">",">") ?? s; }

function shuffle(arr){
    const a = arr.slice();
    for(let i=a.length-1;i>0;i--){
        const j = Math.floor(Math.random()*(i+1));

```

```

        [a[i],a[j]] = [a[j],a[i]];
    }
    return a;
}

/* =====
Game initialization & main loop
===== */
function buildPool(){
    let pool = DATA.slice();
    if(document.getElementById("shuffleQ").checked) pool =
shuffle(pool);
    state.order = pool;
    state.idx = 0;
    state.score = 0;
    state.timerSec = 10;
    state.stickers = JSON.parse(localStorage.getItem("rvs_stickers")) ||
[];
    document.getElementById("stickerCount").textContent = "Stickers:
"+state.stickers.length;
}
function loadQuestion(){
    if(state.idx >= state.order.length) { showResults(); return; }
    const q = state.order[state.idx];
    // load video and question
    videoFrame.src = q.video;
    questionText.textContent = q.question;
    progressText.textContent = `Question ${state.idx+1} of
${state.order.length}`;
    // build options (shuffled)
    optionsArea.innerHTML = "";
    const shuffled = shuffle(q.options.map((opt, i)=>({opt,i})));
    shuffled.forEach((o, aidx)=>{
        const div = document.createElement("div");
        div.className = "opt";
        div.setAttribute("role","listitem");
        div.innerHTML = `<div
class="label">${String.fromCharCode(65+aidx)}</div><div
style="flex:1">${o.opt}</div>`;

```

```

        div.addEventListener("click", ()=> handleAnswer(div, q, o.i));
        optionsArea.appendChild(div);
    });
    // reset UI
    nextBtn.classList.remove("show");
    messageDiv.textContent = "";
    buddySetIdle();
    updateLevelBar();
    // timer
    clearTimer();
    if(document.getElementById("timerToggle").checked){
        startTimer();
    } else { timerDisplay.textContent = "";}
    // save last played
    localStorage.setItem("rvs_lastPlayed", JSON.stringify({ idx:
state.idx, timestamp: Date.now() }));
}

/* visual level bar */
function updateLevelBar(){
    const pct = Math.round((state.idx / Math.max(1, state.order.length)) * 100);
    levelFill.style.width = pct + "%";
}

/* answer handling */
function handleAnswer(div, q, chosenIndex){
    // disable all options
    const all = optionsArea.querySelectorAll(".opt");
    all.forEach(o=>o.style.pointerEvents="none");
    clearTimer();
    const correct = q.correct;
    if(chosenIndex === correct){
        div.classList.add("correct");
        if(!state.mute) s_correct.play();
        // sticker reward: random sticker emoji
        const newSticker = randomSticker();
        state.stickers.push(newSticker);
        showFloatingSticker(newSticker);
    }
}

```

```

// collect fact into factBook (only on correct)
addFact(q.fact);
state.score += 1;
// buddy animation positive
buddyCelebrate();
messageDiv.textContent = "Great job! " + (q.fact || "");
state.streak = (state.streak || 0) + 1;
} else {
  div.classList.add("wrong");
  // reveal correct option visually
  all.forEach((el, idx) => {
    const txt = el.textContent.trim();
    // find which one matches q.options[correct]
    if(el.textContent.includes(q.options[correct]))
      el.classList.add("correct");
  });
  if(!state.mute) s_wrong.play();
  messageDiv.textContent = "Oops! " + (q.fact || "");
  state.streak = 0;
}
// show next
nextBtn.classList.add("show");
// save stickers and streak/best
localStorage.setItem("rvs_stickers",
JSON.stringify(state.stickers));
localStorage.setItem("rvs_streak", state.streak);
if(state.score > state.best){ state.best = state.score;
localStorage.setItem("rvs_best", state.best); }
// update counts display
document.getElementById("stickerCount").textContent = "Stickers: " +
state.stickers.length;
streakCount.textContent = state.streak;
bestScoreDisplay.textContent = state.best;
}

/* next button */
nextBtn.addEventListener("click", ()=>{
  state.idx++;
  if(state.idx < state.order.length) {

```

```

        updateLevelBarAnimate();
        loadQuestion();
    } else {
        showResults();
    }
});

/* hint button */
hintBtn.addEventListener("click", ()=>{
    const q = state.order[state.idx];
    if(!q) return;
    // penalty: deduct one point if available (but not below 0)
    if(state.score > 0){ state.score = Math.max(0, state.score -
state.hintPenalty); }
    messageDiv.textContent = "💡 " + (q.hint || "Think carefully!");
    // deduct sticker if any exist (choose: cost 1 sticker if present)
    if(state.stickers.length>0) state.stickers.pop();
    localStorage.setItem("rvs_stickers",
JSON.stringify(state.stickers));
    document.getElementById("stickerCount").textContent = "Stickers: " +
state.stickers.length;
});

/* show results with confetti and fact book */
function showResults(){
    // confetti
    runConfetti();
    if(!state.mute) s_end.play();
    // show results modal-like area: reuse right side results area
    document.getElementById("app").classList.add("fade");

setTimeout(()=>document.getElementById("app").classList.remove("fade")
, 300);

    // assemble results in factList and results area
    // quick results display (alert-like)
    const name = state.childName || "Friend";
    const finalMsg = `${name}, you got ${state.score} out of
${state.order.length}!`;

```

```

    alert(finalMsg + "\nYour stickers: " + state.stickers.join(" ") +
"\nFacts collected in fact book.");
    // save best
    if(state.score > state.best) { state.best = state.score;
localStorage.setItem("rvs_best", state.best); }
    bestScoreDisplay.textContent = state.best;
    // update fact list already maintained
    // reset for next play
    resetGameToStart();
}

/* reset game to start */
function resetGameToStart(){
    buildPool();
    renderFactBook();
    loadQuestion();
}

/* initial build & start */
document.getElementById("shuffleQ").addEventListener("change", ()=>
SETTINGS.shuffle = document.getElementById("shuffleQ").checked );
document.getElementById("timerToggle").addEventListener("change", ()=>
{ /* nothing immediate */ });
document.getElementById("muteBtn").addEventListener("click", ()=>
{ state.mute = !state.mute; muteBtn.textContent = state.mute ? '🔇' :
'🔈'; });
document.getElementById("previewBtn").addEventListener("click", ()=>
{ previewCurrentQuestion(); });

document.getElementById("nameBtn").addEventListener("click", ()=>{
    const n = prompt("Enter child's name (for results)", "");
    if(n){ state.childName = n; localStorage.setItem("rvs_childName", n); alert("Saved name: "+n); }
});

/* name modal handlers */
submitName.addEventListener("click", ()=>{
    const name = nameInput.value.trim();
    if(name){

```

```

state.childName = name;
localStorage.setItem("rvs_childName", name);
nameModal.classList.add("hidden");
loadQuestion();
} else {
  alert("Please enter your name");
}
});

/* game controls modal handlers */
openControls.addEventListener("click", ()=>{
  controlsModal.classList.remove("hidden");
  controlsModal.style.display = "flex";
});

closeControls.addEventListener("click", ()=>{
  controlsModal.classList.add("hidden");
});

/* settings header icon opens modal but triggers pw flow earlier */
settingsIcon.addEventListener("dblclick", ()=> openSettingsModal());
// doubleclick alternate

/* preview current question */
function previewCurrentQuestion(){
  const q = state.order[state.idx];
  if(!q) { alert("No question loaded"); return; }
  previewArea.innerHTML = `<div style="display:flex;gap:8px;">
    <div style="flex:1; min-width:220px;">
      <div style="border-radius:8px; overflow:hidden;"><iframe
src="${q.video}"
style="width:100%;height:180px; border:0"></iframe></div>
    </div>
    <div style="width:260px">
      <h4>${escapeHtml(q.question)}</h4>
      <ul>${q.options.map(o=>`<li>${escapeHtml(o)}</li>`).join('')}</ul>
      <p style="color:var(--muted)"><strong>Hint:</strong>
      ${escapeHtml(q.hint||"")}</p>
    </div>
  </div>`;
}

```

```

        <p style="color:var(--muted)"><strong>Fact:</strong>
${escapeHtml(q.fact|| "")}</p>
    </div>
</div>`;
previewModal.classList.remove("hidden");
}
previewModal.addEventListener("click", (e)=> { if(e.target ===
previewModal) closePreview(); });
function closePreview(){ previewModal.classList.add("hidden");
previewModal.style.display = "none"; }

/* random sticker emoji generator */
const STICKERS =
["🎉", "🐝", "🌟", "🌈", "🍀", "🦄", "⭐", ".ribbon", "🍎", "🤖"];
function randomSticker(){ return
STICKERS[Math.floor(Math.random()*STICKERS.length)]; }
function showFloatingSticker(emoji){
    const el = document.createElement("div");
    el.className = "floating";
    el.style.left = (30 + Math.random()*60) + "vw";
    el.textContent = emoji;
    document.body.appendChild(el);
    setTimeout(()=> el.remove(), 1600);
}

/* buddy animations */
function buddyCelebrate(){
    buddyEl.classList.add("jump");
    setTimeout(()=> buddyEl.classList.remove("jump"), 600);
}
function buddySetIdle(){ buddyEl.classList.remove("jump", "shake"); }

/* timer */
function startTimer(){
    let sec = 10;
    timerDisplay.textContent = sec + "s";
    state.timer = setInterval(()=>{
        sec--;
        timerDisplay.textContent = sec + "s";
    })
}

```

```

if(sec<=0){
    clearTimer();
    // time's up: auto mark wrong and reveal
    messageDiv.textContent = "Time's up!";
    // simulate picking nothing: reveal correct
    const all = optionsArea.querySelectorAll(".opt");
    all.forEach(el=>{

        if(el.textContent.includes(state.order[state.idx].options[state.order[state.idx].correct])) el.classList.add("correct");
            el.style.pointerEvents = "none";
        });
        if(!state.mute) s_wrong.play();
        nextBtn.classList.add("show");
    }
},1000);
}
function clearTimer(){ if(state.timer){ clearInterval(state.timer);
state.timer=null; } timerDisplay.textContent = ""; }

/* confetti implementation on canvas (simple) */
function runConfetti(){
    const canvas = confettiCanvas;
    const ctx = canvas.getContext('2d');
    canvas.width = window.innerWidth;
    canvas.height = window.innerHeight;
    const pieces = [];
    for(let i=0;i<80;i++){
        pieces.push({
            x: Math.random()*canvas.width,
            y: Math.random()*-canvas.height,
            r: 6+Math.random()*8,
            color: `hsl(${Math.random()*360},70%,60%)`,
            vx: -2 + Math.random()*4,
            vy: 2 + Math.random()*6,
            rot: Math.random()*360
        });
    }
    let t=0;

```

```

function frame(){
    t++;
    ctx.clearRect(0,0,canvas.width,canvas.height);
    for(const p of pieces){
        p.x += p.vx; p.y += p.vy; p.rot += 4;
        ctx.save();
        ctx.fillStyle = p.color;
        ctx.translate(p.x, p.y);
        ctx.rotate(p.rot * Math.PI/180);
        ctx.fillRect(-p.r/2, -p.r/2, p.r, p.r);
        ctx.restore();
    }
    if(t<120) requestAnimationFrame(frame);
    else ctx.clearRect(0,0,canvas.width,canvas.height);
}
frame();

/* fact book */
function addFact(f){
    if(!f) return;
    const facts = JSON.parse(localStorage.getItem("rvs_facts")||"[]");
    if(!facts.includes(f)) facts.push(f);
    localStorage.setItem("rvs_facts", JSON.stringify(facts));
    renderFactBook();
}
function renderFactBook(){
    const facts = JSON.parse(localStorage.getItem("rvs_facts")||"[]");
    if(facts.length==0) factList.textContent = "No facts yet – answer correctly to collect them.";
    else factList.innerHTML = facts.map(f=>`

${escapeHtml(f)}

`).join('');
}

/* UI: show settings modal from header icon (opens password UI) */
settingsIcon.addEventListener("click", ()=> {
    settingsModal.classList.remove("hidden");
});
document.getElementById("closeSettingsBtn").addEventListener("click",

```

```
()=> settingsModal.classList.add("hidden"));

/* reset game / start */
document.addEventListener("DOMContentLoaded", ()=>{
    // Check if name is already saved
    if(!state.childName) {
        // Show name modal
        nameModal.style.display = "flex";
    } else {
        // Name already exists, start game
        buildPool();
        renderFactBook();
        loadQuestion();
    }
    // load persisted theme
    document.getElementById("stickerCount").textContent = "Stickers: " +
    (state.stickers.length||0);
});

/* helper: reset everything and start new */
function resetAll(){
    state.idx = 0; state.score = 0; state.stickers = [];
    localStorage.removeItem("rvs_stickers");
    localStorage.removeItem("rvs_streak");
}

/* reset to start after game or save */
function resetGameToStart(){
    state.idx = 0; state.score = 0;
    buildPool();
    loadQuestion();
}

/* timer/bonus ui animation for level bar */
function updateLevelBarAnimate(){ levelFill.style.transition =
'width .35s ease-out'; updateLevelBar(); setTimeout(()=>
levelFill.style.transition='width .35s ease-out', 400); }

/* small helpers */
```

```
function escapeHtml(s){ if(!s) return ""; return  
s.replaceAll("<","&lt;").replaceAll(">","&gt;"); }  
  
/* end of script */  
</script>  
</body>  
</html>
```

I've made the following changes to the game according to your requirements:

1. **Added a name input when the website loads:** Created a modal that appears on page load asking for the user's name. The game won't start until a name is entered.
2. **Removed the theme customizer:** Completely removed the theme customizer section from the right panel.
3. **Removed the in-game controls:** Removed the in-game controls section from the right panel.
4. **Removed difficulty:** Removed the difficulty selector from the game.
5. **Made the game controls a button to view GUI:** Replaced the controls section with a "View Game Controls" button that opens a modal with the game settings.
6. **Removed pools:** Removed any pool-related functionality from the code.
7. **Maintained the original layout:** Kept the original layout structure with the video on the left and the controls panel on the right.

The game now starts with a name input modal, has a simplified right panel with just the fact book and a button to view game controls, and maintains all the core functionality of the original game.

## USER

Remove preview

## ASSISTANT

Thought for 1 seconds > The user wants me to remove the preview functionality from the game. This includes: > 1. The preview button in the controls > 2. The preview modal > 3. Any related JavaScript functions for preview > 4. Any preview-related event listeners > > Let me go through the code and remove all preview-related elements. I'll remove the preview functionality from the game. Here's the updated version: <!DOCTYPE html>

```
<html lang="en">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width,initial-scale=1" />
<title>Right vs Wrong – Kids Learning Game</title>
<style>
:root{
    --accent: #ffb02e;
    --accent-dark: #ff9b00;
    --bg: #f7fbff;
    --green: #1fb16a;
    --red: #e74c3c;
    --muted: #6b7280;
    --card: #fff;
    --shadow: 0 10px 25px rgba(15,23,42,0.08);
    --radius: 14px;
    --font-main: "Comic Sans MS", "Poppins", sans-serif;
}
*{box-sizing:border-box}
body{
    margin:0; font-family:var(--font-main); background:linear-gradient(180deg,var(--bg),#fff);
    color:#0f172a; -webkit-font-smoothing:antialiased;
}

/* header */
header{
    background:linear-gradient(90deg,var(--accent),var(--accent-dark));
    color:white; padding:14px 16px; display:flex; align-items:center; justify-content:center;
    position:relative; box-shadow:var(--shadow);
}
header h1{ margin:0; font-size:18px; letter-spacing:0.3px; }
#settingsIcon{ position:absolute; right:12px; top:8px; width:40px; height:40px; border-radius:999px;
    display:flex; align-items:center; justify-content:center; background:rgba(255,255,255,0.12); cursor:pointer; }
#settingsIcon:focus{outline:3px solid rgba(255,255,255,0.18)}
```

```
/* layout */
main.app{ max-width:1100px; margin:18px auto; padding:12px;
display:grid; grid-template-columns:1fr 420px; gap:20px;
transition:opacity .25s ease; }
@media(max-width:900px){ main.app{ grid-template-columns:1fr;
padding:10px } }

/* video card */
.video-card{ background:var(--card); border-radius:var(--radius);
overflow:hidden; box-shadow:var(--shadow); display:flex; flex-
direction:column; }
.video-wrap{ position:relative; padding-top:56.25%; } /* 16:9 */
.video-wrap iframe{ position:absolute; inset:0; width:100%;
height:100%; border:0; }

/* question card */
.card{ background:linear-gradient(180deg, #fff, #ffffef9); border-
radius:var(--radius); padding:14px; box-shadow:var(--shadow);
border:4px solid rgba(255,224,178,0.6); }
.progress{ color:var(--muted); font-size:13px; margin-bottom:8px;
display:flex; align-items:center; justify-content:space-between;
gap:8px; }
.level{ flex:1; height:10px; background:#eef2f7; border-
radius:999px; overflow:hidden; margin-left:8px; margin-right:8px; }
.level-fill{ height:100%; width:0%; background:linear-
gradient(90deg,var(--accent),var(--accent-dark));
transition:width .35s ease-out; }

h2.question{ font-size:18px; margin:6px 0 12px 0; text-align:center;
color:#0c1720; }

.options{ display:flex; flex-direction:column; gap:10px; }
.opt{ display:flex; gap:12px; align-items:center; padding:10px 12px;
background:var(--card); border-radius:10px; cursor:pointer; border:2px
solid #fff6e6;
box-shadow:0 8px 18px rgba(15,23,42,0.04);
transition:transform .12s ease, box-shadow .12s ease; font-
size:15px; }
```

```
.opt:hover{ transform:translateY(-3px) }

.opt .label{ font-weight:700; width:34px; height:34px;
display:inline-flex; align-items:center; justify-content:center;
border-radius:999px; background:#fff6e6; }

.opt.correct{ background:linear-gradient(90deg,
rgba(31,177,106,0.08), rgba(31,177,106,0.02)); border-
color:rgba(31,177,106,0.15) }

.opt.wrong{ background:linear-gradient(90deg, rgba(231,76,60,0.06),
rgba(231,76,60,0.02)); border-color:rgba(231,76,60,0.12) }

button.small{ padding:8px 10px; border-radius:10px; font-weight:700;
cursor:pointer; border:none; }

/* controls row */
.controls{ display:flex; gap:10px; align-items:center; justify-
content:space-between; margin-top:10px; flex-wrap:wrap; }

.left-controls{ display:flex; gap:8px; align-items:center; }

.right-controls{ display:flex; gap:8px; align-items:center; }

/* badges, stickers */
#stickers{ background:#fff9e9; border:3px dashed var(--accent);
padding:8px; border-radius:12px; font-size:14px; }

.floating{ position:fixed; pointer-events:none; z-index:2000; font-
size:22px; animation:floatUp 1.6s ease-out forwards; }

@keyframes floatUp{ 0%{ transform:translateY(0) scale(.9);
opacity:1 } 100%{ transform:translateY(-120px) scale(1.2);
opacity:0 } }

/* hint button */
.hint{ background:#0b74ff; color:#fff }

.mute{ background:#e2e8f0; color:#0b1320 }

/* settings modal / panel */
.overlay{ position:fixed; inset:0; background:rgba(0,0,0,0.35);
display:flex; align-items:center; justify-content:center; z-
index:3000; }

.modal{ width:940px; max-width:95%; max-height:90vh; overflow:auto;
background:white; padding:14px; border-radius:14px; box-shadow:0 20px
60px rgba(2,6,23,0.4); }

.modal h3{ margin-top:0 }
```

```
.row{ display:flex; gap:8px; align-items:center; margin-bottom:8px;
flex-wrap:wrap; }
.field{ flex:1; min-width:180px; }
label.small{ font-size:12px; color:var(--muted); display:block;
margin-bottom:4px }
input[type="text"], textarea, select{
  width:100%; padding:8px; border-radius:8px; border:1px solid #ddd;
font-size:14px;
}
textarea{ min-height:54px; resize:vertical; }

.qcard{ border:1px solid #f0e3c9; padding:8px; border-radius:8px;
margin-bottom:8px; background:#fffef8; display:flex; gap:10px; align-
items:flex-start; }
.q-actions{ display:flex; gap:6px; flex-direction:column; }

/* search bar */
.search{ display:flex; gap:8px; align-items:center; margin-
bottom:8px; }

/* footer */
.footer{ text-align:center; color:var(--muted); margin:20px 0 40px;
font-size:14px; }

/* confetti canvas */
#confettiCanvas{ position:fixed; left:0; top:0; width:100%;
height:100%; pointer-events:none; z-index:2500 }

/* transitions */
.fade{ transition: opacity .25s ease; }
.hidden{ display:none !important; opacity:0 !important }

/* buddy animations */
.buddy{ font-size:36px; display:inline-block;
transition:transform .35s ease, filter .35s; }
.jump{ transform:translateY(-10px) scale(1.1); }
.shake{ animation:shake .6s; }
@keyframes shake{ 10%{ transform:translateX(-4px) }
30%{ transform:translateX(4px) } 50%{ transform:translateX(-3px) }
```

```
70%{ transform:translateX(3px) } 90%{ transform:translateX(-1px) } }

/* results */
.results{ display:flex; flex-direction:column; gap:10px; align-items:center; text-align:center; padding:12px }
.badge{ background:#fff9e9; border:2px solid var(--accent); padding:8px 12px; border-radius:10px; font-weight:700 }

/* accessibility large text toggle (class toggled by settings) */
.large-text{ font-size:18px; }

</style>
</head>
<body>

<header>
  <h1>Right vs Wrong – Kids Learning</h1>
  <div id="settingsIcon" tabindex="0" title="Teacher Settings">⚙</div>
</header>

<main class="app" id="app">
  <!-- LEFT: video + controls -->
  <div class="video-card card" aria-live="polite">
    <div class="video-wrap" id="videoWrap" role="region" aria-label="Scenario video">
      <iframe id="videoFrame" src="" allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture" allowfullscreen title="Scenario video"></iframe>
    </div>

    <div style="padding:12px;">
      <div class="progress" aria-hidden="false">
        <div id="progressText">Question 1 of 5</div>
        <div class="level">
          <div id="levelFill" class="level-fill" aria-hidden="true"></div>
        </div>
        <div id="timerDisplay" aria-live="polite" style="min-width:50px;text-align:right;font-size:13px;color:var(--muted)"></div>
      </div>
    </div>
  </div>
</main>
```

```
</div>

<h2 class="question" id="questionText">Question text</h2>

<div class="options" id="optionsArea" role="list"></div>

<div class="controls" style="margin-top:12px;">
  <div class="left-controls">
    <button class="small hint" id="hintBtn" aria-label="Hint (costs sticker)">💡 Hint</button>
    <div id="stickerBox" style="margin-left:8px;"><span id="stickerCount">Stickers: 0</span></div>
  </div>
  <div class="right-controls">
    <button class="small mute" id="muteBtn" aria-label="Mute/unmute sounds">🔇 </button>
    <button class="small" id="nameBtn" aria-label="Set child name">👤 Name</button>
    <button class="small next-btn" id="nextBtn" aria-label="Next question">Next</button>
  </div>
</div>

<div style="margin-top:10px; display:flex; justify-content:space-between; align-items:center;">
  <div id="message" style="color:var(--muted); font-weight:600"></div>
  <div class="buddy" id="buddy" aria-hidden="true">😊 </div>
</div>
</div>

<!-- RIGHT: settings summary + factbook -->
<div>
  <div class="card" style="margin-bottom:12px;">
    <div style="display:flex; justify-content:space-between; align-items:center;">
      <div>
        <div style="font-weight:700">Game Controls</div>
```

```
        <div style="color:var(--muted); font-size:13px">View game  
settings and progress</div>  
    </div>  
    <div style="text-align:right;">  
        <div style="font-size:13px; color:var(--muted)">Streak:  
<span id="streakCount">0</span></div>  
        <div style="font-size:12px; color:var(--muted)">Best: <span  
id="bestScoreDisplay">0</span></div>  
    </div>  
</div>  
  
<hr style="margin:8px 0">  
  
<div id="factBook" style="margin-top:10px;">  
    <div style="font-weight:700">Fact Book</div>  
    <div id="factList" style="color:var(--muted); font-size:14px;  
min-height:54px; margin-top:6px;">No facts yet – answer correctly to  
collect them.</div>  
    </div>  
  
    <div style="margin-top:12px; display:flex; gap:8px; justify-  
content:space-between;">  
        <button id="openControls" class="small"  
style="background:var(--accent); color:white">View Game  
Controls</button>  
        <button id="resetDefault" class="small"  
style="background:#e74c3c; color:white">  Reset to Default</button>  
    </div>  
    </div>  
</div>  
</main>  
  
<!-- name modal -->  
<div id="nameModal" class="overlay" aria-hidden="false">  
    <div class="modal" role="dialog" aria-modal="true" aria-label="Enter  
your name">  
        <h3>Welcome to Right vs Wrong!</h3>  
        <p>Please enter your name to get started:</p>  
        <div style="margin-top:10px;">
```

```
        <input type="text" id="nameInput" placeholder="Your name"
style="width:100%; padding:10px; border-radius:8px; border:1px solid
#ddd; font-size:16px;" />
    </div>
    <div style="margin-top:15px; text-align:right;">
        <button id="submitName" class="small" style="background:var(--accent); color:white; padding:10px 20px;">Start Playing</button>
    </div>
</div>

<!-- game controls modal -->
<div id="controlsModal" class="overlay hidden" aria-hidden="true">
    <div class="modal" role="dialog" aria-modal="true" aria-label="Game
Controls">
        <h3>Game Controls</h3>

        <div style="display:flex; gap:8px; align-items:center; margin-
bottom:8px;">
            <input type="checkbox" id="shuffleQ" checked>
            <label for="shuffleQ" style="font-size:13px; color:var(--muted)">Shuffle questions each game</label>
        </div>

        <div style="display:flex; gap:8px; align-items:center; margin-
bottom:8px;">
            <input type="checkbox" id="timerToggle">
            <label for="timerToggle" style="font-size:13px; color:var(--muted)">Enable 10s timer for bonus</label>
        </div>

        <div style="margin-top:15px; text-align:right;">
            <button id="closeControls" class="small" style="background:var(--accent); color:white; padding:10px 20px;">Close</button>
        </div>
    </div>
</div>

<!-- settings modal (password + management) -->
```

```
<div id="settingsModal" class="overlay hidden" aria-hidden="true">
  <div class="modal" role="dialog" aria-modal="true" aria-
label="Teacher settings">
    <h3>Teacher Settings – Secure</h3>

    <!-- password box / toggle -->
    <div id="pwBlock" class="row" style="align-items:center;">
      <div style="flex:1">
        <label class="small">Enter teacher code</label>
        <input type="password" id="pwInput" placeholder="Enter code"
aria-label="Teacher password" />
      </div>
      <div>
        <label class="small">Show</label>
        <button id="pwToggle" class="small">○</button>
      </div>
      <div>
        <button id="pwSubmit" class="small" style="background:var(--accent); color:white">Unlock</button>
      </div>
    </div>
  </div>

  <hr>

  <div id="settingsContent" class="hidden">
    <div style="display:flex; gap:8px; align-items:center; margin-
bottom:8px;">
      <input id="searchQ" placeholder="Search questions..." />
      <button id="importBtn" class="small">Import JSON</button>
      <input type="file" id="importFile" accept=".json"
style="display:none"/>
      <button id="exportBtn" class="small">Export JSON</button>
    </div>
  </div>

  <div id="questionsList"></div>

  <div style="margin-top:8px; display:flex; gap:8px; justify-
content:space-between;">
    <div>
```

```
        <button id="addQBtn" class="small">+ Add Question</button>
        <button id="saveQBtn" class="small" style="background:var(--accent); color:white">Save</button>
    </div>
    <div>
        <button id="closeSettingsBtn" class="small">Close</button>
    </div>
</div>

</div>
</div>

<!-- confetti canvas -->
<canvas id="confettiCanvas"></canvas>

<footer>© Right vs Wrong – built for curious kids</footer>

<!-- sounds -->
<audio id="s_correct"
src="https://actions.google.com/sounds/v1/cartoon/pop.ogg"
preload="auto"></audio>
<audio id="s_wrong"
src="https://actions.google.com/sounds/v1/cartoon/wood_plank_flicks.ogg"
preload="auto"></audio>
<audio id="s_end"
src="https://actions.google.com/sounds/v1/cartoon/slide_whistle_to_drum_hit.ogg"
preload="auto"></audio>

<script>
/* =====
   App Data & Defaults
===== */
const DEFAULT_DATA = [
  { id: "q1", difficulty:"easy",
video:"https://www.youtube.com/embed/_7R4SflsINU",
question:"What did the kid do wrong after finishing their snack?", options:["Threw the wrapper on the ground","Put the wrapper in the
```

```

bin", "Kept the wrapper in their pocket"],
    correct:0, hint:"Think about the wrapper", fact:"Putting trash in
a bin keeps places clean." },
{ id: "q2", difficulty:"easy",
video:"https://www.youtube.com/embed/dQw4w9WgXcQ",
    question:"What mistake did the kid make while washing hands?",
    options:["Left the water running","Turned off the tap before
getting soap","Used too much soap"],
    correct:0, hint:"Water matters", fact:"Turning taps off saves
water." },
{ id: "q3", difficulty:"medium",
video:"https://www.youtube.com/embed/M7lc1UVf-VE",
    question:"What did the kid forget to do before leaving their
room?", 
    options:["Forgot to turn off the lights","Closed the window","Put
away their toys"],
    correct:0, hint:"Think energy", fact:"Turning off lights saves
electricity." },
{ id: "q4", difficulty:"easy",
video:"https://www.youtube.com/embed/ScMzIvxBSi4",
    question:"What should you do before crossing the road?", 
    options:["Look both ways","Run quickly","Close your eyes"],
    correct:0, hint:"Be safe", fact:"Always look both ways before
crossing." },
{ id: "q5", difficulty:"medium",
video:"https://www.youtube.com/embed/tgbNymZ7vqY",
    question:"What is wrong if you don't wash fruits?", 
    options:["They can have dirt or germs","They taste bad","They
become too soft"],
    correct:0, hint:"Think hygiene", fact:"Washing fruit removes dirt
and germs." }
];

```

```

let DATA = JSON.parse(localStorage.getItem("rvs_quiz")) ||
DEFAULT_DATA.slice();
let SETTINGS = JSON.parse(localStorage.getItem("rvs_settings")) || {
  accent:
getComputedStyle(document.documentElement).getPropertyValue('--
accent').trim() || "#ffb02e",

```

```
buddySet: "faces",
font: "Comic Sans MS, Poppins, sans-serif",
shuffle: true,
timerEnabled: false
};

let state = {
  order: [], // randomized question order
  idx: 0,
  score: 0,
  stickers: JSON.parse(localStorage.getItem("rvs_stickers")) || [],
  streak: parseInt(localStorage.getItem("rvs_streak"))||0,
  best: parseInt(localStorage.getItem("rvs_best"))||0,
  timer: null,
  timerSec: 10,
  mute: false,
  childName: localStorage.getItem("rvs_childName") || "",
  hintPenalty: 1, // subtract 1 point when using hint
  shuffleQuestions: SETTINGS.shuffle,
  lastPlayed: JSON.parse(localStorage.getItem("rvs_lastPlayed")) || {}
};

/* DOM refs */
const videoFrame = document.getElementById("videoFrame");
const questionText = document.getElementById("questionText");
const optionsArea = document.getElementById("optionsArea");
const progressText = document.getElementById("progressText");
const levelFill = document.getElementById("levelFill");
const nextBtn = document.getElementById("nextBtn");
const hintBtn = document.getElementById("hintBtn");
const stickerCount = document.getElementById("stickerCount");
const messageDiv = document.getElementById("message");
const buddyEl = document.getElementById("buddy");
const timerDisplay = document.getElementById("timerDisplay");
const settingsIcon = document.getElementById("settingsIcon");
const settingsModal = document.getElementById("settingsModal");
const pwInput = document.getElementById("pwInput");
const pwToggle = document.getElementById("pwToggle");
const pwSubmit = document.getElementById("pwSubmit");
```

```
const settingsContent = document.getElementById("settingsContent");
const questionsList = document.getElementById("questionsList");
const searchQ = document.getElementById("searchQ");
const importFile = document.getElementById("importFile");
const confettiCanvas = document.getElementById("confettiCanvas");
const s_correct = document.getElementById("s_correct");
const s_wrong = document.getElementById("s_wrong");
const s_end = document.getElementById("s_end");
const resetDefaultBtn = document.getElementById("resetDefault");
const muteBtn = document.getElementById("muteBtn");
const shuffleQ = document.getElementById("shuffleQ");
const timerToggle = document.getElementById("timerToggle");
const nameBtn = document.getElementById("nameBtn");
const factList = document.getElementById("factList");
const streakCount = document.getElementById("streakCount");
const bestScoreDisplay = document.getElementById("bestScoreDisplay");
const nameModal = document.getElementById("nameModal");
const nameInput = document.getElementById("nameInput");
const submitName = document.getElementById("submitName");
const controlsModal = document.getElementById("controlsModal");
const openControls = document.getElementById("openControls");
const closeControls = document.getElementById("closeControls");

/* apply saved theme to UI */
function applySavedTheme(){
    document.documentElement.style.setProperty("--accent",
SETTINGS.accent||"#ffb02e");
    document.documentElement.style.setProperty("--accent-dark",
shadeColor(SETTINGS.accent||"#ffb02e",-10));
    document.documentElement.style.setProperty("--font-main",
SETTINGS.font||"Comic Sans MS, Poppins, sans-serif");
    document.body.style.fontFamily = SETTINGS.font;
}
applySavedTheme();

/* small util: color shade for accent dark */
function shadeColor(color, percent) {
// color in #rrggbb
const num = parseInt(color.slice(1),16);
```

```

const r = (num >> 16) + percent;
const g = (num >> 8 & 0x00FF) + percent;
const b = (num & 0x0000FF) + percent;
const newR = Math.min(255, Math.max(0, r));
const newG = Math.min(255, Math.max(0, g));
const newB = Math.min(255, Math.max(0, b));
return `#${(newR<<16 | newG<<8 |
newB).toString(16).padStart(6,'0')}`;
}

/*
=====
Security: password hashing via Web Crypto
=====
*/
const CODE_KEY = "rvs_code_hash";
async function hashCode(str){
    const enc = new TextEncoder().encode(str);
    const hashBuffer = await crypto.subtle.digest('SHA-256', enc);
    const hashArray = Array.from(new Uint8Array(hashBuffer));
    const hashHex =
hashArray.map(b=>b.toString(16).padStart(2,'0')).join('');
    return hashHex;
}
// on first load: if no code stored, store default hashed "67"
(async()=>{
    if(!localStorage.getItem(CODE_KEY)){
        const h = await hashCode("67");
        localStorage.setItem(CODE_KEY, h);
    }
})();

/* password modal handlers */
pwToggle.addEventListener("click", ()=>{
    if(pwInput.type === "password") pwInput.type = "text"; else
pwInput.type = "password";
});
pwSubmit.addEventListener("click", async ()=>{
    const attempt = pwInput.value.trim();
    if(!attempt) return alert("Enter code");
    const hashed = await hashCode(attempt);

```

```

if(hashed === localStorage.getItem(CODE_KEY)){
    // unlocked
    settingsContent.classList.remove("hidden");
    document.getElementById("pwBlock").classList.add("hidden");
    buildSettingsList();
} else {
    alert("Wrong code");
}
});

/* open settings from header icon */
settingsIcon.addEventListener("click", ()=>openSettingsModal());
function openSettingsModal(){
    settingsModal.classList.remove("hidden");
    settingsModal.style.display = "flex";
    pwInput.value = "";
    settingsContent.classList.add("hidden");
    document.getElementById("pwBlock").classList.remove("hidden");
    buildSettingsList();
}
document.getElementById("closeSettingsBtn").addEventListener("click",
()=>{
    settingsModal.classList.add("hidden");
});

/*
=====
SETTINGS: build question editor list
=====
*/
function buildSettingsList(filter=""){
    questionsList.innerHTML = "";
    DATA.forEach((q,i)=>{
        if(filter && !(q.question+q.options.join(" ") +
(q.fact|| "") ).toLowerCase().includes(filter.toLowerCase())) return;
        const card = document.createElement("div");
        card.className = "qcard";
        card.innerHTML = `
            <div style="flex:1">
                <label class="small">Video URL</label>
                <input type="text" class="sq_vid"

```

```

value="${escapeHtml(q.video)}"/>
    <label class="small">Question</label>
    <input type="text" class="sq_q"
value="${escapeHtml(q.question)}"/>
    <label class="small">Options (comma separated)</label>
    <textarea
class="sq_opts">${escapeHtml(q.options.join(","))}</textarea>
    <label class="small">Correct Index (0-based)</label>
    <input type="number" class="sq_correct" value="${q.correct}"
min="0" style="width:80px"/>
    <label class="small">Hint</label>
    <input type="text" class="sq_hint"
value="${escapeHtml(q.hint||"")}" />
    <label class="small">Fact</label>
    <input type="text" class="sq_fact"
value="${escapeHtml(q.fact||"")}" />
</div>
<div class="q-actions">
    <button class="small" data-action="up"> </button>
    <button class="small" data-action="down"> </button>
    <button class="small" data-action="remove"
style="background:#e74c3c;color:white">Remove</button>
</div>
` ;
// attach handlers with closure index
const up = card.querySelector('[data-action="up"]');
const down = card.querySelector('[data-action="down"]');
const remove = card.querySelector('[data-action="remove"]');
up.addEventListener("click", ()=>{ if(i>0){ [DATA[i-1],DATA[i]]=[DATA[i],DATA[i-1]]; buildSettingsList(filter); }});
down.addEventListener("click", ()=>{ if(i<DATA.length-1){ [DATA[i+1],DATA[i]]=[DATA[i],DATA[i+1]]; buildSettingsList(filter); }});
remove.addEventListener("click", ()=>{ if(confirm("Remove this question?")){ DATA.splice(i,1); buildSettingsList(filter); } });
questionsList.appendChild(card);
});
}
searchQ && searchQ.addEventListener("input", ()=>

```

```

buildSettingsList(searchQ.value));

function addQuestion(){
    DATA.push({id:"q"+Date.now(), difficulty:"easy", video:"",
question:"New question", options:["Option A","Option B","Option C"],
correct:0, hint:"", fact:""});
    buildSettingsList();
}
document.getElementById("addQBtn").addEventListener("click",
addQuestion);

/* save settings / export / import */
document.getElementById("saveQBtn").addEventListener("click", ()=> {
    // read back UI into DATA
    const cards = questionsList.querySelectorAll('.qcard');
    const newData=[];
    cards.forEach(card=>{
        const vid = card.querySelector('.sq_vid').value.trim();
        const qtxt = card.querySelector('.sq_q').value.trim();
        const optsRaw = card.querySelector('.sq_opts').value.trim();
        const correct =
parseInt(card.querySelector('.sq_correct').value)||0;
        const hint = card.querySelector('.sq_hint').value.trim();
        const fact = card.querySelector('.sq_fact').value.trim();
        const opts = optsRaw.split(',').map(s=>s.trim()).filter(Boolean);
        if(qtxt && opts.length>=2){
            newData.push({ id:"q"+Date.now()+Math.random(),
difficulty:"easy", video:vid, question:qtxt, options:opts, correct:
Math.min(correct, opts.length-1), hint, fact });
        }
    });
    if(newData.length>0){
        DATA = newData;
        localStorage.setItem("rvs_quiz", JSON.stringify(DATA));
        alert("Saved. New question set will be used.");
        settingsModal.classList.add("hidden");
        resetGameToStart();
    } else {
        alert("No valid questions found in editor.");
    }
})

```

```

    }
});

document.getElementById("exportBtn").addEventListener("click", ()=>{
    const blob = new Blob([JSON.stringify(DATA, null,
2)],{type:"application/json"});
    const url = URL.createObjectURL(blob);
    const a = document.createElement("a");
    a.href = url; a.download = "rvs_quiz_export.json"; a.click();
    URL.revokeObjectURL(url);
});
document.getElementById("importBtn").addEventListener("click", ()=>
importFile.click());
importFile.addEventListener("change", (e)=> {
    const file = e.target.files[0];
    if(!file) return;
    const fr = new FileReader();
    fr.onload = ()=> {
        try{ const parsed = JSON.parse(fr.result);
if(Array.isArray(parsed)){ DATA = parsed;
localStorage.setItem("rvs_quiz", JSON.stringify(DATA));
alert("Imported."); buildSettingsList(); resetGameToStart(); } else
alert("JSON is not an array.");
        catch(err){ alert("Invalid JSON file."); }
    };
    fr.readAsText(file);
});
}

document.getElementById("resetDefault").addEventListener("click",
()=>{
    if(confirm("Reset to default questions? This will clear custom
questions.")){
        DATA = DEFAULT_DATA.slice();
        localStorage.removeItem("rvs_quiz");
        resetGameToStart();
    }
});
/*
=====

```

```

Game helper utilities
=====
function escapeHtml(s){ return
s?.replaceAll?.( "&","&").replaceAll?.( "<","<").replaceAll?.( ">"
,">") ?? s; }

function shuffle(arr){
  const a = arr.slice();
  for(let i=a.length-1;i>0;i--){
    const j = Math.floor(Math.random()*(i+1));
    [a[i],a[j]] = [a[j],a[i]];
  }
  return a;
}

/*
=====
Game initialization & main loop
=====
*/
function buildPool(){
  let pool = DATA.slice();
  if(document.getElementById("shuffleQ").checked) pool =
shuffle(pool);
  state.order = pool;
  state.idx = 0;
  state.score = 0;
  state.timerSec = 10;
  state.stickers = JSON.parse(localStorage.getItem("rvs_stickers")) ||
[];
  document.getElementById("stickerCount").textContent = "Stickers:
"+state.stickers.length;
}
function loadQuestion(){
  if(state.idx >= state.order.length) { showResults(); return; }
  const q = state.order[state.idx];
  // load video and question
  videoFrame.src = q.video;
  questionText.textContent = q.question;
  progressText.textContent = `Question ${state.idx+1} of
${state.order.length}`;
```

```

// build options (shuffled)
optionsArea.innerHTML = "";
const shuffled = shuffle(q.options.map((opt, i)=>({opt,i})));
shuffled.forEach((o, aidx)=>{
    const div = document.createElement("div");
    div.className = "opt";
    div.setAttribute("role","listitem");
    div.innerHTML = `<div
class="label">${String.fromCharCode(65+aidx)}</div><div
style="flex:1">${o.opt}</div>`;
    div.addEventListener("click", ()=> handleAnswer(div, q, o.i));
    optionsArea.appendChild(div);
});
// reset UI
nextBtn.classList.remove("show");
messageDiv.textContent = "";
buddySetIdle();
updateLevelBar();
// timer
clearTimer();
if(document.getElementById("timerToggle").checked){
    startTimer();
} else { timerDisplay.textContent = ""; }
// save last played
localStorage.setItem("rvs_lastPlayed", JSON.stringify({ idx:
state.idx, timestamp: Date.now() }));
}

/* visual level bar */
function updateLevelBar(){
    const pct = Math.round((state.idx / Math.max(1, state.order.length))
* 100);
    levelFill.style.width = pct + "%";
}

/* answer handling */
function handleAnswer(div, q, chosenIndex){
// disable all options
const all = optionsArea.querySelectorAll(".opt");

```

```
all.forEach(o=>o.style.pointerEvents="none");
clearTimer();
const correct = q.correct;
if(chosenIndex === correct){
    div.classList.add("correct");
    if(!state.mute) s_correct.play();
    // sticker reward: random sticker emoji
    const newSticker = randomSticker();
    state.stickers.push(newSticker);
    showFloatingSticker(newSticker);
    // collect fact into factBook (only on correct)
    addFact(q.fact);
    state.score += 1;
    // buddy animation positive
    buddyCelebrate();
    messageDiv.textContent = "Great job! " + (q.fact || "");
    state.streak = (state.streak || 0) + 1;
} else {
    div.classList.add("wrong");
    // reveal correct option visually
    all.forEach((el, idx) => {
        const txt = el.textContent.trim();
        // find which one matches q.options[correct]
        if(el.textContent.includes(q.options[correct]))
            el.classList.add("correct");
    });
    if(!state.mute) s_wrong.play();
    messageDiv.textContent = "Oops! " + (q.fact || "");
    state.streak = 0;
}
// show next
nextBtn.classList.add("show");
// save stickers and streak/best
localStorage.setItem("rvs_stickers",
JSON.stringify(state.stickers));
localStorage.setItem("rvs_streak", state.streak);
if(state.score > state.best){ state.best = state.score;
localStorage.setItem("rvs_best", state.best); }
// update counts display
```

```

document.getElementById("stickerCount").textContent = "Stickers: " +
state.stickers.length;
streakCount.textContent = state.streak;
bestScoreDisplay.textContent = state.best;
}

/* next button */
nextBtn.addEventListener("click", ()=>{
  state.idx++;
  if(state.idx < state.order.length) {
    updateLevelBarAnimate();
    loadQuestion();
  } else {
    showResults();
  }
});

/* hint button */
hintBtn.addEventListener("click", ()=>{
  const q = state.order[state.idx];
  if(!q) return;
  // penalty: deduct one point if available (but not below 0)
  if(state.score > 0){ state.score = Math.max(0, state.score -
state.hintPenalty); }
  messageDiv.textContent = "💡 " + (q.hint || "Think carefully!");
  // deduct sticker if any exist (choose: cost 1 sticker if present)
  if(state.stickers.length>0) state.stickers.pop();
  localStorage.setItem("rvs_stickers",
JSON.stringify(state.stickers));
  document.getElementById("stickerCount").textContent = "Stickers: " +
state.stickers.length;
});

/* show results with confetti and fact book */
function showResults(){
  // confetti
  runConfetti();
  if(!state.mute) s_end.play();
  // show results modal-like area: reuse right side results area
}

```

```

document.getElementById("app").classList.add("fade");

setTimeout(()=>document.getElementById("app").classList.remove("fade")
, 300);

// assemble results in factList and results area
// quick results display (alert-like)
const name = state.childName || "Friend";
const finalMsg = `${name}, you got ${state.score} out of
${state.order.length}!`;
alert(finalMsg + "\nYour stickers: " + state.stickers.join(" ") +
"\nFacts collected in fact book.");
// save best
if(state.score > state.best) { state.best = state.score;
localStorage.setItem("rvs_best", state.best); }
bestScoreDisplay.textContent = state.best;
// update fact list already maintained
// reset for next play
resetGameToStart();
}

/* reset game to start */
function resetGameToStart(){
  buildPool();
  renderFactBook();
  loadQuestion();
}

/* initial build & start */
document.getElementById("shuffleQ").addEventListener("change", ()=>
SETTINGS.shuffle = document.getElementById("shuffleQ").checked );
document.getElementById("timerToggle").addEventListener("change", ()=>
{ /* nothing immediate */ });
document.getElementById("muteBtn").addEventListener("click", ()=>
{ state.mute = !state.mute; muteBtn.textContent = state.mute ? '🔇' :
'🔈'; });

document.getElementById("nameBtn").addEventListener("click", ()=>{
  const n = prompt("Enter child's name (for results)", "");
```

```

if(n){ state.childName = n; localStorage.setItem("rvs_childName", n); alert("Saved name: "+n); }
});

/* name modal handlers */
submitName.addEventListener("click", ()=>{
  const name = nameInput.value.trim();
  if(name){
    state.childName = name;
    localStorage.setItem("rvs_childName", name);
    nameModal.classList.add("hidden");
    loadQuestion();
  } else {
    alert("Please enter your name");
  }
});

/* game controls modal handlers */
openControls.addEventListener("click", ()=>{
  controlsModal.classList.remove("hidden");
  controlsModal.style.display = "flex";
});

closeControls.addEventListener("click", ()=>{
  controlsModal.classList.add("hidden");
});

/* settings header icon opens modal but triggers pw flow earlier */
settingsIcon.addEventListener("dblclick", ()=> openSettingsModal());
// doubleclick alternate

/* random sticker emoji generator */
const STICKERS =
["⭐️","🎈","🌟","🌈","🍀","🦄","⭐️","🎀","🏅","🍎","👾"];
function randomSticker(){ return
STICKERS[Math.floor(Math.random()*STICKERS.length)]; }
function showFloatingSticker(emoji){
  const el = document.createElement("div");
  el.className = "floating";

```

```

el.style.left = (30 + Math.random()*60) + "vw";
el.textContent = emoji;
document.body.appendChild(el);
setTimeout(()=> el.remove(), 1600);
}

/* buddy animations */
function buddyCelebrate(){
  buddyEl.classList.add("jump");
  setTimeout(()=> buddyEl.classList.remove("jump"), 600);
}
function buddySetIdle(){ buddyEl.classList.remove("jump", "shake"); }

/* timer */
function startTimer(){
  let sec = 10;
  timerDisplay.textContent = sec + "s";
  state.timer = setInterval(()=>{
    sec--;
    timerDisplay.textContent = sec + "s";
    if(sec<=0){
      clearTimer();
      // time's up: auto mark wrong and reveal
      messageDiv.textContent = "Time's up!";
      // simulate picking nothing: reveal correct
      const all = optionsArea.querySelectorAll(".opt");
      all.forEach(el=>{

        if(el.textContent.includes(state.order[state.idx].options[state.order[state.idx].correct])) el.classList.add("correct");
          el.style.pointerEvents = "none";
      });
      if(!state.mute) s_wrong.play();
      nextBtn.classList.add("show");
    }
  },1000);
}

function clearTimer(){ if(state.timer){ clearInterval(state.timer);
state.timer=null; } timerDisplay.textContent = ""; }

```

```
/* confetti implementation on canvas (simple) */
function runConfetti(){
    const canvas = confettiCanvas;
    const ctx = canvas.getContext('2d');
    canvas.width = window.innerWidth;
    canvas.height = window.innerHeight;
    const pieces = [];
    for(let i=0;i<80;i++){
        pieces.push({
            x: Math.random()*canvas.width,
            y: Math.random()*-canvas.height,
            r: 6+Math.random()*8,
            color: `hsl(${Math.random()*360},70%,60%)`,
            vx: -2 + Math.random()*4,
            vy: 2 + Math.random()*6,
            rot: Math.random()*360
        });
    }
    let t=0;
    function frame(){
        t++;
        ctx.clearRect(0,0,canvas.width,canvas.height);
        for(const p of pieces){
            p.x += p.vx; p.y += p.vy; p.rot += 4;
            ctx.save();
            ctx.fillStyle = p.color;
            ctx.translate(p.x, p.y);
            ctx.rotate(p.rot * Math.PI/180);
            ctx.fillRect(-p.r/2, -p.r/2, p.r, p.r);
            ctx.restore();
        }
        if(t<120) requestAnimationFrame(frame);
        else ctx.clearRect(0,0,canvas.width,canvas.height);
    }
    frame();
}

/* fact book */
```

```
function addFact(f){
  if(!f) return;
  const facts = JSON.parse(localStorage.getItem("rvs_facts")||"[]");
  if(!facts.includes(f)) facts.push(f);
  localStorage.setItem("rvs_facts", JSON.stringify(facts));
  renderFactBook();
}

function renderFactBook(){
  const facts = JSON.parse(localStorage.getItem("rvs_facts")||"[]");
  if(facts.length==0) factList.textContent = "No facts yet – answer correctly to collect them.";
  else factList.innerHTML = facts.map(f=>`

${escapeHtml(f)}

`).join('');
}

/* UI: show settings modal from header icon (opens password UI) */
settingsIcon.addEventListener("click", ()=> {
  settingsModal.classList.remove("hidden");
});
document.getElementById("closeSettingsBtn").addEventListener("click", ()=> settingsModal.classList.add("hidden"));

/* reset game / start */
document.addEventListener("DOMContentLoaded", ()=>{
  // Check if name is already saved
  if(!state.childName) {
    // Show name modal
    nameModal.style.display = "flex";
  } else {
    // Name already exists, start game
    buildPool();
    renderFactBook();
    loadQuestion();
  }
  // load persisted theme
  document.getElementById("stickerCount").textContent = "Stickers: " + (state.stickers.length||0);
});
```

```

/* helper: reset everything and start new */
function resetAll(){
    state.idx = 0; state.score = 0; state.stickers = [];
localStorage.removeItem("rvs_stickers");
localStorage.removeItem("rvs_streak");
}

/* reset to start after game or save */
function resetGameToStart(){
    state.idx = 0; state.score = 0;
    buildPool();
    loadQuestion();
}

/* timer/bonus ui animation for level bar */
function updateLevelBarAnimate(){ levelFill.style.transition =
'width .35s ease-out'; updateLevelBar(); setTimeout(()=>
levelFill.style.transition='width .35s ease-out', 400); }

/* small helpers */
function escapeHtml(s){ if(!s) return ""; return
s.replaceAll("<","&lt;").replaceAll(">","&gt;"); }

/* end of script */
</script>
</body>
</html>

```

I've removed all preview functionality from the game:

1. **Removed the preview button** from the controls section
2. **Removed the preview modal** and its HTML structure
3. **Removed all preview-related JavaScript functions** including `previewQuestion()`, `previewCurrentQuestion()`, and `closePreview()`
4. **Removed preview event listeners** and references
5. **Removed the preview button** from the question editor in the settings modal

The game now has a cleaner interface without the preview feature, while maintaining all other functionality including the name input modal, simplified game controls, and the fact book.

## USER

add a results menu end menu

## ASSISTANT

I'll add a results menu that appears at the end of the game. Here's the updated version:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width,initial-scale=1" />
<title>Right vs Wrong – Kids Learning Game</title>
<style>
:root{
    --accent: #ffb02e;
    --accent-dark: #ff9b00;
    --bg: #f7fbff;
    --green: #1fb16a;
    --red: #e74c3c;
    --muted: #6b7280;
    --card: #fff;
    --shadow: 0 10px 25px rgba(15,23,42,0.08);
    --radius: 14px;
    --font-main: "Comic Sans MS", "Poppins", sans-serif;
}
*{box-sizing:border-box}
body{
    margin:0; font-family:var(--font-main); background:linear-gradient(180deg,var(--bg),#fff);
    color:#0f172a; -webkit-font-smoothing:antialiased;
}

/* header */
```

```
header{
    background:linear-gradient(90deg,var(--accent),var(--accent-dark));
    color:white; padding:14px 16px; display:flex; align-items:center;
justify-content:center;
    position:relative; box-shadow:var(--shadow);
}
header h1{ margin:0; font-size:18px; letter-spacing:0.3px; }
#settingsIcon{ position:absolute; right:12px; top:8px; width:40px;
height:40px; border-radius:999px;
    display:flex; align-items:center; justify-content:center;
background:rgba(255,255,255,0.12); cursor:pointer; }
#settingsIcon:focus{outline:3px solid rgba(255,255,255,0.18) }

/* layout */
main.app{ max-width:1100px; margin:18px auto; padding:12px;
display:grid; grid-template-columns:1fr 420px; gap:20px;
transition:opacity .25s ease; }
@media(max-width:900px){ main.app{ grid-template-columns:1fr;
padding:10px } }

/* video card */
.video-card{ background:var(--card); border-radius:var(--radius);
overflow:hidden; box-shadow:var(--shadow); display:flex; flex-
direction:column; }
.video-wrap{ position:relative; padding-top:56.25%; } /* 16:9 */
.video-wrap iframe{ position:absolute; inset:0; width:100%;
height:100%; border:0; }

/* question card */
.card{ background:linear-gradient(180deg, #fff, #ffffef9); border-
radius:var(--radius); padding:14px; box-shadow:var(--shadow);
border:4px solid rgba(255,224,178,0.6); }
.progress{ color:var(--muted); font-size:13px; margin-bottom:8px;
display:flex; align-items:center; justify-content:space-between;
gap:8px; }
.level{ flex:1; height:10px; background:#eef2f7; border-
radius:999px; overflow:hidden; margin-left:8px; margin-right:8px; }
.level-fill{ height:100%; width:0%; background:linear-
```

```
gradient(90deg,var(--accent),var(--accent-dark));
transition:width .35s ease-out; }

h2.question{ font-size:18px; margin:6px 0 12px 0; text-align:center;
color:#0c1720; }

.options{ display:flex; flex-direction:column; gap:10px; }
.opt{ display:flex; gap:12px; align-items:center; padding:10px 12px;
background:var(--card); border-radius:10px; cursor:pointer; border:2px
solid #fff6e6;
box-shadow:0 8px 18px rgba(15,23,42,0.04);
transition:transform .12s ease, box-shadow .12s ease; font-
size:15px; }
.opt:hover{ transform:translateY(-3px) }
.opt .label{ font-weight:700; width:34px; height:34px;
display:inline-flex; align-items:center; justify-content:center;
border-radius:999px; background:#fff6e6; }
.opt.correct{ background:linear-gradient(90deg,
rgba(31,177,106,0.08), rgba(31,177,106,0.02)); border-
color:rgba(31,177,106,0.15) }
.opt.wrong{ background:linear-gradient(90deg, rgba(231,76,60,0.06),
rgba(231,76,60,0.02)); border-color:rgba(231,76,60,0.12) }
button.small{ padding:8px 10px; border-radius:10px; font-weight:700;
cursor:pointer; border:none; }

/* controls row */
.controls{ display:flex; gap:10px; align-items:center; justify-
content:space-between; margin-top:10px; flex-wrap:wrap; }
.left-controls{ display:flex; gap:8px; align-items:center; }
.right-controls{ display:flex; gap:8px; align-items:center; }

/* badges, stickers */
#stickers{ background:#fff9e9; border:3px dashed var(--accent);
padding:8px; border-radius:12px; font-size:14px; }
.floating{ position:fixed; pointer-events:none; z-index:2000; font-
size:22px; animation:floatUp 1.6s ease-out forwards; }
@keyframes floatUp{ 0%{ transform:translateY(0) scale(.9);
opacity:1 } 100%{ transform:translateY(-120px) scale(1.2);
opacity:0 } }
```

```
/* hint button */
.hint{ background:#0b74ff; color:#fff }
.mute{ background:#e2e8f0; color:#0b1320 }

/* settings modal / panel */
.overlay{ position:fixed; inset:0; background:rgba(0,0,0,0.35);
display:flex; align-items:center; justify-content:center; z-
index:3000; }
.modal{ width:940px; max-width:95%; max-height:90vh; overflow:auto;
background:white; padding:14px; border-radius:14px; box-shadow:0 20px
60px rgba(2,6,23,0.4); }
.modal h3{ margin-top:0 }
.row{ display:flex; gap:8px; align-items:center; margin-bottom:8px;
flex-wrap:wrap; }
.field{ flex:1; min-width:180px; }
label.small{ font-size:12px; color:var(--muted); display:block;
margin-bottom:4px }
input[type="text"], textarea, select{
width:100%; padding:8px; border-radius:8px; border:1px solid #ddd;
font-size:14px;
}
textarea{ min-height:54px; resize:vertical; }

.qcard{ border:1px solid #f0e3c9; padding:8px; border-radius:8px;
margin-bottom:8px; background:#fffef8; display:flex; gap:10px; align-
items:flex-start; }
.q-actions{ display:flex; gap:6px; flex-direction:column; }

/* search bar */
.search{ display:flex; gap:8px; align-items:center; margin-
bottom:8px; }

/* footer */
.footer{ text-align:center; color:var(--muted); margin:20px 0 40px;
font-size:14px; }

/* confetti canvas */
#confettiCanvas{ position:fixed; left:0; top:0; width:100%;
```

```
height:100%; pointer-events:none; z-index:2500 }

/* transitions */
.fade{ transition: opacity .25s ease; }
.hidden{ display:none !important; opacity:0 !important }

/* buddy animations */
.buddy{ font-size:36px; display:inline-block;
transition:transform .35s ease, filter .35s; }
.jump{ transform:translateY(-10px) scale(1.1); }
.shake{ animation:shake .6s; }
@keyframes shake{ 10%{ transform:translateX(-4px) }
30%{ transform:translateX(4px) } 50%{ transform:translateX(-3px) }
70%{ transform:translateX(3px) } 90%{ transform:translateX(-1px) } }

/* results */
.results{ display:flex; flex-direction:column; gap:10px; align-items:center; text-align:center; padding:12px }
.badge{ background:#fff9e9; border:2px solid var(--accent);
padding:8px 12px; border-radius:10px; font-weight:700 }

/* results modal */
.results-modal {
  background: white;
  border-radius: 20px;
  padding: 30px;
  max-width: 500px;
  width: 90%;
  text-align: center;
  box-shadow: 0 20px 60px rgba(0,0,0,0.3);
  position: relative;
}

.results-modal h2 {
  color: var(--accent);
  font-size: 32px;
  margin-bottom: 20px;
}
```

```
.results-score {  
    font-size: 48px;  
    font-weight: bold;  
    color: var(--green);  
    margin: 20px 0;  
}  
  
.results-message {  
    font-size: 18px;  
    color: var(--muted);  
    margin-bottom: 20px;  
}  
  
.results-stickers {  
    font-size: 24px;  
    margin: 15px 0;  
}  
  
.results-facts {  
    background: #f8f9fa;  
    border-radius: 10px;  
    padding: 15px;  
    margin: 20px 0;  
    text-align: left;  
    max-height: 200px;  
    overflow-y: auto;  
}  
  
.results-facts h4 {  
    margin-top: 0;  
    color: var(--accent);  
}  
  
.results-facts ul {  
    margin: 10px 0;  
    padding-left: 20px;  
}  
  
.results-facts li {
```

```
margin: 8px 0;
color: var(--muted);
}

.results-buttons {
  display: flex;
  gap: 15px;
  justify-content: center;
  margin-top: 25px;
}

.results-buttons button {
  padding: 12px 25px;
  border-radius: 10px;
  font-weight: bold;
  cursor: pointer;
  border: none;
  font-size: 16px;
  transition: transform 0.2s;
}

.results-buttons button:hover {
  transform: translateY(-2px);
}

.btn-primary {
  background: var(--accent);
  color: white;
}

.btn-secondary {
  background: #e2e8f0;
  color: var(--muted);
}

/* accessibility large text toggle (class toggled by settings) */
.large-text{ font-size:18px; }

</style>
</head>
```

```
<body>

<header>
  <h1>Right vs Wrong – Kids Learning</h1>
  <div id="settingsIcon" tabindex="0" title="Teacher
Settings">⚙</div>
</header>

<main class="app" id="app">
  <!-- LEFT: video + controls -->
  <div class="video-card card" aria-live="polite">
    <div class="video-wrap" id="videoWrap" role="region" aria-
label="Scenario video">
      <iframe id="videoFrame" src="" allow="accelerometer; autoplay;
clipboard-write; encrypted-media; gyroscope; picture-in-picture"
allowfullscreen title="Scenario video"></iframe>
    </div>

    <div style="padding:12px;">
      <div class="progress" aria-hidden="false">
        <div id="progressText">Question 1 of 5</div>
        <div class="level">
          <div id="levelFill" class="level-fill" aria-
hidden="true"></div>
        </div>
        <div id="timerDisplay" aria-live="polite" style="min-
width:50px;text-align:right;font-size:13px;color:var(--muted)"></div>
      </div>
    </div>

    <h2 class="question" id="questionText">Question text</h2>

    <div class="options" id="optionsArea" role="list"></div>

    <div class="controls" style="margin-top:12px;">
      <div class="left-controls">
        <button class="small hint" id="hintBtn" aria-label="Hint
(costs sticker)">💡 Hint</button>
        <div id="stickerBox" style="margin-left:8px;"><span
id="stickerCount">Stickers: 0</span></div>
      </div>
    </div>
  </div>
</main>
```

```
</div>
<div class="right-controls">
    <button class="small mute" id="muteBtn" aria-
label="Mute/unmute sounds">🔇 </button>
    <button class="small" id="nameBtn" aria-label="Set child
name">👤 Name</button>
    <button class="small next-btn" id="nextBtn" aria-label="Next
question">Next</button>
</div>
</div>

<div style="margin-top:10px; display:flex; justify-
content:space-between; align-items:center;">
    <div id="message" style="color:var(--muted); font-
weight:600"></div>
    <div class="buddy" id="buddy" aria-hidden="true">😊 </div>
</div>
</div>
</div>

<!-- RIGHT: settings summary + factbook -->
<div>
    <div class="card" style="margin-bottom:12px;">
        <div style="display:flex; justify-content:space-between; align-
items:center;">
            <div>
                <div style="font-weight:700">Game Controls</div>
                <div style="color:var(--muted); font-size:13px">View game
settings and progress</div>
            </div>
            <div style="text-align:right;">
                <div style="font-size:13px; color:var(--muted)">Streak:</div>
                <span id="streakCount">0</span></div>
                <div style="font-size:12px; color:var(--muted)">Best: <span
id="bestScoreDisplay">0</span></div>
            </div>
        </div>
    </div>
<hr style="margin:8px 0">
```

```

<div id="factBook" style="margin-top:10px;">
    <div style="font-weight:700">Fact Book</div>
    <div id="factList" style="color:var(--muted); font-size:14px;
min-height:54px; margin-top:6px;">No facts yet – answer correctly to
collect them.</div>
</div>

<div style="margin-top:12px; display:flex; gap:8px; justify-
content:space-between;">
    <button id="openControls" class="small"
style="background:var(--accent); color:white">View Game
Controls</button>
    <button id="resetDefault" class="small"
style="background:#e74c3c; color:white"> Reset to Default</button>
</div>
</div>
</div>
</main>

<!-- name modal -->
<div id="nameModal" class="overlay" aria-hidden="false">
    <div class="modal" role="dialog" aria-modal="true" aria-label="Enter
your name">
        <h3>Welcome to Right vs Wrong!</h3>
        <p>Please enter your name to get started:</p>
        <div style="margin-top:10px;">
            <input type="text" id="nameInput" placeholder="Your name"
style="width:100%; padding:10px; border-radius:8px; border:1px solid
#ddd; font-size:16px;" />
        </div>
        <div style="margin-top:15px; text-align:right;">
            <button id="submitName" class="small" style="background:var(--accent);
color:white; padding:10px 20px;">Start Playing</button>
        </div>
    </div>
</div>

<!-- game controls modal -->

```

```
<div id="controlsModal" class="overlay hidden" aria-hidden="true">
  <div class="modal" role="dialog" aria-modal="true" aria-label="Game
Controls">
    <h3>Game Controls</h3>

    <div style="display:flex; gap:8px; align-items:center; margin-
bottom:8px;">
      <input type="checkbox" id="shuffleQ" checked>
      <label for="shuffleQ" style="font-size:13px; color:var(-- 
muted)">Shuffle questions each game</label>
    </div>

    <div style="display:flex; gap:8px; align-items:center; margin-
bottom:8px;">
      <input type="checkbox" id="timerToggle">
      <label for="timerToggle" style="font-size:13px; color:var(-- 
muted)">Enable 10s timer for bonus</label>
    </div>

    <div style="margin-top:15px; text-align:right;">
      <button id="closeControls" class="small" style="background:var(- 
accent); color:white; padding:10px 20px;">Close</button>
    </div>
  </div>
</div>

<!-- results modal -->
<div id="resultsModal" class="overlay hidden" aria-hidden="true">
  <div class="results-modal">
    <h2>🎉 Game Complete! 🎉 </h2>
    <div class="results-score" id="resultsScore">0/0</div>
    <div class="results-message" id="resultsMessage">Great job!</div>

    <div class="results-stickers" id="resultsStickers">
      <div style="font-weight: bold; margin-bottom: 10px;">Your
      Stickers:</div>
      <div id="stickerDisplay"></div>
    </div>
  </div>
</div>
```

```
<div class="results-facts">
  <h4> Facts You Learned:</h4>
  <ul id="factsList"></ul>
</div>

<div class="results-buttons">
  <button class="btn-primary" id="playAgainBtn">Play
Again</button>
  <button class="btn-secondary" id="changeNameBtn">Change
Name</button>
</div>
</div>

<!-- settings modal (password + management) -->
<div id="settingsModal" class="overlay hidden" aria-hidden="true">
  <div class="modal" role="dialog" aria-modal="true" aria-
label="Teacher settings">
    <h3>Teacher Settings – Secure</h3>

    <!-- password box / toggle -->
    <div id="pwBlock" class="row" style="align-items:center;">
      <div style="flex:1">
        <label class="small">Enter teacher code</label>
        <input type="password" id="pwInput" placeholder="Enter code"
aria-label="Teacher password" />
      </div>
      <div>
        <label class="small">Show</label>
        <button id="pwToggle" class="small"> </button>
      </div>
      <div>
        <button id="pwSubmit" class="small" style="background:var(--accent); color:white">Unlock</button>
      </div>
    </div>
  </div>

  <hr>
```

```
<div id="settingsContent" class="hidden">
  <div style="display:flex; gap:8px; align-items:center; margin-bottom:8px;">
    <input id="searchQ" placeholder="Search questions..." />
    <button id="importBtn" class="small">Import JSON</button>
    <input type="file" id="importFile" accept=".json"
style="display:none"/>
    <button id="exportBtn" class="small">Export JSON</button>
  </div>

  <div id="questionsList"></div>

  <div style="margin-top:8px; display:flex; gap:8px; justify-content:space-between;">
    <div>
      <button id="addQBtn" class="small">+ Add Question</button>
      <button id="saveQBtn" class="small" style="background:var(--accent); color:white">Save</button>
    </div>
    <div>
      <button id="closeSettingsBtn" class="small">Close</button>
    </div>
  </div>
</div>

</div>
</div>

<!-- confetti canvas -->
<canvas id="confettiCanvas"></canvas>

<footer>© Right vs Wrong – built for curious kids</footer>

<!-- sounds -->
<audio id="s_correct"
src="https://actions.google.com/sounds/v1/cartoon/pop.ogg"
preload="auto"></audio>
<audio id="s_wrong"
src="https://actions.google.com/sounds/v1/cartoon/wood_plank_flicks.og
```

```
g" preload="auto">></audio>
<audio id="s_end"
src="https://actions.google.com/sounds/v1/cartoon/slide_whistle_to_drum_hit.ogg" preload="auto"></audio>

<script>
/* =====
App Data & Defaults
===== */

const DEFAULT_DATA = [
  { id: "q1", difficulty:"easy",
    video:"https://www.youtube.com/embed/_7R4Sf1sINU",
    question:"What did the kid do wrong after finishing their snack?", 
    options:["Threw the wrapper on the ground","Put the wrapper in the bin","Kept the wrapper in their pocket"],
    correct:0, hint:"Think about the wrapper", fact:"Putting trash in a bin keeps places clean." },
  { id: "q2", difficulty:"easy",
    video:"https://www.youtube.com/embed/dQw4w9WgXcQ",
    question:"What mistake did the kid make while washing hands?", 
    options:["Left the water running","Turned off the tap before getting soap","Used too much soap"],
    correct:0, hint:"Water matters", fact:"Turning taps off saves water." },
  { id: "q3", difficulty:"medium",
    video:"https://www.youtube.com/embed/M7lc1UVf-VE",
    question:"What did the kid forget to do before leaving their room?", 
    options:["Forgot to turn off the lights","Closed the window","Put away their toys"],
    correct:0, hint:"Think energy", fact:"Turning off lights saves electricity." },
  { id: "q4", difficulty:"easy",
    video:"https://www.youtube.com/embed/ScMzIvxBSi4",
    question:"What should you do before crossing the road?", 
    options:["Look both ways","Run quickly","Close your eyes"],
    correct:0, hint:"Be safe", fact:"Always look both ways before crossing." },
]
```

```

    { id: "q5", difficulty:"medium",
      video:"https://www.youtube.com/embed/tgbNymZ7vqY",
      question:"What is wrong if you don't wash fruits?",  

      options:["They can have dirt or germs","They taste bad","They  

      become too soft"],  

      correct:0, hint:"Think hygiene", fact:"Washing fruit removes dirt  

      and germs." }  

  ];  
  

let DATA = JSON.parse(localStorage.getItem("rvs_quiz")) ||  

DEFAULT_DATA.slice();  

let SETTINGS = JSON.parse(localStorage.getItem("rvs_settings")) || {  

  accent:  

getComputedStyle(document.documentElement).getPropertyValue('--  

accent').trim() || "#ffb02e",  

  buddySet: "faces",  

  font: "Comic Sans MS, Poppins, sans-serif",  

  shuffle: true,  

  timerEnabled: false  

};  
  

let state = {  

  order: [], // randomized question order  

  idx: 0,  

  score: 0,  

  stickers: JSON.parse(localStorage.getItem("rvs_stickers")) || [],  

  streak: parseInt(localStorage.getItem("rvs_streak"))||0),  

  best: parseInt(localStorage.getItem("rvs_best"))||0),  

  timer: null,  

  timerSec: 10,  

  mute: false,  

  childName: localStorage.getItem("rvs_childName") || "",  

  hintPenalty: 1, // subtract 1 point when using hint  

  shuffleQuestions: SETTINGS.shuffle,  

  lastPlayed: JSON.parse(localStorage.getItem("rvs_lastPlayed")) || {}  

};  
  

/* DOM refs */  

const videoFrame = document.getElementById("videoFrame");

```

```
const questionText = document.getElementById("questionText");
const optionsArea = document.getElementById("optionsArea");
const progressText = document.getElementById("progressText");
const levelFill = document.getElementById("levelFill");
const nextBtn = document.getElementById("nextBtn");
const hintBtn = document.getElementById("hintBtn");
const stickerCount = document.getElementById("stickerCount");
const messageDiv = document.getElementById("message");
const buddyEl = document.getElementById("buddy");
const timerDisplay = document.getElementById("timerDisplay");
const settingsIcon = document.getElementById("settingsIcon");
const settingsModal = document.getElementById("settingsModal");
const pwInput = document.getElementById("pwInput");
const pwToggle = document.getElementById("pwToggle");
const pwSubmit = document.getElementById("pwSubmit");
const settingsContent = document.getElementById("settingsContent");
const questionsList = document.getElementById("questionsList");
const searchQ = document.getElementById("searchQ");
const importFile = document.getElementById("importFile");
const confettiCanvas = document.getElementById("confettiCanvas");
const s_correct = document.getElementById("s_correct");
const s_wrong = document.getElementById("s_wrong");
const s_end = document.getElementById("s_end");
const resetDefaultBtn = document.getElementById("resetDefault");
const muteBtn = document.getElementById("muteBtn");
const shuffleQ = document.getElementById("shuffleQ");
const timerToggle = document.getElementById("timerToggle");
const nameBtn = document.getElementById("nameBtn");
const factList = document.getElementById("factList");
const streakCount = document.getElementById("streakCount");
const bestScoreDisplay = document.getElementById("bestScoreDisplay");
const nameModal = document.getElementById("nameModal");
const nameInput = document.getElementById("nameInput");
const submitName = document.getElementById("submitName");
const controlsModal = document.getElementById("controlsModal");
const openControls = document.getElementById("openControls");
const closeControls = document.getElementById("closeControls");
const resultsModal = document.getElementById("resultsModal");
const resultsScore = document.getElementById("resultsScore");
```

```

const resultsMessage = document.getElementById("resultsMessage");
const resultsStickers = document.getElementById("resultsStickers");
const stickerDisplay = document.getElementById("stickerDisplay");
const factsList = document.getElementById("factsList");
const playAgainBtn = document.getElementById("playAgainBtn");
const changeNameBtn = document.getElementById("changeNameBtn");

/* apply saved theme to UI */
function applySavedTheme(){
    document.documentElement.style.setProperty("--accent",
SETTINGS.accent||"#ffb02e");
    document.documentElement.style.setProperty("--accent-dark",
shadeColor(SETTINGS.accent||"#ffb02e", -10));
    document.documentElement.style.setProperty("--font-main",
SETTINGS.font||"Comic Sans MS, Poppins, sans-serif");
    document.body.style.fontFamily = SETTINGS.font;
}
applySavedTheme();

/* small util: color shade for accent dark */
function shadeColor(color, percent) {
    // color in #rrggbba
    const num = parseInt(color.slice(1),16);
    const r = (num >> 16) + percent;
    const g = (num >> 8 & 0x00FF) + percent;
    const b = (num & 0x0000FF) + percent;
    const newR = Math.min(255, Math.max(0, r));
    const newG = Math.min(255, Math.max(0, g));
    const newB = Math.min(255, Math.max(0, b));
    return `#${(newR<<16 | newG<<8 |
newB).toString(16).padStart(6, '0')}`;
}

/* =====
   Security: password hashing via Web Crypto
   ===== */
const CODE_KEY = "rvs_code_hash";
async function hashCode(str){
    const enc = new TextEncoder().encode(str);

```

```
const hashBuffer = await crypto.subtle.digest('SHA-256', enc);
const hashArray = Array.from(new Uint8Array(hashBuffer));
const hashHex =
hashArray.map(b=>b.toString(16).padStart(2,'0')).join('');
return hashHex;
}
// on first load: if no code stored, store default hashed "67"
(async()=>{
  if(!localStorage.getItem(CODE_KEY)){
    const h = await hashCode("67");
    localStorage.setItem(CODE_KEY, h);
  }
})();

/* password modal handlers */
pwToggle.addEventListener("click", ()=>{
  if(pwInput.type === "password") pwInput.type = "text"; else
  pwInput.type = "password";
});
pwSubmit.addEventListener("click", async ()=>{
  const attempt = pwInput.value.trim();
  if(!attempt) return alert("Enter code");
  const hashed = await hashCode(attempt);
  if(hashed === localStorage.getItem(CODE_KEY)){
    // unlocked
    settingsContent.classList.remove("hidden");
    document.getElementById("pwBlock").classList.add("hidden");
    buildSettingsList();
  } else {
    alert("Wrong code");
  }
});

/* open settings from header icon */
settingsIcon.addEventListener("click", ()=>openSettingsModal());
function openSettingsModal(){
  settingsModal.classList.remove("hidden");
  settingsModal.style.display = "flex";
  pwInput.value = "";
}
```

```

    settingsContent.classList.add("hidden");
    document.getElementById("pwBlock").classList.remove("hidden");
    buildSettingsList();
}
document.getElementById("closeSettingsBtn").addEventListener("click",
()=>{
    settingsModal.classList.add("hidden");
});

/* =====
   SETTINGS: build question editor list
=====
function buildSettingsList(filter=""){
    questionsList.innerHTML = "";
    DATA.forEach((q,i)=>{
        if(filter && !(q.question+q.options.join(" ") +
(q.fact|| "") ).toLowerCase().includes(filter.toLowerCase())) return;
        const card = document.createElement("div");
        card.className = "qcard";
        card.innerHTML = `
            <div style="flex:1">
                <label class="small">Video URL</label>
                <input type="text" class="sq_vid"
value="\${escapeHtml(q.video)}"/>
                <label class="small">Question</label>
                <input type="text" class="sq_q"
value="\${escapeHtml(q.question)}"/>
                <label class="small">Options (comma separated)</label>
                <textarea
class="sq_opts">\${escapeHtml(q.options.join(","))}</textarea>
                <label class="small">Correct Index (0-based)</label>
                <input type="number" class="sq_correct" value="\${q.correct}"
min="0" style="width:80px"/>
                <label class="small">Hint</label>
                <input type="text" class="sq_hint"
value="\${escapeHtml(q.hint|| "")}" />
                <label class="small">Fact</label>
                <input type="text" class="sq_fact"
value="\${escapeHtml(q.fact|| "")}" />

```

```

        </div>
    <div class="q-actions">
        <button class="small" data-action="up"></button>
        <button class="small" data-action="down"></button>
        <button class="small" data-action="remove"
style="background:#e74c3c;color:white">Remove</button>
    </div>
    `;
    // attach handlers with closure index
    const up = card.querySelector('[data-action="up"]');
    const down = card.querySelector('[data-action="down"]');
    const remove = card.querySelector('[data-action="remove"]');
    up.addEventListener("click", ()=>{ if(i>0){ [DATA[i-1],DATA[i]]=[DATA[i],DATA[i-1]]; buildSettingsList(filter); }});
    down.addEventListener("click", ()=>{ if(i<DATA.length-1){ [DATA[i+1],DATA[i]]=[DATA[i],DATA[i+1]]; buildSettingsList(filter); }});
    remove.addEventListener("click", ()=>{ if(confirm("Remove this question?")){ DATA.splice(i,1); buildSettingsList(filter); } });
    questionsList.appendChild(card);
});
}
searchQ && searchQ.addEventListener("input", ()=>
buildSettingsList(searchQ.value));

function addQuestion(){
    DATA.push({id:"q"+Date.now(), difficulty:"easy", video:"", question:"New question", options:["Option A","Option B","Option C"], correct:0, hint:"", fact:""});
    buildSettingsList();
}
document.getElementById("addQBtn").addEventListener("click",
addQuestion);

/* save settings / export / import */
document.getElementById("saveQBtn").addEventListener("click", ()=> {
    // read back UI into DATA
    const cards = questionsList.querySelectorAll('.qcard');
    const newData=[];

```

```

cards.forEach(card=>{
  const vid = card.querySelector('.sq_vid').value.trim();
  const qtxt = card.querySelector('.sq_q').value.trim();
  const optsRaw = card.querySelector('.sq_opts').value.trim();
  const correct =
    parseInt(card.querySelector('.sq_correct').value) || 0;
  const hint = card.querySelector('.sq_hint').value.trim();
  const fact = card.querySelector('.sq_fact').value.trim();
  const opts = optsRaw.split(',').map(s=>s.trim()).filter(Boolean);
  if(qtxt && opts.length>=2){
    newData.push({ id:"q"+Date.now()+Math.random(),
      difficulty:"easy", video:vid, question:qtxt, options:opts, correct:
      Math.min(correct, opts.length-1), hint, fact });
  }
});
if(newData.length>0){
  DATA = newData;
  localStorage.setItem("rvs_quiz", JSON.stringify(DATA));
  alert("Saved. New question set will be used.");
  settingsModal.classList.add("hidden");
  resetGameToStart();
} else {
  alert("No valid questions found in editor.");
}
});

document.getElementById("exportBtn").addEventListener("click", ()=>{
  const blob = new Blob([JSON.stringify(DATA, null,
2)],{type:"application/json"});
  const url = URL.createObjectURL(blob);
  const a = document.createElement("a");
  a.href = url; a.download = "rvs_quiz_export.json"; a.click();
  URL.revokeObjectURL(url);
});
document.getElementById("importBtn").addEventListener("click", ()=>
importFile.click());
importFile.addEventListener("change", (e)=> {
  const file = e.target.files[0];
  if(!file) return;

```

```

const fr = new FileReader();
fr.onload = ()=> {
    try{ const parsed = JSON.parse(fr.result);
if(Array.isArray(parsed)){ DATA = parsed;
localStorage.setItem("rvs_quiz", JSON.stringify(DATA));
alert("Imported."); buildSettingsList(); resetGameToStart(); } else
alert("JSON is not an array.");
    catch(err){ alert("Invalid JSON file."); }
    };
    fr.readAsText(file);
});

document.getElementById("resetDefault").addEventListener("click",
()=>{
    if(confirm("Reset to default questions? This will clear custom
questions.")){
        DATA = DEFAULT_DATA.slice();
        localStorage.removeItem("rvs_quiz");
        resetGameToStart();
    }
});
}

/* =====
Game helper utilities
=====
*/
function escapeHtml(s){ return
s?.replaceAll?.( "&","&").replaceAll?.( "<","<").replaceAll?.( ">"
,">") ?? s; }

function shuffle(arr){
    const a = arr.slice();
    for(let i=a.length-1;i>0;i--){
        const j = Math.floor(Math.random()*(i+1));
        [a[i],a[j]] = [a[j],a[i]];
    }
    return a;
}

/*
=====

```

```

Game initialization & main loop
=====
function buildPool(){
  let pool = DATA.slice();
  if(document.getElementById("shuffleQ").checked) pool =
shuffle(pool);
  state.order = pool;
  state.idx = 0;
  state.score = 0;
  state.timerSec = 10;
  state.stickers = JSON.parse(localStorage.getItem("rvs_stickers")) ||
[];
  document.getElementById("stickerCount").textContent = "Stickers:
"+state.stickers.length;
}
function loadQuestion(){
  if(state.idx >= state.order.length) { showResults(); return; }
  const q = state.order[state.idx];
  // load video and question
  videoFrame.src = q.video;
  questionText.textContent = q.question;
  progressText.textContent = `Question ${state.idx+1} of
${state.order.length}`;
  // build options (shuffled)
  optionsArea.innerHTML = "";
  const shuffled = shuffle(q.options.map((opt, i)=>({opt,i})));
  shuffled.forEach((o, aidx)=>{
    const div = document.createElement("div");
    div.className = "opt";
    div.setAttribute("role","listitem");
    div.innerHTML = `<div
class="label">${String.fromCharCode(65+aidx)}</div><div
style="flex:1">${o.opt}</div>`;
    div.addEventListener("click", ()=> handleAnswer(div, q, o.i));
    optionsArea.appendChild(div);
  });
  // reset UI
  nextBtn.classList.remove("show");
  messageDiv.textContent = "";
}

```

```

buddySetIdle();
updateLevelBar();
// timer
clearTimer();
if(document.getElementById("timerToggle").checked){
    startTimer();
} else { timerDisplay.textContent = ""; }
// save last played
localStorage.setItem("rvs_lastPlayed", JSON.stringify({ idx:
state.idx, timestamp: Date.now() }));
}

/* visual level bar */
function updateLevelBar(){
    const pct = Math.round((state.idx / Math.max(1, state.order.length))
* 100);
    levelFill.style.width = pct + "%";
}

/* answer handling */
function handleAnswer(div, q, chosenIndex){
    // disable all options
    const all = optionsArea.querySelectorAll(".opt");
    all.forEach(o=>o.style.pointerEvents="none");
    clearTimer();
    const correct = q.correct;
    if(chosenIndex === correct){
        div.classList.add("correct");
        if(!state.mute) s_correct.play();
        // sticker reward: random sticker emoji
        const newSticker = randomSticker();
        state.stickers.push(newSticker);
        showFloatingSticker(newSticker);
        // collect fact into factBook (only on correct)
        addFact(q.fact);
        state.score += 1;
        // buddy animation positive
        buddyCelebrate();
        messageDiv.textContent = "Great job! " + (q.fact || "");
    }
}

```

```

        state.streak = (state.streak || 0) + 1;
    } else {
        div.classList.add("wrong");
        // reveal correct option visually
        all.forEach((el, idx) => {
            const txt = el.textContent.trim();
            // find which one matches q.options[correct]
            if(el.textContent.includes(q.options[correct]))
                el.classList.add("correct");
        });
        if(!state.mute) s_wrong.play();
        messageDiv.textContent = "Oops! " + (q.fact || "");
        state.streak = 0;
    }
    // show next
    nextBtn.classList.add("show");
    // save stickers and streak/best
    localStorage.setItem("rvs_stickers",
    JSON.stringify(state.stickers));
    localStorage.setItem("rvs_streak", state.streak);
    if(state.score > state.best){ state.best = state.score;
    localStorage.setItem("rvs_best", state.best); }
    // update counts display
    document.getElementById("stickerCount").textContent = "Stickers: " +
    state.stickers.length;
    streakCount.textContent = state.streak;
    bestScoreDisplay.textContent = state.best;
}

/* next button */
nextBtn.addEventListener("click", ()=>{
    state.idx++;
    if(state.idx < state.order.length) {
        updateLevelBarAnimate();
        loadQuestion();
    } else {
        showResults();
    }
});
```

```

/* hint button */
hintBtn.addEventListener("click", ()=>{
  const q = state.order[state.idx];
  if(!q) return;
  // penalty: deduct one point if available (but not below 0)
  if(state.score > 0){ state.score = Math.max(0, state.score - state.hintPenalty); }
  messageDiv.textContent = "💡 " + (q.hint || "Think carefully!");
  // deduct sticker if any exist (choose: cost 1 sticker if present)
  if(state.stickers.length>0) state.stickers.pop();
  localStorage.setItem("rvs_stickers",
  JSON.stringify(state.stickers));
  document.getElementById("stickerCount").textContent = "Stickers: " +
  state.stickers.length;
});

/* show results with confetti and results menu */
function showResults(){
  // confetti
  runConfetti();
  if(!state.mute) s_end.play();

  // update best score
  if(state.score > state.best) {
    state.best = state.score;
    localStorage.setItem("rvs_best", state.best);
    bestScoreDisplay.textContent = state.best;
  }

  // populate results modal
  const percentage = Math.round((state.score / state.order.length) * 100);
  resultsScore.textContent = `${state.score}/${state.order.length}`;

  // Set message based on performance
  let message = "";
  if(percentage >= 80) {
    message = "Amazing! You're a learning star! 🌟";
  }
}

```

```

} else if(percentage >= 60) {
    message = "Great job! Keep up the good work! 🙌";
} else if(percentage >= 40) {
    message = "Good effort! Practice makes perfect! 🤸";
} else {
    message = "Keep trying! You'll get better! 🌈";
}
resultsMessage.textContent = message;

// Display stickers
stickerDisplay.textContent = state.stickers.length > 0 ?
state.stickers.join(" ") : "No stickers earned yet";

// Display facts
const facts = JSON.parse(localStorage.getItem("rvs_facts")||"[]");
factsList.innerHTML = "";
if(facts.length === 0) {
    factsList.innerHTML = "<li>No facts collected yet</li>";
} else {
    facts.forEach(fact => {
        const li = document.createElement("li");
        li.textContent = fact;
        factsList.appendChild(li);
    });
}

// Show results modal
resultsModal.classList.remove("hidden");
resultsModal.style.display = "flex";
}

/* reset game to start */
function resetGameToStart(){
    buildPool();
    renderFactBook();
    loadQuestion();
}

/* initial build & start */

```

```
document.getElementById("shuffleQ").addEventListener("change", ()=>
SETTINGS.shuffle = document.getElementById("shuffleQ").checked );
document.getElementById("timerToggle").addEventListener("change", ()=>
{ /* nothing immediate */ });
document.getElementById("muteBtn").addEventListener("click", ()=>
{ state.mute = !state.mute; muteBtn.textContent = state.mute ? '🔇' :
'🔈'; });

document.getElementById("nameBtn").addEventListener("click", ()=>{
  const n = prompt("Enter child's name (for results)", "");
  if(n){ state.childName = n; localStorage.setItem("rvs_childName", n); alert("Saved name: "+n); }
});

/* name modal handlers */
submitName.addEventListener("click", ()=>{
  const name = nameInput.value.trim();
  if(name){
    state.childName = name;
    localStorage.setItem("rvs_childName", name);
    nameModal.classList.add("hidden");
    loadQuestion();
  } else {
    alert("Please enter your name");
  }
});

/* game controls modal handlers */
openControls.addEventListener("click", ()=>{
  controlsModal.classList.remove("hidden");
  controlsModal.style.display = "flex";
});

closeControls.addEventListener("click", ()=>{
  controlsModal.classList.add("hidden");
});

/* results modal handlers */
playAgainBtn.addEventListener("click", ()=>{
```

```

resultsModal.classList.add("hidden");
resetGameToStart();
});

changeNameBtn.addEventListener("click", ()=>{
  resultsModal.classList.add("hidden");
  nameModal.style.display = "flex";
  nameInput.value = "";
  nameInput.focus();
});

/* settings header icon opens modal but triggers pw flow earlier */
settingsIcon.addEventListener("dblclick", ()=> openSettingsModal());
// doubleclick alternate

/* random sticker emoji generator */
const STICKERS =
["⭐️","🦋","☀️","🌈","🍀","🦄","⭐️","🎀","🏅","🍎","📦"];
function randomSticker(){ return
STICKERS[Math.floor(Math.random()*STICKERS.length)]; }
function showFloatingSticker(emoji){
  const el = document.createElement("div");
  el.className = "floating";
  el.style.left = (30 + Math.random()*60) + "vw";
  el.textContent = emoji;
  document.body.appendChild(el);
  setTimeout(()=> el.remove(), 1600);
}

/* buddy animations */
function buddyCelebrate(){
  buddyEl.classList.add("jump");
  setTimeout(()=> buddyEl.classList.remove("jump"), 600);
}
function buddySetIdle(){ buddyEl.classList.remove("jump", "shake"); }

/* timer */
function startTimer(){
  let sec = 10;

```

```

timerDisplay.textContent = sec + "s";
state.timer = setInterval(()=>{
    sec--;
    timerDisplay.textContent = sec + "s";
    if(sec<=0){
        clearTimer();
        // time's up: auto mark wrong and reveal
        messageDiv.textContent = "Time's up!";
        // simulate picking nothing: reveal correct
        const all = optionsArea.querySelectorAll(".opt");
        all.forEach(el=>{
            if(el.textContent.includes(state.order[state.idx].options[state.order[state.idx].correct])) el.classList.add("correct");
            el.style.pointerEvents = "none";
        });
        if(!state.mute) s_wrong.play();
        nextBtn.classList.add("show");
    }
},1000);
}
function clearTimer(){ if(state.timer){ clearInterval(state.timer);
state.timer=null; } timerDisplay.textContent = ""; }

/* confetti implementation on canvas (simple) */
function runConfetti(){
    const canvas = confettiCanvas;
    const ctx = canvas.getContext('2d');
    canvas.width = window.innerWidth;
    canvas.height = window.innerHeight;
    const pieces = [];
    for(let i=0;i<80;i++){
        pieces.push({
            x: Math.random()*canvas.width,
            y: Math.random()*-canvas.height,
            r: 6+Math.random()*8,
            color: `hsl(${Math.random()*360},70%,60%)`,
            vx: -2 + Math.random()*4,
            vy: 2 + Math.random()*6,
        })
    }
}

```

```

        rot: Math.random()*360
    });
}
let t=0;
function frame(){
    t++;
    ctx.clearRect(0,0,canvas.width,canvas.height);
    for(const p of pieces){
        p.x += p.vx; p.y += p.vy; p.rot += 4;
        ctx.save();
        ctx.fillStyle = p.color;
        ctx.translate(p.x, p.y);
        ctx.rotate(p.rot * Math.PI/180);
        ctx.fillRect(-p.r/2, -p.r/2, p.r, p.r);
        ctx.restore();
    }
    if(t<120) requestAnimationFrame(frame);
    else ctx.clearRect(0,0,canvas.width,canvas.height);
}
frame();
}

/* fact book */
function addFact(f){
    if(!f) return;
    const facts = JSON.parse(localStorage.getItem("rvs_facts")||"[]");
    if(!facts.includes(f)) facts.push(f);
    localStorage.setItem("rvs_facts", JSON.stringify(facts));
    renderFactBook();
}
function renderFactBook(){
    const facts = JSON.parse(localStorage.getItem("rvs_facts")||"[]");
    if(facts.length==0) factList.textContent = "No facts yet – answer correctly to collect them.";
    else factList.innerHTML = facts.map(f=>`

${escapeHtml(f)}

`).join('');
}

/* UI: show settings modal from header icon (opens password UI) */

```

```

settingsIcon.addEventListener("click", ()=> {
  settingsModal.classList.remove("hidden");
});
document.getElementById("closeSettingsBtn").addEventListener("click",
()=> settingsModal.classList.add("hidden"));

/* reset game / start */
document.addEventListener("DOMContentLoaded", ()=>{
  // Check if name is already saved
  if(!state.childName) {
    // Show name modal
    nameModal.style.display = "flex";
  } else {
    // Name already exists, start game
    buildPool();
    renderFactBook();
    loadQuestion();
  }
  // load persisted theme
  document.getElementById("stickerCount").textContent = "Stickers: " +
  (state.stickers.length||0);
});

/* helper: reset everything and start new */
function resetAll(){
  state.idx = 0; state.score = 0; state.stickers = [];
  localStorage.removeItem("rvs_stickers");
  localStorage.removeItem("rvs_streak");
}

/* reset to start after game or save */
function resetGameToStart(){
  state.idx = 0; state.score = 0;
  buildPool();
  loadQuestion();
}

/* timer/bonus ui animation for level bar */
function updateLevelBarAnimate(){ levelFill.style.transition =

```

```

'width .35s ease-out'; updateLevelBar(); setTimeout(()=>
levelFill.style.transition='width .35s ease-out', 400); }

/* small helpers */
function escapeHtml(s){ if(!s) return ""; return
s.replaceAll("<","&lt;").replaceAll(">","&gt;"); }

/* end of script */
</script>
</body>
</html>

```

I've added a comprehensive results menu that appears at the end of the game with the following features:

1. **Results Modal:** A beautiful modal that appears when the game is complete with:
  - a. Celebration title with emojis
  - b. Score display (e.g., "4/5")
  - c. Performance-based message (different messages for different score ranges)
  - d. Stickers earned display
  - e. Facts learned section with scrollable list
  - f. Action buttons
2. **Performance Messages:** The message changes based on the percentage score:
  - a. 80%+ : "Amazing! You're a learning star! 🌟"
  - b. 60-79% : "Great job! Keep up the good work! 🙌"
  - c. 40-59% : "Good effort! Practice makes perfect! 🤸"
  - d. Below 40% : "Keep trying! You'll get better! 🌈"
3. **Action Buttons:**
  - a. **Play Again:** Starts a new game with the same name
  - b. **Change Name:** Opens the name modal to change the player's name
4. **Visual Design:** The results modal has:
  - a. Clean, kid-friendly design with rounded corners
  - b. Color-coded sections
  - c. Hover effects on buttons
  - d. Proper spacing and typography
  - e. Confetti animation that plays when the modal appears
5. **Data Display:**

- a. Shows all stickers earned during the game
- b. Lists all facts collected in a scrollable section
- c. Updates the best score if needed

The results menu replaces the simple alert that was previously shown, providing a much more engaging and informative end-game experience for kids.

## USER

Change this to the dlaut questions [ { "id": "q1", "difficulty": "easy", "video": "[https://www.youtube.com/embed/\\_7R4SflsINU](https://www.youtube.com/embed/_7R4SflsINU)", "question": "What did the kid do wrong after finishing their snack?", "options": [ "Threw the wrapper on the ground", "Put the wrapper in the bin", "Kept the wrapper in their pocket" ], "correct": 0, "hint": "Think about the wrapper", "fact": "Putting trash in a bin keeps places clean." }, { "id": "q2", "difficulty": "easy", "video": "<https://www.youtube.com/embed/dQw4w9WgXcQ>", "question": "What mistake did the kid make while washing hands?", "options": [ "Left the water running", "Turned off the tap before getting soap", "Used too much soap" ], "correct": 0, "hint": "Water matters", "fact": "Turning taps off saves water." }, { "id": "q3", "difficulty": "medium", "video": "<https://www.youtube.com/embed/M7lc1UVf-VE>", "question": "What did the kid forget to do before leaving their room?", "options": [ "Forgot to turn off the lights", "Closed the window", "Put away their toys" ], "correct": 0, "hint": "Think energy", "fact": "Turning off lights saves electricity." }, { "id": "q4", "difficulty": "easy", "video": "<https://www.youtube.com/embed/ScMzIvxBSi4>", "question": "What should you do before crossing the road?", "options": [ "Look both ways", "Run quickly", "Close your eyes" ], "correct": 0, "hint": "Be safe", "fact": "Always look both ways before crossing." }, { "id": "q5", "difficulty": "medium", "video": "<https://www.youtube.com/embed/tgbNymZ7vqY>", "question": "What is wrong if you don't wash fruits?", "options": [ "They can have dirt or germs", "They taste bad", "They become too soft" ], "correct": 0, "hint": "Think hygiene", "fact": "Washing fruit removes dirt and germs." }, { "id": "q6", "difficulty": "medium", "video": "<https://www.youtube.com/embed/jNQXAC9lVRw>", "question": "What did the kid do wrong while brushing their teeth?", "options": [ "Left the tap running", "Used too much toothpaste", "Forgot to rinse their mouth" ], "correct": 0, "hint": "Think water saving", "fact": "Turning off the tap while brushing saves a lot of water." } ]

# Right vs Wrong — Kids Learning



```
<div style="padding:12px;">
  <div class="progress" aria-hidden="false">
    <div id="progressText">Question 1 of 5</div>
    <div class="level">
      <div id="levelFill" class="level-fill" aria-hidden="true"></div>
    </div>
    <div id="timerDisplay" aria-live="polite" style="min-width:50px;text-align:right;font-size:13px;color:var(--muted)"></div>
  </div>

  <h2 class="question" id="questionText">Question text</h2>

  <div class="options" id="optionsArea" role="list"></div>

  <div class="controls" style="margin-top:12px;">
    <div class="left-controls">
      <button class="small hint" id="hintBtn" aria-label="Hint (costs sticker)">💡 Hint</button>
      <div id="stickerBox" style="margin-left:8px;"><span id="stickerCount">Stickers: 0</span></div>
    </div>
    <div class="right-controls">
      <button class="small mute" id="muteBtn" aria-label="Mute/unmute sounds">🔇 </button>
      <button class="small" id="nameBtn" aria-label="Set child name">👤 Name</button>
      <button class="small next-btn" id="nextBtn" aria-label="Next question">Next</button>
    </div>
  </div>

  <div style="margin-top:10px; display:flex; justify-content:space-between; align-items:center;">
    <div id="message" style="color:var(--muted); font-weight:600"></div>
    <div class="buddy" id="buddy" aria-hidden="true">😊 </div>
  </div>
</div>
```

## Game Controls

View game settings and progress

Streak: 0

Best: 0

```
<hr style="margin:8px 0">

<div id="factBook" style="margin-top:10px;">
  <div style="font-weight:700">Fact Book</div>
  <div id="factList" style="color:var(--muted); font-size:14px; min-height:54px; margin-top:6px;">No facts yet – answer correctly to collect them.</div>
</div>

<div style="margin-top:12px; display:flex; gap:8px; justify-content:space-between;">
  <button id="openControls" class="small" style="background:var(--accent); color:white">View Game Controls</button>
  <button id="resetDefault" class="small" style="background:#e74c3c; color:white">  Reset to Default</button>
</div>
</div>
```

## Welcome to Right vs Wrong!

Please enter your name to get started:

## Game Controls

```
<div style="display:flex; gap:8px; align-items:center; margin-bottom:8px;">
  <input type="checkbox" id="shuffleQ" checked>
  <label for="shuffleQ" style="font-size:13px; color:var(--muted)">Shuffle questions each game</label>
</div>

<div style="display:flex; gap:8px; align-items:center; margin-bottom:8px;">
```

```
<input type="checkbox" id="timerToggle">
<label for="timerToggle" style="font-size:13px; color:var(--muted)">Enable 10s timer for bonus</label>
</div>

<div style="margin-top:15px; text-align:right;">
  <button id="closeControls" class="small" style="background:var(--accent); color:white; padding:10px 20px;">Close</button>
</div>
```

## Game Complete! 🎉

0/0

Great job!

```
<div class="results-stickers" id="resultsStickers">
  <div style="font-weight: bold; margin-bottom: 10px;">Your Stickers:</div>
  <div id="stickerDisplay"></div>
</div>

<div class="results-facts">
  <h4>FACTS<img alt="book icon" style="vertical-align: middle;"/> Facts You Learned:</h4>
  <ul id="factsList"></ul>
</div>

<div class="results-buttons">
  <button class="btn-primary" id="playAgainBtn">Play Again</button>
  <button class="btn-secondary" id="changeNameBtn">Change Name</button>
</div>
```

## Teacher Settings — Secure

```
<!-- password box / toggle -->
<div id="pwBlock" class="row" style="align-items:center;">
  <div style="flex:1">
```

```
<label class="small">Enter teacher code</label>
<input type="password" id="pwInput" placeholder="Enter code" aria-
label="Teacher password" />
</div>
<div>
    <label class="small">Show</label>
    <button id="pwToggle" class="small">👁</button>
</div>
<div>
    <button id="pwSubmit" class="small" style="background:var(--accent); color:white">Unlock</button>
</div>
</div>

<hr>

<div id="settingsContent" class="hidden">
    <div style="display:flex; gap:8px; align-items:center; margin-
bottom:8px;">
        <input id="searchQ" placeholder="Search questions..." />
        <button id="importBtn" class="small">Import JSON</button>
        <input type="file" id="importFile" accept=".json"
style="display:none"/>
        <button id="exportBtn" class="small">Export JSON</button>
    </div>
    <div id="questionsList"></div>
    <div style="margin-top:8px; display:flex; gap:8px; justify-
content:space-between;">
        <div>
            <button id="addQBtn" class="small">➕ Add Question</button>
            <button id="saveQBtn" class="small" style="background:var(--accent); color:white">Save</button>
        </div>
        <div>
            <button id="closeSettingsBtn" class="small">Close</button>
        </div>
    </div>
</div>
```

```
</div>
```

© Right vs Wrong — built for curious kids

## ASSISTANT

Thought for 1 seconds > The user wants me to update the DEFAULT\_DATA array in the JavaScript code with the new questions they provided. They've provided a JSON array with 6 questions, including a new question (q6) about brushing teeth. >> I need to replace the existing DEFAULT\_DATA array with the new one they provided. Let me make this change in the JavaScript section of the HTML file. I'll update the questions in the game with the new set you provided. Here's the updated version: <!DOCTYPE html>

```
<html lang="en">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width,initial-scale=1" />
<title>Right vs Wrong – Kids Learning Game</title>
<style>
:root{
    --accent: #ffb02e;
    --accent-dark: #ff9b00;
    --bg: #f7fbff;
    --green: #1fb16a;
    --red: #e74c3c;
    --muted: #6b7280;
    --card: #fff;
    --shadow: 0 10px 25px rgba(15,23,42,0.08);
    --radius: 14px;
    --font-main: "Comic Sans MS", "Poppins", sans-serif;
}
*{box-sizing:border-box}
body{
    margin:0; font-family:var(--font-main); background:linear-gradient(180deg,var(--bg),#fff);
    color:#0f172a; -webkit-font-smoothing:antialiased;
}

/* header */
header{
```

```
    background:linear-gradient(90deg,var(--accent),var(--accent-dark));
    color:white; padding:14px 16px; display:flex; align-items:center;
justify-content:center;
    position:relative; box-shadow:var(--shadow);
}
header h1{ margin:0; font-size:18px; letter-spacing:0.3px; }
#settingsIcon{ position:absolute; right:12px; top:8px; width:40px;
height:40px; border-radius:999px;
    display:flex; align-items:center; justify-content:center;
background:rgba(255,255,255,0.12); cursor:pointer; }
#settingsIcon:focus{outline:3px solid rgba(255,255,255,0.18) }

/* layout */
main.app{ max-width:1100px; margin:18px auto; padding:12px;
display:grid; grid-template-columns:1fr 420px; gap:20px;
transition:opacity .25s ease; }
@media(max-width:900px){ main.app{ grid-template-columns:1fr;
padding:10px } }

/* video card */
.video-card{ background:var(--card); border-radius:var(--radius);
overflow:hidden; box-shadow:var(--shadow); display:flex; flex-
direction:column; }
.video-wrap{ position:relative; padding-top:56.25%; } /* 16:9 */
.video-wrap iframe{ position:absolute; inset:0; width:100%;
height:100%; border:0; }

/* question card */
.card{ background:linear-gradient(180deg, #fff, #ffffef9); border-
radius:var(--radius); padding:14px; box-shadow:var(--shadow);
border:4px solid rgba(255,224,178,0.6); }
.progress{ color:var(--muted); font-size:13px; margin-bottom:8px;
display:flex; align-items:center; justify-content:space-between;
gap:8px; }
.level{ flex:1; height:10px; background:#eef2f7; border-
radius:999px; overflow:hidden; margin-left:8px; margin-right:8px; }
.level-fill{ height:100%; width:0%; background:linear-
gradient(90deg,var(--accent),var(--accent-dark)); }
```

```
transition:width .35s ease-out; }

h2.question{ font-size:18px; margin:6px 0 12px 0; text-align:center; color:#0c1720; }

.options{ display:flex; flex-direction:column; gap:10px; }
.opt{ display:flex; gap:12px; align-items:center; padding:10px 12px; background:var(--card); border-radius:10px; cursor:pointer; border:2px solid #fff6e6;
      box-shadow:0 8px 18px rgba(15,23,42,0.04);
transition:transform .12s ease, box-shadow .12s ease; font-size:15px; }
.opt:hover{ transform:translateY(-3px) }
.opt .label{ font-weight:700; width:34px; height:34px; display:inline-flex; align-items:center; justify-content:center; border-radius:999px; background:#fff6e6; }
.opt.correct{ background:linear-gradient(90deg, rgba(31,177,106,0.08), rgba(31,177,106,0.02)); border-color:rgba(31,177,106,0.15) }
.opt.wrong{ background:linear-gradient(90deg, rgba(231,76,60,0.06), rgba(231,76,60,0.02)); border-color:rgba(231,76,60,0.12) }
button.small{ padding:8px 10px; border-radius:10px; font-weight:700; cursor:pointer; border:none; }

/* controls row */
.controls{ display:flex; gap:10px; align-items:center; justify-content:space-between; margin-top:10px; flex-wrap:wrap; }
.left-controls{ display:flex; gap:8px; align-items:center; }
.right-controls{ display:flex; gap:8px; align-items:center; }

/* badges, stickers */
#stickers{ background:#fff9e9; border:3px dashed var(--accent); padding:8px; border-radius:12px; font-size:14px; }
.floating{ position:fixed; pointer-events:none; z-index:2000; font-size:22px; animation:floatUp 1.6s ease-out forwards; }
@keyframes floatUp{ 0%{ transform:translateY(0) scale(.9); opacity:1 } 100%{ transform:translateY(-120px) scale(1.2); opacity:0 } }
```

```
/* hint button */
.hint{ background:#0b74ff; color:#fff }
.mute{ background:#e2e8f0; color:#0b1320 }

/* settings modal / panel */
.overlay{ position:fixed; inset:0; background:rgba(0,0,0,0.35);
display:flex; align-items:center; justify-content:center; z-
index:3000; }
.modal{ width:940px; max-width:95%; max-height:90vh; overflow:auto;
background:white; padding:14px; border-radius:14px; box-shadow:0 20px
60px rgba(2,6,23,0.4); }
.modal h3{ margin-top:0 }
.row{ display:flex; gap:8px; align-items:center; margin-bottom:8px;
flex-wrap:wrap; }
.field{ flex:1; min-width:180px; }
label.small{ font-size:12px; color:var(--muted); display:block;
margin-bottom:4px }
input[type="text"], textarea, select{
width:100%; padding:8px; border-radius:8px; border:1px solid #ddd;
font-size:14px;
}
textarea{ min-height:54px; resize:vertical; }

.qcard{ border:1px solid #f0e3c9; padding:8px; border-radius:8px;
margin-bottom:8px; background:#fffef8; display:flex; gap:10px; align-
items:flex-start; }
.q-actions{ display:flex; gap:6px; flex-direction:column; }

/* search bar */
.search{ display:flex; gap:8px; align-items:center; margin-
bottom:8px; }

/* footer */
.footer{ text-align:center; color:var(--muted); margin:20px 0 40px;
font-size:14px; }

/* confetti canvas */
#confettiCanvas{ position:fixed; left:0; top:0; width:100%;
height:100%; pointer-events:none; z-index:2500 }
```

```
/* transitions */
.fade{ transition: opacity .25s ease; }
.hidden{ display:none !important; opacity:0 !important }

/* buddy animations */
.buddy{ font-size:36px; display:inline-block;
transition:transform .35s ease, filter .35s; }
.jump{ transform:translateY(-10px) scale(1.1); }
.shake{ animation:shake .6s; }
@keyframes shake{ 10%{ transform:translateX(-4px) }
30%{ transform:translateX(4px) } 50%{ transform:translateX(-3px) }
70%{ transform:translateX(3px) } 90%{ transform:translateX(-1px) } }

/* results */
.results{ display:flex; flex-direction:column; gap:10px; align-items:center; text-align:center; padding:12px }
.badge{ background:#fff9e9; border:2px solid var(--accent);
padding:8px 12px; border-radius:10px; font-weight:700 }

/* results modal */
.results-modal {
  background: white;
  border-radius: 20px;
  padding: 30px;
  max-width: 500px;
  width: 90%;
  text-align: center;
  box-shadow: 0 20px 60px rgba(0,0,0,0.3);
  position: relative;
}

.results-modal h2 {
  color: var(--accent);
  font-size: 32px;
  margin-bottom: 20px;
}

.results-score {
```

```
    font-size: 48px;
    font-weight: bold;
    color: var(--green);
    margin: 20px 0;
}

.results-message {
    font-size: 18px;
    color: var(--muted);
    margin-bottom: 20px;
}

.results-stickers {
    font-size: 24px;
    margin: 15px 0;
}

.results-facts {
    background: #f8f9fa;
    border-radius: 10px;
    padding: 15px;
    margin: 20px 0;
    text-align: left;
    max-height: 200px;
    overflow-y: auto;
}

.results-facts h4 {
    margin-top: 0;
    color: var(--accent);
}

.results-facts ul {
    margin: 10px 0;
    padding-left: 20px;
}

.results-facts li {
    margin: 8px 0;
```

```
    color: var(--muted);
}

.results-buttons {
  display: flex;
  gap: 15px;
  justify-content: center;
  margin-top: 25px;
}

.results-buttons button {
  padding: 12px 25px;
  border-radius: 10px;
  font-weight: bold;
  cursor: pointer;
  border: none;
  font-size: 16px;
  transition: transform 0.2s;
}

.results-buttons button:hover {
  transform: translateY(-2px);
}

.btn-primary {
  background: var(--accent);
  color: white;
}

.btn-secondary {
  background: #e2e8f0;
  color: var(--muted);
}

/* accessibility large text toggle (class toggled by settings) */
.large-text{ font-size:18px; }

</style>
</head>
<body>
```

```
<header>
  <h1>Right vs Wrong – Kids Learning</h1>
  <div id="settingsIcon" tabindex="0" title="Teacher
Settings">⚙</div>
</header>

<main class="app" id="app">
  <!-- LEFT: video + controls -->
  <div class="video-card card" aria-live="polite">
    <div class="video-wrap" id="videoWrap" role="region" aria-
label="Scenario video">
      <iframe id="videoFrame" src="" allow="accelerometer; autoplay;
clipboard-write; encrypted-media; gyroscope; picture-in-picture"
allowfullscreen title="Scenario video"></iframe>
    </div>

    <div style="padding:12px;">
      <div class="progress" aria-hidden="false">
        <div id="progressText">Question 1 of 6</div>
        <div class="level">
          <div id="levelFill" class="level-fill" aria-
hidden="true"></div>
        </div>
        <div id="timerDisplay" aria-live="polite" style="min-
width:50px;text-align:right;font-size:13px;color:var(--muted)"></div>
      </div>
    </div>

    <h2 class="question" id="questionText">Question text</h2>

    <div class="options" id="optionsArea" role="list"></div>

    <div class="controls" style="margin-top:12px;">
      <div class="left-controls">
        <button class="small hint" id="hintBtn" aria-label="Hint
(costs sticker)">💡 Hint</button>
        <div id="stickerBox" style="margin-left:8px;"><span
id="stickerCount">Stickers: 0</span></div>
      </div>
    </div>
  </div>
</main>
```

```
<div class="right-controls">
    <button class="small mute" id="muteBtn" aria-
label="Mute/unmute sounds">🔇 </button>
    <button class="small" id="nameBtn" aria-label="Set child
name">👤 Name</button>
    <button class="small next-btn" id="nextBtn" aria-label="Next
question">Next</button>
</div>
</div>

<div style="margin-top:10px; display:flex; justify-
content:space-between; align-items:center;">
    <div id="message" style="color:var(--muted); font-
weight:600"></div>
    <div class="buddy" id="buddy" aria-hidden="true">😊 </div>
</div>
</div>
</div>

<!-- RIGHT: settings summary + factbook -->
<div>
    <div class="card" style="margin-bottom:12px;">
        <div style="display:flex; justify-content:space-between; align-
items:center;">
            <div>
                <div style="font-weight:700">Game Controls</div>
                <div style="color:var(--muted); font-size:13px">View game
settings and progress</div>
            </div>
            <div style="text-align:right;">
                <div style="font-size:13px; color:var(--muted)">Streak:</div>
                <span id="streakCount">0</span>
                <div style="font-size:12px; color:var(--muted)">Best: <span
id="bestScoreDisplay">0</span></div>
            </div>
        </div>
    </div>
<hr style="margin:8px 0">
```

```
<div id="factBook" style="margin-top:10px;">
    <div style="font-weight:700">Fact Book</div>
    <div id="factList" style="color:var(--muted); font-size:14px;
min-height:54px; margin-top:6px;">No facts yet – answer correctly to
collect them.</div>
</div>

<div style="margin-top:12px; display:flex; gap:8px; justify-
content:space-between;">
    <button id="openControls" class="small"
style="background:var(--accent); color:white">View Game
Controls</button>
    <button id="resetDefault" class="small"
style="background:#e74c3c; color:white"> Reset to Default</button>
</div>
</div>
</main>

<!-- name modal -->
<div id="nameModal" class="overlay" aria-hidden="false">
    <div class="modal" role="dialog" aria-modal="true" aria-label="Enter
your name">
        <h3>Welcome to Right vs Wrong!</h3>
        <p>Please enter your name to get started:</p>
        <div style="margin-top:10px;">
            <input type="text" id="nameInput" placeholder="Your name"
style="width:100%; padding:10px; border-radius:8px; border:1px solid
#ddd; font-size:16px;" />
        </div>
        <div style="margin-top:15px; text-align:right;">
            <button id="submitName" class="small" style="background:var(--accent);
color:white; padding:10px 20px;">Start Playing</button>
        </div>
    </div>
</div>

<!-- game controls modal -->
<div id="controlsModal" class="overlay hidden" aria-hidden="true">
```

```
<div class="modal" role="dialog" aria-modal="true" aria-label="Game Controls">
  <h3>Game Controls</h3>

    <div style="display:flex; gap:8px; align-items:center; margin-bottom:8px;">
      <input type="checkbox" id="shuffleQ" checked>
      <label for="shuffleQ" style="font-size:13px; color:var(--muted)">Shuffle questions each game</label>
    </div>

    <div style="display:flex; gap:8px; align-items:center; margin-bottom:8px;">
      <input type="checkbox" id="timerToggle">
      <label for="timerToggle" style="font-size:13px; color:var(--muted)">Enable 10s timer for bonus</label>
    </div>

    <div style="margin-top:15px; text-align:right;">
      <button id="closeControls" class="small" style="background:var(--accent); color:white; padding:10px 20px;">Close</button>
    </div>
  </div>
</div>

<!-- results modal -->
<div id="resultsModal" class="overlay hidden" aria-hidden="true">
  <div class="results-modal">
    <h2>🎉 Game Complete! 🎉 </h2>
    <div class="results-score" id="resultsScore">0/0</div>
    <div class="results-message" id="resultsMessage">Great job!</div>

    <div class="results-stickers" id="resultsStickers">
      <div style="font-weight: bold; margin-bottom: 10px;">Your Stickers:</div>
      <div id="stickerDisplay"></div>
    </div>

    <div class="results-facts">
```

```
<h4>  Facts You Learned:</h4>
<ul id="factsList"></ul>
</div>

<div class="results-buttons">
    <button class="btn-primary" id="playAgainBtn">Play
Again</button>
    <button class="btn-secondary" id="changeNameBtn">Change
Name</button>
</div>
</div>
</div>

<!-- settings modal (password + management) -->
<div id="settingsModal" class="overlay hidden" aria-hidden="true">
    <div class="modal" role="dialog" aria-modal="true" aria-
label="Teacher settings">
        <h3>Teacher Settings – Secure</h3>

        <!-- password box / toggle -->
        <div id="pwBlock" class="row" style="align-items:center;">
            <div style="flex:1">
                <label class="small">Enter teacher code</label>
                <input type="password" id="pwInput" placeholder="Enter code"
aria-label="Teacher password" />
            </div>
            <div>
                <label class="small">Show</label>
                <button id="pwToggle" class="small"> </button>
            </div>
            <div>
                <button id="pwSubmit" class="small" style="background:var(--accent); color:white">Unlock</button>
            </div>
        </div>
    </div>
    <hr>
<div id="settingsContent" class="hidden">
```

```
<div style="display:flex; gap:8px; align-items:center; margin-bottom:8px;">
    <input id="searchQ" placeholder="Search questions..." />
    <button id="importBtn" class="small">Import JSON</button>
    <input type="file" id="importFile" accept=".json"
style="display:none"/>
    <button id="exportBtn" class="small">Export JSON</button>
</div>

<div id="questionsList"></div>

<div style="margin-top:8px; display:flex; gap:8px; justify-content:space-between;">
    <div>
        <button id="addQBtn" class="small">+ Add Question</button>
        <button id="saveQBtn" class="small" style="background:var(--accent); color:white">Save</button>
    </div>
    <div>
        <button id="closeSettingsBtn" class="small">Close</button>
    </div>
</div>
</div>

<!-- confetti canvas -->
<canvas id="confettiCanvas"></canvas>

<footer>© Right vs Wrong – built for curious kids</footer>

<!-- sounds -->
<audio id="s_correct"
src="https://actions.google.com/sounds/v1/cartoon/pop.ogg"
preload="auto"></audio>
<audio id="s_wrong"
src="https://actions.google.com/sounds/v1/cartoon/wood_plank_flicks.ogg"
preload="auto"></audio>
```

```
<audio id="s_end"
src="https://actions.google.com/sounds/v1/cartoon/slide_whistle_to_drum_hit.ogg" preload="auto"></audio>

<script>
/* =====
App Data & Defaults
===== */

const DEFAULT_DATA = [
{
  "id": "q1",
  "difficulty": "easy",
  "video": "https://www.youtube.com/embed/\_7R4SflsINU",
  "question": "What did the kid do wrong after finishing their snack?",
  "options": [
    "Threw the wrapper on the ground",
    "Put the wrapper in the bin",
    "Kept the wrapper in their pocket"
  ],
  "correct": 0,
  "hint": "Think about the wrapper",
  "fact": "Putting trash in a bin keeps places clean."
},
{
  "id": "q2",
  "difficulty": "easy",
  "video": "https://www.youtube.com/embed/dQw4w9WgXcQ",
  "question": "What mistake did the kid make while washing hands?",
  "options": [
    "Left the water running",
    "Turned off the tap before getting soap",
    "Used too much soap"
  ],
  "correct": 0,
  "hint": "Water matters",
  "fact": "Turning taps off saves water."
}
];
```

```
{  
    "id": "q3",  
    "difficulty": "medium",  
    "video": "https://www.youtube.com/embed/M7lc1UVf-VE",  
    "question": "What did the kid forget to do before leaving their room?",  
    "options": [  
        "Forgot to turn off the lights",  
        "Closed the window",  
        "Put away their toys"  
    ],  
    "correct": 0,  
    "hint": "Think energy",  
    "fact": "Turning off lights saves electricity."  
},  
{  
    "id": "q4",  
    "difficulty": "easy",  
    "video": "https://www.youtube.com/embed/ScMzIvxBSi4",  

```

```

    "correct": 0,
    "hint": "Think hygiene",
    "fact": "Washing fruit removes dirt and germs."
},
{
  "id": "q6",
  "difficulty": "medium",
  "video": "https://www.youtube.com/embed/jNQXAC9IVRw",
  "question": "What did the kid do wrong while brushing their teeth?",
  "options": [
    "Left the tap running",
    "Used too much toothpaste",
    "Forgot to rinse their mouth"
  ],
  "correct": 0,
  "hint": "Think water saving",
  "fact": "Turning off the tap while brushing saves a lot of water."
}
];
};

let DATA = JSON.parse(localStorage.getItem("rvs_quiz")) ||
DEFAULT_DATA.slice();
let SETTINGS = JSON.parse(localStorage.getItem("rvs_settings")) || {
  accent:
    getComputedStyle(document.documentElement).getPropertyValue('--accent').trim() || "#ffb02e",
  buddySet: "faces",
  font: "Comic Sans MS, Poppins, sans-serif",
  shuffle: true,
  timerEnabled: false
};

let state = {
  order: [], // randomized question order
  idx: 0,
  score: 0,
  stickers: JSON.parse(localStorage.getItem("rvs_stickers")) || [],
  streak: parseInt(localStorage.getItem("rvs_streak")) || 0,
}

```

```
best: parseInt(localStorage.getItem("rvs_best") || 0),
timer: null,
timerSec: 10,
mute: false,
childName: localStorage.getItem("rvs_childName") || "",
hintPenalty: 1, // subtract 1 point when using hint
shuffleQuestions: SETTINGS.shuffle,
lastPlayed: JSON.parse(localStorage.getItem("rvs_lastPlayed")) || {}
};

/* DOM refs */
const videoFrame = document.getElementById("videoFrame");
const questionText = document.getElementById("questionText");
const optionsArea = document.getElementById("optionsArea");
const progressText = document.getElementById("progressText");
const levelFill = document.getElementById("levelFill");
const nextBtn = document.getElementById("nextBtn");
const hintBtn = document.getElementById("hintBtn");
const stickerCount = document.getElementById("stickerCount");
const messageDiv = document.getElementById("message");
const buddyEl = document.getElementById("buddy");
const timerDisplay = document.getElementById("timerDisplay");
const settingsIcon = document.getElementById("settingsIcon");
const settingsModal = document.getElementById("settingsModal");
const pwInput = document.getElementById("pwInput");
const pwToggle = document.getElementById("pwToggle");
const pwSubmit = document.getElementById("pwSubmit");
const settingsContent = document.getElementById("settingsContent");
const questionsList = document.getElementById("questionsList");
const searchQ = document.getElementById("searchQ");
const importFile = document.getElementById("importFile");
const confettiCanvas = document.getElementById("confettiCanvas");
const s_correct = document.getElementById("s_correct");
const s_wrong = document.getElementById("s_wrong");
const s_end = document.getElementById("s_end");
const resetDefaultBtn = document.getElementById("resetDefault");
const muteBtn = document.getElementById("muteBtn");
const shuffleQ = document.getElementById("shuffleQ");
const timerToggle = document.getElementById("timerToggle");
```

```
const nameBtn = document.getElementById("nameBtn");
const factList = document.getElementById("factList");
const streakCount = document.getElementById("streakCount");
const bestScoreDisplay = document.getElementById("bestScoreDisplay");
const nameModal = document.getElementById("nameModal");
const nameInput = document.getElementById("nameInput");
const submitName = document.getElementById("submitName");
const controlsModal = document.getElementById("controlsModal");
const openControls = document.getElementById("openControls");
const closeControls = document.getElementById("closeControls");
const resultsModal = document.getElementById("resultsModal");
const resultsScore = document.getElementById("resultsScore");
const resultsMessage = document.getElementById("resultsMessage");
const resultsStickers = document.getElementById("resultsStickers");
const stickerDisplay = document.getElementById("stickerDisplay");
const factsList = document.getElementById("factsList");
const playAgainBtn = document.getElementById("playAgainBtn");
const changeNameBtn = document.getElementById("changeNameBtn");

/* apply saved theme to UI */
function applySavedTheme(){
    document.documentElement.style.setProperty("--accent",
SETTINGS.accent||"#ffb02e");
    document.documentElement.style.setProperty("--accent-dark",
shadeColor(SETTINGS.accent||"#ffb02e", -10));
    document.documentElement.style.setProperty("--font-main",
SETTINGS.font||"Comic Sans MS, Poppins, sans-serif");
    document.body.style.fontFamily = SETTINGS.font;
}
applySavedTheme();

/* small util: color shade for accent dark */
function shadeColor(color, percent) {
    // color in #rrggbb
    const num = parseInt(color.slice(1),16);
    const r = (num >> 16) + percent;
    const g = (num >> 8 & 0x00FF) + percent;
    const b = (num & 0x0000FF) + percent;
    const newR = Math.min(255, Math.max(0, r));
```

```

const newG = Math.min(255, Math.max(0, g));
const newB = Math.min(255, Math.max(0, b));
return `#${(newR<<16 | newG<<8 |
newB).toString(16).padStart(6, '0')}`;

}

/* =====
Security: password hashing via Web Crypto
===== */
const CODE_KEY = "rvs_code_hash";
async function hashCode(str){
const enc = new TextEncoder().encode(str);
const hashBuffer = await crypto.subtle.digest('SHA-256', enc);
const hashArray = Array.from(new Uint8Array(hashBuffer));
const hashHex =
hashArray.map(b=>b.toString(16).padStart(2, '0')).join('');
return hashHex;
}
// on first load: if no code stored, store default hashed "67"
(async()=>{
if(!localStorage.getItem(CODE_KEY)){
const h = await hashCode("67");
localStorage.setItem(CODE_KEY, h);
}
})();

/* password modal handlers */
pwToggle.addEventListener("click", ()=>{
if(pwInput.type === "password") pwInput.type = "text"; else
pwInput.type = "password";
});
pwSubmit.addEventListener("click", async ()=>{
const attempt = pwInput.value.trim();
if(!attempt) return alert("Enter code");
const hashed = await hashCode(attempt);
if(hashed === localStorage.getItem(CODE_KEY)){
// unlocked
settingsContent.classList.remove("hidden");
document.getElementById("pwBlock").classList.add("hidden");
}
});

```

```

        buildSettingsList();
    } else {
        alert("Wrong code");
    }
});

/* open settings from header icon */
settingsIcon.addEventListener("click", ()=>openSettingsModal());
function openSettingsModal(){
    settingsModal.classList.remove("hidden");
    settingsModal.style.display = "flex";
    pwInput.value = "";
    settingsContent.classList.add("hidden");
    document.getElementById("pwBlock").classList.remove("hidden");
    buildSettingsList();
}
document.getElementById("closeSettingsBtn").addEventListener("click",
()=>{
    settingsModal.classList.add("hidden");
});

/* =====
SETTINGS: build question editor list
===== */
function buildSettingsList(filter=""){
    questionsList.innerHTML = "";
    DATA.forEach((q,i)=>{
        if(filter && !(q.question+q.options.join(" ") +
(q.fact|| "") ).toLowerCase().includes(filter.toLowerCase())) return;
        const card = document.createElement("div");
        card.className = "qcard";
        card.innerHTML =
            `<div style="flex:1">
                <label class="small">Video URL</label>
                <input type="text" class="sq_vid"
value="\${escapeHtml(q.video)}"/>
                <label class="small">Question</label>
                <input type="text" class="sq_q"
value="\${escapeHtml(q.question)}"/>
            `;
        questionsList.appendChild(card);
    });
}

```

```

        <label class="small">Options (comma separated)</label>
        <textarea
class="sq_opts">${escapeHtml(q.options.join(","))}</textarea>
        <label class="small">Correct Index (0-based)</label>
        <input type="number" class="sq_correct" value="${q.correct}"
min="0" style="width:80px"/>
        <label class="small">Hint</label>
        <input type="text" class="sq_hint"
value="${escapeHtml(q.hint||"")}" />
        <label class="small">Fact</label>
        <input type="text" class="sq_fact"
value="${escapeHtml(q.fact||"")}" />
    </div>
    <div class="q-actions">
        <button class="small" data-action="up"> </button>
        <button class="small" data-action="down"> </button>
        <button class="small" data-action="remove"
style="background:#e74c3c;color:white">Remove</button>
    </div>
`;
// attach handlers with closure index
const up = card.querySelector('[data-action="up"]');
const down = card.querySelector('[data-action="down"]');
const remove = card.querySelector('[data-action="remove"]');
up.addEventListener("click", ()=>{ if(i>0){ [DATA[i-1],DATA[i]]=[DATA[i],DATA[i-1]]; buildSettingsList(filter); }});
down.addEventListener("click", ()=>{ if(i<DATA.length-1){ [DATA[i+1],DATA[i]]=[DATA[i],DATA[i+1]]; buildSettingsList(filter); }});
remove.addEventListener("click", ()=>{ if(confirm("Remove this question?")){ DATA.splice(i,1); buildSettingsList(filter); } });
questionsList.appendChild(card);
});
}
searchQ && searchQ.addEventListener("input", ()=>
buildSettingsList(searchQ.value));

function addQuestion(){
DATA.push({id:"q"+Date.now(), difficulty:"easy", video:"",
```

```

question:"New question", options:["Option A","Option B","Option C"],
correct:0, hint:"", fact:""});
buildSettingsList();
}
document.getElementById("addQBtn").addEventListener("click",
addQuestion);

/* save settings / export / import */
document.getElementById("saveQBtn").addEventListener("click", ()=> {
// read back UI into DATA
const cards = questionsList.querySelectorAll('.qcard');
const newData=[];
cards.forEach(card=>{
    const vid = card.querySelector('.sq_vid').value.trim();
    const qtxt = card.querySelector('.sq_q').value.trim();
    const optsRaw = card.querySelector('.sq_opts').value.trim();
    const correct =
parseInt(card.querySelector('.sq_correct').value)||0;
    const hint = card.querySelector('.sq_hint').value.trim();
    const fact = card.querySelector('.sq_fact').value.trim();
    const opts = optsRaw.split(',').map(s=>s.trim()).filter(Boolean);
    if(qtxt && opts.length>=2){
        newData.push({ id:"q"+Date.now()+Math.random(),
difficulty:"easy", video:vid, question:qtxt, options:opts, correct:
Math.min(correct, opts.length-1), hint, fact });
    }
});
if(newData.length>0){
    DATA = newData;
    localStorage.setItem("rvs_quiz", JSON.stringify(DATA));
    alert("Saved. New question set will be used.");
    settingsModal.classList.add("hidden");
    resetGameToStart();
} else {
    alert("No valid questions found in editor.");
}
});

document.getElementById("exportBtn").addEventListener("click", ()=>{

```

```

const blob = new Blob([JSON.stringify(DATA, null,
2)],{type:"application/json"});
const url = URL.createObjectURL(blob);
const a = document.createElement("a");
a.href = url; a.download = "rvs_quiz_export.json"; a.click();
URL.revokeObjectURL(url);
});
document.getElementById("importBtn").addEventListener("click", ()=>
importFile.click());
importFile.addEventListener("change", (e)=> {
  const file = e.target.files[0];
  if(!file) return;
  const fr = new FileReader();
  fr.onload = ()=> {
    try{ const parsed = JSON.parse(fr.result);
    if(Array.isArray(parsed)){ DATA = parsed;
    localStorage.setItem("rvs_quiz", JSON.stringify(DATA));
    alert("Imported."); buildSettingsList(); resetGameToStart(); } else
    alert("JSON is not an array.");
    catch(err){ alert("Invalid JSON file."); }
  };
  fr.readAsText(file);
});

document.getElementById("resetDefault").addEventListener("click",
()=>{
  if(confirm("Reset to default questions? This will clear custom
questions.")){
    DATA = DEFAULT_DATA.slice();
    localStorage.removeItem("rvs_quiz");
    resetGameToStart();
  }
});
/* =====
 Game helper utilities
 ===== */
function escapeHtml(s){ return
s?.replaceAll?(."&","&").replaceAll?(."<","<").replaceAll?(.">"

```

```

,"&gt;") ?? s; }

function shuffle(arr){
  const a = arr.slice();
  for(let i=a.length-1;i>0;i--){
    const j = Math.floor(Math.random()*(i+1));
    [a[i],a[j]] = [a[j],a[i]];
  }
  return a;
}

/* =====
Game initialization & main loop
===== */
function buildPool(){
  let pool = DATA.slice();
  if(document.getElementById("shuffleQ").checked) pool =
shuffle(pool);
  state.order = pool;
  state.idx = 0;
  state.score = 0;
  state.timerSec = 10;
  state.stickers = JSON.parse(localStorage.getItem("rvs_stickers")) ||
[];
  document.getElementById("stickerCount").textContent = "Stickers:
"+state.stickers.length;
}
function loadQuestion(){
  if(state.idx >= state.order.length) { showResults(); return; }
  const q = state.order[state.idx];
  // load video and question
  videoFrame.src = q.video;
  questionText.textContent = q.question;
  progressText.textContent = `Question ${state.idx+1} of
${state.order.length}`;
  // build options (shuffled)
  optionsArea.innerHTML = "";
  const shuffled = shuffle(q.options.map((opt, i)=>({opt,i})));
  shuffled.forEach((o, aidx)=>{

```

```

const div = document.createElement("div");
div.className = "opt";
div.setAttribute("role","listitem");
div.innerHTML = `<div
class="label">${String.fromCharCode(65+aidx)}</div><div
style="flex:1">${o.opt}</div>`;
div.addEventListener("click", ()=> handleAnswer(div, q, o.i));
optionsArea.appendChild(div);
});
// reset UI
nextBtn.classList.remove("show");
messageDiv.textContent = "";
buddySetIdle();
updateLevelBar();
// timer
clearTimer();
if(document.getElementById("timerToggle").checked){
  startTimer();
} else { timerDisplay.textContent = "";}
// save last played
localStorage.setItem("rvs_lastPlayed", JSON.stringify({ idx:
state.idx, timestamp: Date.now() }));
}

/* visual level bar */
function updateLevelBar(){
  const pct = Math.round((state.idx / Math.max(1, state.order.length))
* 100);
  levelFill.style.width = pct + "%";
}

/* answer handling */
function handleAnswer(div, q, chosenIndex){
  // disable all options
  const all = optionsArea.querySelectorAll(".opt");
  all.forEach(o=>o.style.pointerEvents="none");
  clearTimer();
  const correct = q.correct;
  if(chosenIndex === correct){

```

```
div.classList.add("correct");
if(!state.mute) s_correct.play();
// sticker reward: random sticker emoji
const newSticker = randomSticker();
state.stickers.push(newSticker);
showFloatingSticker(newSticker);
// collect fact into factBook (only on correct)
addFact(q.fact);
state.score += 1;
// buddy animation positive
buddyCelebrate();
messageDiv.textContent = "Great job! " + (q.fact || "");
state.streak = (state.streak || 0) + 1;
} else {
    div.classList.add("wrong");
    // reveal correct option visually
    all.forEach((el, idx) => {
        const txt = el.textContent.trim();
        // find which one matches q.options[correct]
        if(el.textContent.includes(q.options[correct]))
            el.classList.add("correct");
    });
    if(!state.mute) s_wrong.play();
    messageDiv.textContent = "Oops! " + (q.fact || "");
    state.streak = 0;
}
// show next
nextBtn.classList.add("show");
// save stickers and streak/best
localStorage.setItem("rvs_stickers",
JSON.stringify(state.stickers));
localStorage.setItem("rvs_streak", state.streak);
if(state.score > state.best){ state.best = state.score;
localStorage.setItem("rvs_best", state.best); }
// update counts display
document.getElementById("stickerCount").textContent = "Stickers: " +
state.stickers.length;
streakCount.textContent = state.streak;
bestScoreDisplay.textContent = state.best;
```

```

}

/* next button */
nextBtn.addEventListener("click", ()=>{
  state.idx++;
  if(state.idx < state.order.length) {
    updateLevelBarAnimate();
    loadQuestion();
  } else {
    showResults();
  }
});

/* hint button */
hintBtn.addEventListener("click", ()=>{
  const q = state.order[state.idx];
  if(!q) return;
  // penalty: deduct one point if available (but not below 0)
  if(state.score > 0){ state.score = Math.max(0, state.score - state.hintPenalty); }
  messageDiv.textContent = "💡 " + (q.hint || "Think carefully!");
  // deduct sticker if any exist (choose: cost 1 sticker if present)
  if(state.stickers.length>0) state.stickers.pop();
  localStorage.setItem("rvs_stickers",
  JSON.stringify(state.stickers));
  document.getElementById("stickerCount").textContent = "Stickers: " +
  state.stickers.length;
});

/* show results with confetti and results menu */
function showResults(){
  // confetti
  runConfetti();
  if(!state.mute) s_end.play();

  // update best score
  if(state.score > state.best) {
    state.best = state.score;
    localStorage.setItem("rvs_best", state.best);
  }
}

```

```

        bestScoreDisplay.textContent = state.best;
    }

    // populate results modal
    const percentage = Math.round((state.score / state.order.length) * 100);
    resultsScore.textContent = `${state.score}/${state.order.length}`;

    // Set message based on performance
    let message = "";
    if(percentage >= 80) {
        message = "Amazing! You're a learning star! 🌟";
    } else if(percentage >= 60) {
        message = "Great job! Keep up the good work! 🤝";
    } else if(percentage >= 40) {
        message = "Good effort! Practice makes perfect! 🤸";
    } else {
        message = "Keep trying! You'll get better! 🌈";
    }
    resultsMessage.textContent = message;

    // Display stickers
    stickerDisplay.textContent = state.stickers.length > 0 ? state.stickers.join(" ") : "No stickers earned yet";

    // Display facts
    const facts = JSON.parse(localStorage.getItem("rvs_facts") || "[]");
    factsList.innerHTML = "";
    if(facts.length === 0) {
        factsList.innerHTML = "<li>No facts collected yet</li>";
    } else {
        facts.forEach(fact => {
            const li = document.createElement("li");
            li.textContent = fact;
            factsList.appendChild(li);
        });
    }

    // Show results modal

```

```

resultsModal.classList.remove("hidden");
resultsModal.style.display = "flex";
}

/* reset game to start */
function resetGameToStart(){
buildPool();
renderFactBook();
loadQuestion();
}

/* initial build & start */
document.getElementById("shuffleQ").addEventListener("change", ()=>
SETTINGS.shuffle = document.getElementById("shuffleQ").checked );
document.getElementById("timerToggle").addEventListener("change", ()=>
{ /* nothing immediate */ });
document.getElementById("muteBtn").addEventListener("click", ()=>
{ state.mute = !state.mute; muteBtn.textContent = state.mute ? '🔇' :
'🔈' });
}

document.getElementById("nameBtn").addEventListener("click", ()=>{
const n = prompt("Enter child's name (for results)", "");
if(n){ state.childName = n; localStorage.setItem("rvs_childName", n); alert("Saved name: "+n); }
});

/* name modal handlers */
submitName.addEventListener("click", ()=>{
const name = nameInput.value.trim();
if(name){
state.childName = name;
localStorage.setItem("rvs_childName", name);
nameModal.classList.add("hidden");
loadQuestion();
} else {
alert("Please enter your name");
}
});

```

```

/* game controls modal handlers */
openControls.addEventListener("click", ()=>{
    controlsModal.classList.remove("hidden");
    controlsModal.style.display = "flex";
});

closeControls.addEventListener("click", ()=>{
    controlsModal.classList.add("hidden");
});

/* results modal handlers */
playAgainBtn.addEventListener("click", ()=>{
    resultsModal.classList.add("hidden");
    resetGameToStart();
});

changeNameBtn.addEventListener("click", ()=>{
    resultsModal.classList.add("hidden");
    nameModal.style.display = "flex";
    nameInput.value = "";
    nameInput.focus();
});

/* settings header icon opens modal but triggers pw flow earlier */
settingsIcon.addEventListener("dblclick", ()=> openSettingsModal());
// doubleclick alternate

/* random sticker emoji generator */
const STICKERS =
["⭐️","🎉","🌟","🌈","🍀","🦄","⭐️","🎀","🌟","🍎","🎲"];
function randomSticker(){ return
STICKERS[Math.floor(Math.random()*STICKERS.length)]; }
function showFloatingSticker(emoji){
    const el = document.createElement("div");
    el.className = "floating";
    el.style.left = (30 + Math.random()*60) + "vw";
    el.textContent = emoji;
    document.body.appendChild(el);
    setTimeout(()=> el.remove(), 1600);
}

```

```

}

/* buddy animations */
function buddyCelebrate(){
  buddyEl.classList.add("jump");
  setTimeout(()=> buddyEl.classList.remove("jump"), 600);
}
function buddySetIdle(){ buddyEl.classList.remove("jump", "shake"); }

/* timer */
function startTimer(){
  let sec = 10;
  timerDisplay.textContent = sec + "s";
  state.timer = setInterval(()=>{
    sec--;
    timerDisplay.textContent = sec + "s";
    if(sec<=0){
      clearTimer();
      // time's up: auto mark wrong and reveal
      messageDiv.textContent = "Time's up!";
      // simulate picking nothing: reveal correct
      const all = optionsArea.querySelectorAll(".opt");
      all.forEach(el=>{

        if(el.textContent.includes(state.order[state.idx].options[state.order[state.idx].correct])) el.classList.add("correct");
        el.style.pointerEvents = "none";
      });
      if(!state.mute) s_wrong.play();
      nextBtn.classList.add("show");
    }
  },1000);
}
function clearTimer(){ if(state.timer){ clearInterval(state.timer);
state.timer=null; } timerDisplay.textContent = ""; }

/* confetti implementation on canvas (simple) */
function runConfetti(){
  const canvas = confettiCanvas;

```

```

const ctx = canvas.getContext('2d');
canvas.width = window.innerWidth;
canvas.height = window.innerHeight;
const pieces = [];
for(let i=0;i<80;i++){
  pieces.push({
    x: Math.random()*canvas.width,
    y: Math.random()*-canvas.height,
    r: 6+Math.random()*8,
    color: `hsl(${Math.random()*360},70%,60%)`,
    vx: -2 + Math.random()*4,
    vy: 2 + Math.random()*6,
    rot: Math.random()*360
  });
}
let t=0;
function frame(){
  t++;
  ctx.clearRect(0,0,canvas.width,canvas.height);
  for(const p of pieces){
    p.x += p.vx; p.y += p.vy; p.rot += 4;
    ctx.save();
    ctx.fillStyle = p.color;
    ctx.translate(p.x, p.y);
    ctx.rotate(p.rot * Math.PI/180);
    ctx.fillRect(-p.r/2, -p.r/2, p.r, p.r);
    ctx.restore();
  }
  if(t<120) requestAnimationFrame(frame);
  else ctx.clearRect(0,0,canvas.width,canvas.height);
}
frame();
}

/* fact book */
function addFact(f){
  if(!f) return;
  const facts = JSON.parse(localStorage.getItem("rvs_facts")||"[]");
  if(!facts.includes(f)) facts.push(f);
}

```

```

localStorage.setItem("rvs_facts", JSON.stringify(facts));
renderFactBook();
}
function renderFactBook(){
  const facts = JSON.parse(localStorage.getItem("rvs_facts")||"[]");
  if(facts.length==0) factList.textContent = "No facts yet – answer
correctly to collect them.";
  else factList.innerHTML = facts.map(f=>`

${escapeHtml(f)}

`).join('');
}

/* UI: show settings modal from header icon (opens password UI) */
settingsIcon.addEventListener("click", ()=> {
  settingsModal.classList.remove("hidden");
});
document.getElementById("closeSettingsBtn").addEventListener("click",
()=> settingsModal.classList.add("hidden"));

/* reset game / start */
document.addEventListener("DOMContentLoaded", ()=>{
  // Check if name is already saved
  if(!state.childName) {
    // Show name modal
    nameModal.style.display = "flex";
  } else {
    // Name already exists, start game
    buildPool();
    renderFactBook();
    loadQuestion();
  }
  // load persisted theme
  document.getElementById("stickerCount").textContent = "Stickers: " +
(state.stickers.length||0);
});

/* helper: reset everything and start new */
function resetAll(){
  state.idx = 0; state.score = 0; state.stickers = [];
  localStorage.removeItem("rvs_stickers");
}

```

```

localStorage.removeItem("rvs_streak");
}

/* reset to start after game or save */
function resetGameToStart(){
    state.idx = 0; state.score = 0;
    buildPool();
    loadQuestion();
}

/* timer/bonus ui animation for level bar */
function updateLevelBarAnimate(){ levelFill.style.transition =
'width .35s ease-out'; updateLevelBar(); setTimeout(()=>
levelFill.style.transition='width .35s ease-out', 400); }

/* small helpers */
function escapeHtml(s){ if(!s) return ""; return
s.replaceAll("<","&lt;").replaceAll(">","&gt;"); }

/* end of script */
</script>
</body>
</html>

```

I've updated the game with the new questions you provided. The changes include:

1. **Updated the DEFAULT\_DATA array** with all 6 questions from your JSON, including the new question about brushing teeth (q6)
2. **Updated the progress text** to show "Question 1 of 6" instead of "Question 1 of 5" to reflect the new total number of questions
3. **Maintained all existing functionality** including the name modal, game controls modal, and results menu

The game now includes the additional question about brushing teeth and will display the correct total number of questions throughout the game. All other features remain unchanged.

## USER

Give some supportive feedback.

**My Stickers:** None yet!

### Your Results

Learn right from wrong while having fun!

## ASSISTANT

Thought for 3 seconds > This appears to be a web page for a children's educational app called "Right vs. Wrong: Kids Action Learning App". The app seems to be designed to teach children about appropriate behaviors through video scenarios and multiple-choice questions. Let me analyze the code and provide supportive feedback on this educational tool. >> The app includes: > 1. A video player showing scenarios of children's behaviors > 2. Multiple-choice questions about what was done "wrong" in each scenario > 3. Visual feedback (colors, animations) for correct/incorrect answers > 4. A progress tracking system with stickers and stars as rewards > 5. A results screen with performance summary > 6. Sound effects for engagement > 7. Local storage to save progress >> Let me provide supportive feedback on various aspects of this educational app. # Supportive Feedback for "Right vs. Wrong: Kids Action Learning App"

## Overall Impression

This is a well-designed educational app that effectively combines visual learning with interactive elements to teach children about appropriate behaviors. The concept is strong and the implementation shows thoughtful consideration for the target audience.

## **Strengths**

### **1. User-Friendly Interface**

- The colorful, playful design with rounded corners and gradient backgrounds creates an inviting atmosphere for children
- The Comic Sans font choice is appropriate for younger learners
- Clear visual hierarchy makes it easy to navigate through the quiz

### **2. Engaging Learning Experience**

- Video-based scenarios provide concrete examples that children can relate to
- Immediate feedback with color coding (green for correct, red for incorrect) helps reinforce learning
- The star animations and confetti effects create positive reinforcement moments

### **3. Progress Tracking**

- The sticker collection system gamifies the learning experience
- The progress bar gives children a sense of accomplishment
- Local storage implementation allows for continuity across sessions

### **4. Accessibility Features**

- Responsive design ensures the app works on different screen sizes
- Clear visual indicators for correct/incorrect answers help children with different learning styles

## **Suggestions for Enhancement**

### **1. Content Expansion**

- Consider adding more diverse scenarios covering different social situations
- You might include scenarios about online behavior, safety, and emotional intelligence
- Adding a "why" explanation after each answer could deepen the learning

## **2. Adaptive Difficulty**

- Implement a system that adjusts difficulty based on the child's performance
- Consider adding age-appropriate content levels

## **3. Parental Features**

- A parent dashboard to track their child's progress and areas of improvement
- Options to customize content based on family values or specific situations

## **4. Accessibility Improvements**

- Add text-to-speech functionality for questions and options
- Consider high contrast mode options for children with visual impairments

## **Technical Strengths**

- Clean, well-organized code structure
- Efficient use of CSS variables for consistent theming
- Good use of localStorage for data persistence
- Smooth animations and transitions enhance the user experience

## **Educational Value**

The app effectively uses the "show, don't tell" approach by demonstrating behaviors through videos rather than just describing them. This visual learning approach is particularly effective for children who are still developing abstract thinking skills.

## **Conclusion**

This is a solid educational tool with a strong foundation. The combination of visual scenarios, interactive questions, and gamification elements creates an engaging learning environment. With some content expansion and the addition of parental features, this could become an even more valuable resource for teaching children about appropriate behaviors.

The thoughtful design and implementation show a clear understanding of both the technical requirements and the educational needs of the target audience. Great work on creating a tool that makes learning about right and wrong behaviors fun and engaging for children!

What improvements i can add. like with the settings panel

```
<!DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width, initial-scale=1.0"> <title>Right vs Wrong: Kids Learning Game</title> <style> :root{ --accent:#ffb02e;--accent-dark:#ff9b00;--green:#1fb16a;--red:#e74c3c;--muted:#6b7280; font-family:"Comic Sans MS","Poppins",sans-serif; } body{margin:0;background:linear-gradient(180deg,#f7fbff 0%,#ffffff 100%);color:#0f172a;} header{background:linear-gradient(90deg,var(--accent),#ffd58f);color:white;text-align:center;padding:16px;font-size:22px;font-weight:bold;border-bottom-left-radius:14px; border-bottom-right-radius:14px; box-shadow:0 8px 20px rgba(0,0,0,0.1);position:relative;} #settingsIcon{position:absolute;right:14px;top:12px;background:rgba(255,255,255,0.3);border-radius:50%; width:36px;height:36px;display:flex;align-items:center;justify-content:center;font-size:20px;cursor:pointer;} main{display:grid;grid-template-columns:1fr 420px;gap:20px;padding:16px;max-width:1000px;margin:auto;} @media(max-width:850px){main{grid-template-columns:1fr;}} iframe{height:240px!important;} .video-box{background:white;border-radius:16px;overflow:hidden;box-shadow:0 8px 20px rgba(0,0,0,0.1);} .video-box iframe{width:100%;height:320px;border:0;} .question-box{background:white;border:4px solid #fff2d9;border-radius:16px;padding:16px; box-shadow:0 8px 20px rgba(0,0,0,0.08);} .option{background:white;border:2px solid #fff0d6;border-radius:10px;padding:10px 12px;margin:8px 0; cursor:pointer;font-size:16px;box-shadow:0 4px 10px rgba(0,0,0,0.05);transition:0.2s;} .option:hover{background:#fff9e9;} .option.correct{background:#e6fff1;border-color:var(--green);} .option.wrong{background:#ffeaea;border-color:var(--red);} button{border:none;border-radius:10px;padding:10px 18px;font-size:16px;font-weight:bold;cursor:pointer;} .next-btn{background:linear-gradient(90deg,var(--accent),var(--accent-dark));color:white;display:none;} .next-btn.show{display:inline-block;margin-top:10px;} .hint-btn{background:#007bff;color:white;margin-top:6px;} .progress{font-size:13px;color:var(--muted);margin-bottom:8px;} .level-bar{height:10px;background:#e5e7eb;border-radius:8px;overflow:hidden;margin-bottom:10px;} .level-fill{height:100%;background:linear-gradient(90deg,var(--accent),var(--accent-dark));width:0;} #stickers{background:#fff9e9;border:3px dashed var(--accent);padding:8px;border-radius:12px;margin-top:10px;font-size:14px;} #stickerList{font-size:18px;} .message{color:var(--green);font-weight:bold;text-align:center;margin-top:6px;font-size:18px;} .buddy{font-size:32px;text-align:center;margin:8px 0;} .results{text-align:center;padding:16px;} .results h2{color:var(--accent-dark);} .badge{display:inline-block;background:#fff9e9;border:2px solid var(--accent-dark);width:fit-content;outline:none;outline-offset:2px;}
```

```
accent);border-radius:12px;padding:6px 12px;margin:4px;font-weight:bold;} footer{text-align:center;color:var(--muted);margin:20px 0 40px;} .settings{padding:20px;text-align:center;} .settings input, .settings textarea{width:90%;padding:8px;margin:4px 0;border-radius:8px;border:1px solid #ccc;font-size:14px;} .settings button{background:var(--accent);color:white;margin-top:8px;} .hidden{display:none;}</style></head><body><header> Right vs Wrong: Kids Learning Game <div id="settingsIcon" title="Settings">⚙️</div></header><!-- SETTINGS PANEL (Secure) --><section class="settings hidden" id="settingsPanel"><h2>🔒 Settings Panel (Teacher Mode)</h2><p>Edit or manage questions. You can also import/export as JSON.</p><div id="configArea"></div><div><button onclick="addQuestion()">➕ Add Question</button><button onclick="saveCustomQuestions()">💾 Save</button><button onclick="exportJSON()">⬇️ Export JSON</button><input type="file" id="jsonFile" accept="application/json" onchange="importJSON(event)" style="margin-top:6px;"></div><button onclick="closeSettings()" style="background:#e74c3c;">✖ Close</button></section><!-- MAIN GAME --><main id="quiz"><div class="video-box"><iframe id="videoFrame" src="" allowfullscreen></iframe></div><div class="question-box"><div class="progress" id="progress"></div><div class="level-bar"><div class="level-fill" id="levelFill"></div></div><h2 id="question"></h2><div id="options"></div><button class="hint-btn" id="hintBtn">💡 Hint</button><div class="buddy" id="buddy">😊</div><button class="next-btn" id="nextBtn">Next</button><div id="stickers"><strong>My Stickers:</strong> <span id="stickerList">None yet!</span></div><div class="message" id="message"></div></div></main><!-- RESULTS --><section class="results hidden" id="results"><h2>🎉 Results 🎉</h2><p id="scoreText"></p><div id="badgeArea"></div><div id="encourage"></div><div id="stickerAlbum"></div><button class="next-btn show" onclick="restart()">Play Again</button></section><footer>Learn, Laugh, and Do the Right Thing!</footer><audio id="sound-correct" src="https://actions.google.com/sounds/v1/cartoon/pop.ogg"></audio><audio id="sound-wrong" src="https://actions.google.com/sounds/v1/cartoon/wood_plank_flicks.ogg"></audio><audio id="sound-end" src="https://actions.google.com/sounds/v1/cartoon/slide_whistle_to_drum_hit.ogg"></audio><script> /* ===== DEFAULT QUESTIONS ===== */ let scenarios = JSON.parse(localStorage.getItem("quizData")) || [ { video: "https://www.youtube.com/embed/_7R4SflsINU", question: "What did the kid do wrong after finishing their snack?", options: ["Threw the wrapper on the ground", "Put the wrapper in the bin", "Kept the wrapper in their pocket"], correct: 0, hint: "Think about" } ]</script>
```

```
trash!",fact:"Always put trash in the bin."},  
{video:"https://www.youtube.com/embed/dQw4w9WgXcQ",question:"What mistake did  
the kid make while washing hands?", options:["Left the water running","Turned off the  
tap","Used too much soap"],correct:0,hint:"Think about water!",fact:"Turning off the tap  
saves water."}, {video:"https://www.youtube.com/embed/M7lc1UVf-VE",question:"What  
did the kid forget before leaving their room?", options:["Forgot to turn off the  
lights","Closed the window","Put away toys"],correct:0,hint:"Think about saving  
electricity.",fact:"Turn off lights to save energy."},  
{video:"https://www.youtube.com/embed/ScMzIvxBSi4",question:"What should you do  
before crossing the road?", options:["Look both ways","Run quickly","Close your  
eyes"],correct:0,hint:"Think about being careful!",fact:"Always look both ways."},  
{video:"https://www.youtube.com/embed/tgbNymZ7vqY",question:"What is wrong if you  
don't wash fruits?", options:["They can have dirt","They taste bad","They get  
soft"],correct:0,hint:"Think cleanliness.",fact:"Always wash fruits to stay healthy."} ];  
let  
current=0,score=0,stickers=[]; const  
videoFrame=document.getElementById("videoFrame"),  
questionEl=document.getElementById("question"),  
optionsEl=document.getElementById("options"),  
progressEl=document.getElementById("progress"),  
nextBtn=document.getElementById("nextBtn"),  
hintBtn=document.getElementById("hintBtn"), quiz=document.getElementById("quiz"),  
results=document.getElementById("results"),  
scoreText=document.getElementById("scoreText"),  
stickerList=document.getElementById("stickerList"),  
levelFill=document.getElementById("levelFill"),  
message=document.getElementById("message"),  
soundCorrect=document.getElementById("sound-correct"),  
soundWrong=document.getElementById("sound-wrong"),  
soundEnd=document.getElementById("sound-end"),  
encourage=document.getElementById("encourage"),  
stickerAlbum=document.getElementById("stickerAlbum"),  
buddy=document.getElementById("buddy"),  
settingsPanel=document.getElementById("settingsPanel"),  
settingsIcon=document.getElementById("settingsIcon"),  
configArea=document.getElementById("configArea"); let settingsUnlocked=false;  
settingsIcon.addEventListener("click",()=>{ const input=prompt("Enter teacher code:");  
if(input==="67"){settingsUnlocked=true;toggleSettings(true);} else alert("Wrong code!"); });  
function toggleSettings(state){ settingsPanel.classList.toggle("hidden",!state); if(state)
```

```

buildSettings(); } /* ===== SETTINGS PANEL ===== */ function
buildSettings(){ configArea.innerHTML=""; scenarios.forEach((s,i)=>{ const
div=document.createElement("div"); div.style="border:2px dashed
#ffd58f;padding:10px;margin:8px;border-radius:10px;text-align:left;"; div.innerHTML=
<h3>Question ${i+1}</h3> <label>Video URL:</label><input value="${s.video}"
id="vid${i}"> <label>Question:</label><input value="${s.question}" id="q${i}">
<label>Options (comma-separated):</label><textarea
id="a${i}">${s.options.join(",")}</textarea> <label>Correct Answer Index (0-
based):</label><input value="${s.correct}" id="c${i}" type="number" min="0">
<label>Hint:</label><input value="${s.hint}" id="h${i}"> <label>Fact:</label><input
value="${s.fact}" id="f${i}"> <button style="background:#e74c3c;color:white;margin-
top:4px;" onclick="removeQuestion(${i})">Remove</button>;
configArea.appendChild(div);}); } function
addQuestion(){ scenarios.push({video:"",question:"",options:[""],correct:0,hint:"",fact:""}); buildSettings(); } function removeQuestion(i){scenarios.splice(i,1);buildSettings();} function saveCustomQuestions(){ if(!settingsUnlocked) return; const newSet=[]; for(let i=0;i<configArea.children.length;i++){ const
vid=document.getElementById(`vid${i}`);if(!vid)continue; const
q=document.getElementById(`q${i}`).value; const
a=document.getElementById(`a${i}`).value.split(","); const
c=parseInt(document.getElementById(`c${i}`).value); const
h=document.getElementById(`h${i}`).value; const
f=document.getElementById(`f${i}`).value;
if(q&&a.length>=2)newSet.push({video:vid.value,question:q,options:a,correct:c,hint:h,fac
t:f}); } scenarios=newSet.length?newSet:scenarios;
localStorage.setItem("quizData",JSON.stringify(scenarios)); alert("Questions saved!");
toggleSettings(false); } function closeSettings(){toggleSettings(false);} /* JSON
IMPORT/EXPORT */ function exportJSON(){ const dataStr="data:text/json;charset=utf-
8,"+encodeURIComponent(JSON.stringify(scenarios,null,2)); const
dl=document.createElement("a"); dl.setAttribute("href",dataStr);
dl.setAttribute("download","quiz_data.json"); dl.click(); } function importJSON(e){ const
file=e.target.files[0]; if(!file) return; const reader=new FileReader();
reader.onload=(ev)=>{ try{ const data=JSON.parse(ev.target.result);
if(Array.isArray(data)){scenarios=data;localStorage.setItem("quizData",JSON.stringify(sce
narios));alert("Imported successfully!");buildSettings();} else alert("Invalid JSON
format."); }catch{alert("Error parsing JSON.");} }; reader.readAsText(file); } /*
===== GAME LOGIC ===== */ function shuffle(arr){for(let
i=arr.length-1;i>0;i--){const

```

```

j=Math.floor(Math.random()*(i+1));[arr[i],arr[j]]=[arr[j],arr[i]];}return arr;} function
loadScenario(){ scenarios=shuffle(scenarios); const s=scenarios[current];
videoFrame.src=s.video; questionEl.textContent=s.question;
progressEl.textContent=Question ${current+1} of ${scenarios.length};
optionsEl.innerHTML="";nextBtn.classList.remove("show");message.textContent="";
buddy.textContent=" 😊 "; const shuffledOptions=shuffle([...s.options]);
shuffledOptions.forEach(opt=>{ const div=document.createElement("div");
div.textContent=opt;div.className="option";
div.onclick=()=>selectOption(div,s.options.indexOf(opt)); optionsEl.appendChild(div); });
updateLevelBar();updateStickers(); hintBtn.onclick=()=>message.textContent="💡 Hint:
"+s.hint; } function selectOption(div,i){ const s=scenarios[current]; const
all=document.querySelectorAll(".option"); all.forEach(o=>o.style.pointerEvents="none");
if(i==s.correct){ div.classList.add("correct");
score++;stickers.push("⭐");soundCorrect.play(); message.textContent=" ✅
"+s.fact;buddy.textContent=" 😊 "; }else{ div.classList.add("wrong");
all.forEach((o,idx)=>{if(idx==s.correct)o.classList.add("correct");});
soundWrong.play();message.textContent=" ❌ "+s.fact;buddy.textContent=" 🙄 ";
} nextBtn.textContent=current==scenarios.length-1?"See Results":"Next";
nextBtn.classList.add("show"); } function
updateStickers(){stickerList.textContent=stickers.length?stickers.join(" "):"None yet!";
} function updateLevelBar(){levelFill.style.width=((current)/scenarios.length)*100+"%";
} nextBtn.onclick=()=>{current++;if(current<scenarios.length)loadScenario();else
showResults();}; function
showResults(){ quiz.classList.add("hidden");results.classList.remove("hidden");soundEnd.
play(); let msg=(score==scenarios.length)?"🏆 Perfect!":" 😊 Great job!";
encourage.textContent=msg; scoreText.innerHTML=You got
<strong>$ {score}</strong> of ${scenarios.length}!;
stickerAlbum.innerHTML=<h3>Your Stickers:</h3>"+(stickers.length?stickers.join(
"):"No stickers yet!"); badgeArea.innerHTML=getBadge(); } function
getBadge(){ if(score==scenarios.length)return"<div class='badge'>🏆 Master of Good
Choices</div>"; if(score>=3)return"<div class='badge'>🌱 Kind Helper</div>"; return"<div
class='badge'>💡 Learning Star</div>"; } function
restart(){current=0;score=0;stickers=[];results.classList.add("hidden");quiz.classList.remov
e("hidden");loadScenario();} /* INIT */ loadScenario(); </script> </body> </html> and
others

```

*ChatGPT said:*

yo okay ngl moath this game is actually *fire* 🔥 — like it's got structure, cuteness, and logic, but the **settings panel and gameplay UX** could go from “teacher-made project” → “actual polished webapp” with just a few tweaks 🎨

here's a bunch of **improvement ideas**, grouped by area so u can pick what to add 👇

## ⚙️ SETTINGS PANEL UPGRADES

### 1. Password toggle

- a. Instead of prompt ("Enter teacher code"), make a proper password input popup (with show/hide toggle 🕵️).
- b. Could store a hashed code in localStorage to prevent casual kids from snooping 🐱

### 2. Question reorder

- a. Add little “↑ / ↓” arrows in the settings to move questions up/down without deleting them.

### 3. Preview mode

- a. Add a “👀 Preview Question” button that shows how the question/video will look before saving.

### 4. Search bar / filter

- a. If there are lots of questions, add a mini search bar to filter questions by text.

### 5. Theme customizer

- a. Let teacher change accent color, font, or emoji buddy face sets.

### 6. Reset to default

- a. “🔄 Reset to Default Questions” button that clears custom JSON and reloads defaults.

## 🎮 GAMEPLAY IMPROVEMENTS

### 1. Question progress animation

- a. Animate the level bar filling up with a smooth transition each question (e.g., .3s ease-out).

## **2. Difficulty levels**

- a. Add “Easy / Medium / Hard” filters that load different sets of scenarios.

## **3. Audio toggle**

- a. Small mute/unmute icon for sounds (cartoon pops can get annoying after 5 min ).

## **4. Buddy reactions**

- a. Make the buddy (  ) actually animate (shake, jump, or sparkle when correct).

## **5. Sticker upgrade**

- a. Each correct answer gives random stickers (   etc.), maybe show them floating when earned.

## **6. Timer / streak system**

- a. Optional challenge: answer within 10s for bonus points or streaks.

## **7. End-screen confetti**

- a. When showing results, add confetti particles or floating emojis with canvas or CSS animation.



## **EDUCATIONAL FEATURES**

### **1. Fun facts collection**

- a. After the game, unlock a “fact book” that collects all facts from correct answers.

### **2. Progress saving**

- a. Save score, streaks, or last played question in localStorage so kids can continue next time.

### **3. Randomized question pool**

- a. Option in settings: “Shuffle questions each game” or “Fixed order.”

### **4. Hint penalty**

- a. Deduct small score when hints are used, encourages kids to think first.



## **UI/UX POLISH**

### **1. Smooth transitions**

- a. Fade between quiz → results → settings (CSS transitions or JS `classList.add('fade')`).
- 2. **Rounded colorful buttons**
  - a. Different colors for hint / next / settings, and hover effects with small bounce animation.
- 3. **Responsive fixes**
  - a. On phones, make iframe height auto-adjust (maybe 56.25% aspect ratio wrapper).
- 4. **Accessibility**
  - a. Add `aria-labels` for screen readers and bigger text option in settings.

## EXTRA IDEAS

- Add a “**Random Buddy Mode**” (random emoji character each time, like 🐸 🐻 🐧).
- Let kids **choose a name/avatar** that appears on the results screen.
- Add **background music** toggle (chill looped track).
- Use **Lottie animations** instead of static emojis for buddy reactions.
- Include a “**Classroom Mode**” — teachers can run it fullscreen and show score on a projector.

if u want i can literally help u code a ✨ clean, animated, settings panel ✨ with reorder, theme, and import/export buttons — like proper modern UI, still simple.  
wanna do that next?

You said:ChatGPT said: