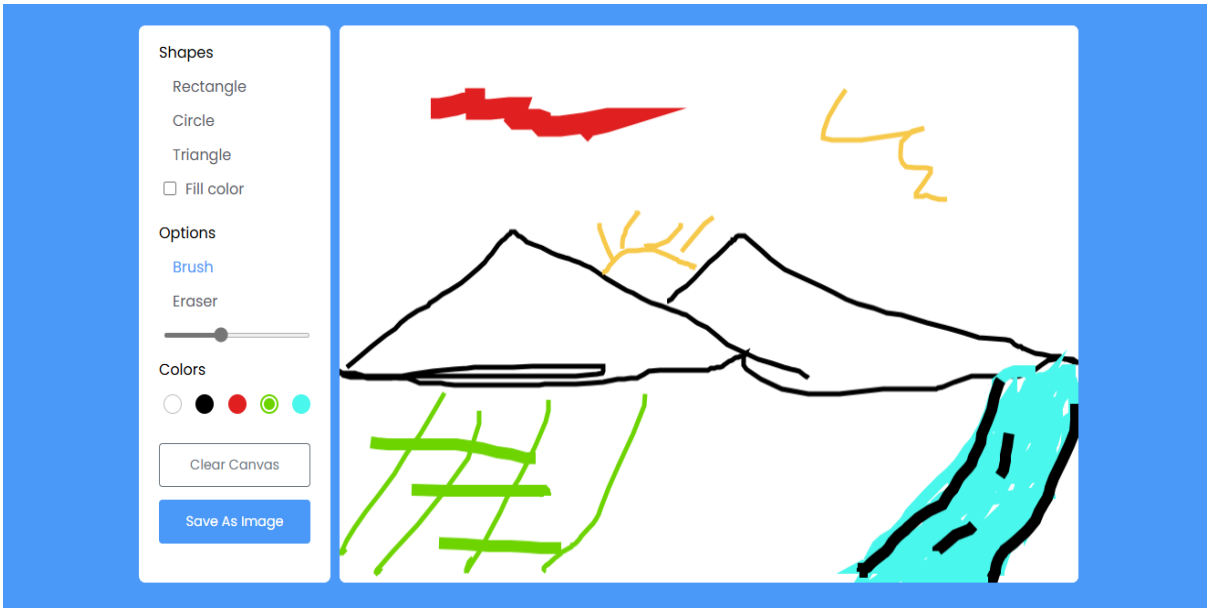
3 Game HTML, CSS, JAVASCRIPT

1. Drawing Game



index.html

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>Drawing App JavaScript | CodingNepal</title>
    <link rel="stylesheet" href="style.css">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <script src="script.js" defer></script>
  </head>
  <body>
    <div class="container">
      <section class="tools-board">
        <div class="row">
          <label class="title">Shapes</label>
          <ul class="options">
            <li class="option tool" id="rectangle">
              <img src="icons/rectangle.svg" alt="">
              <span>Rectangle</span>
            </li>
            <li class="option tool" id="circle">
              <img src="icons/circle.svg" alt="">
              <span>Circle</span>
            </li>
            <li class="option tool" id="triangle">
              <img src="icons/triangle.svg" alt="">
              <span>Triangle</span>
```

```
            </li>
            <li class="option">
              <input type="checkbox" id="fill-color">
              <label for="fill-color">Fill color</label>
            </li>
          </ul>
        </div>
        <div class="row">
          <label class="title">Options</label>
          <ul class="options">
            <li class="option active tool" id="brush">
              <img src="icons/brush.svg" alt="">
              <span>Brush</span>
            </li>
            <li class="option tool" id="eraser">
              <img src="icons/eraser.svg" alt="">
              <span>Eraser</span>
            </li>
            <li class="option">
              <input type="range" id="size-slider" min="1" max="30" value="5">
            </li>
          </ul>
        </div>
        <div class="row colors">
          <label class="title">Colors</label>
          <ul class="options">
            <li class="option"></li>
            <li class="option selected"></li>
            <li class="option"></li>
            <li class="option"></li>
            <li class="option">
              <input type="color" id="color-picker" value="#4A98F7">
            </li>
          </ul>
        </div>
        <div class="row buttons">
          <button class="clear-canvas">Clear Canvas</button>
          <button class="save-img">Save As Image</button>
        </div>
      </section>
      <section class="drawing-board">
        <canvas></canvas>
      </section>
    </div>

  </body>
</html>
```

style.css

```
@import
url('https://fonts.googleapis.com/css2?family=Poppins:wght@400;500;600&display
=swap');
*{

*{
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  font-family: 'Poppins', sans-serif;
}
body{
  display: flex;
  align-items: center;
  justify-content: center;
  min-height: 100vh;
  background: #4A98F7;
}
.container{
  display: flex;
  width: 100%;
  gap: 10px;
  padding: 10px;
  max-width: 1050px;
}
section{
  background: #fff;
  border-radius: 7px;
}
.tools-board{
  width: 210px;
  padding: 15px 22px 0;
}
.tools-board .row{
  margin-bottom: 20px;
}
.row .options{
  list-style: none;
  margin: 10px 0 0 5px;
}
.row .options .option{
  display: flex;
  cursor: pointer;
  align-items: center;
  margin-bottom: 10px;
}
```

```css
.option:is(:hover, .active) img{
  filter: invert(17%) sepia(90%) saturate(3000%) hue-rotate(900deg)
brightness(100%) contrast(100%);
}
.option :where(span, label){
  color: #5A6168;
  cursor: pointer;
  padding-left: 10px;
}
.option:is(:hover, .active) :where(span, label){
  color: #4A98F7;
}
.option #fill-color{
  cursor: pointer;
  height: 14px;
  width: 14px;
}
#fill-color:checked ~ label{
  color: #4A98F7;
}
.option #size-slider{
  width: 100%;
  height: 5px;
  margin-top: 10px;
}
.colors .options{
  display: flex;
  justify-content: space-between;
}
.colors .option{
  height: 20px;
  width: 20px;
  border-radius: 50%;
  margin-top: 3px;
  position: relative;
}
.colors .option:nth-child(1){
  background-color: #fff;
  border: 1px solid #bfbfbf;
}
.colors .option:nth-child(2){
  background-color: #000;
}
.colors .option:nth-child(3){
  background-color: #E02020;
}
.colors .option:nth-child(4){
  background-color: #6DD400;
```

```css
}
.colors .option:nth-child(5){
  background-color: #4A98F7;
}
.colors .option.selected::before{
  position: absolute;
  content: "";
  top: 50%;
  left: 50%;
  height: 12px;
  width: 12px;
  background: inherit;
  border-radius: inherit;
  border: 2px solid #fff;
  transform: translate(-50%, -50%);
}
.colors .option:first-child.selected::before{
  border-color: #ccc;
}
.option #color-picker{
  opacity: 0;
  cursor: pointer;
}
.buttons button{
  width: 100%;
  color: #fff;
  border: none;
  outline: none;
  padding: 11px 0;
  font-size: 0.9rem;
  margin-bottom: 13px;
  background: none;
  border-radius: 4px;
  cursor: pointer;
}
.buttons .clear-canvas{
  color: #6C757D;
  border: 1px solid #6C757D;
  transition: all 0.3s ease;
}
.clear-canvas:hover{
  color: #fff;
  background: #6C757D;
}
.buttons .save-img{
  background: #4A98F7;
  border: 1px solid #4A98F7;
}
```

```css
.drawing-board{
  flex: 1;
  overflow: hidden;
}
.drawing-board canvas{
  width: 100%;
  height: 100%;
}
```

script.js

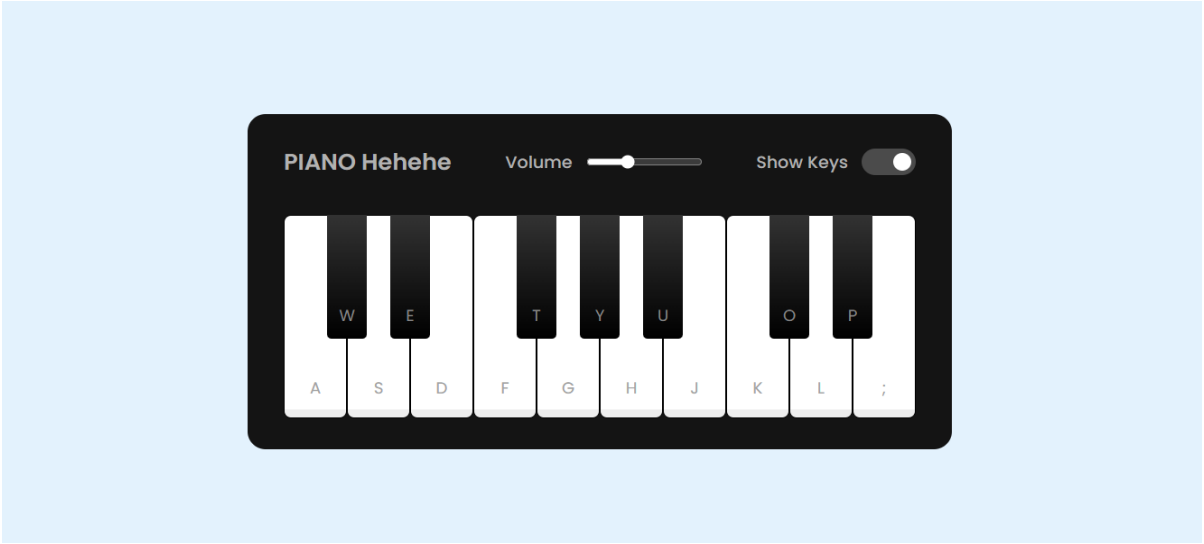
```js
const canvas = document.querySelector("canvas"),
toolBtns = document.querySelectorAll(".tool"),
fillColor = document.querySelector("#fill-color"),
sizeSlider = document.querySelector("#size-slider"),
colorBtns = document.querySelectorAll(".colors .option"),
colorPicker = document.querySelector("#color-picker"),
clearCanvas = document.querySelector(".clear-canvas"),
saveImg = document.querySelector(".save-img"),
ctx = canvas.getContext("2d");
// global variables with default value
let prevMouseX, prevMouseY, snapshot,
isDrawing = false,
selectedTool = "brush",
brushWidth = 5,
selectedColor = "#000";
const setCanvasBackground = () => {
    // setting whole canvas background to white, so the downloaded img
background will be white
    ctx.fillStyle = "#fff";
    ctx.fillRect(0, 0, canvas.width, canvas.height);
    ctx.fillStyle = selectedColor; // setting fillstyle back to the
selectedColor, it'll be the brush color
}
window.addEventListener("load", () => {
    // setting canvas width/height.. offsetwidth/height returns viewable
width/height of an element
    canvas.width = canvas.offsetWidth;
    canvas.height = canvas.offsetHeight;
    setCanvasBackground();
});
const drawRect = (e) => {
    // if fillColor isn't checked draw a rect with border else draw rect with
background
    if(!fillColor.checked) {
        // creating circle according to the mouse pointer
        return ctx.strokeRect(e.offsetX, e.offsetY, prevMouseX - e.offsetX,
prevMouseY - e.offsetY);
    }
```

```
    ctx.fillRect(e.offsetX, e.offsetY, prevMouseX - e.offsetX, prevMouseY -
e.offsetY);
}
const drawCircle = (e) => {
    ctx.beginPath(); // creating new path to draw circle
    // getting radius for circle according to the mouse pointer
    let radius = Math.sqrt(Math.pow((prevMouseX - e.offsetX), 2) +
Math.pow((prevMouseY - e.offsetY), 2));
    ctx.arc(prevMouseX, prevMouseY, radius, 0, 2 * Math.PI); // creating
circle according to the mouse pointer
    fillColor.checked ? ctx.fill() : ctx.stroke(); // if fillColor is checked
fill circle else draw border circle
}
const drawTriangle = (e) => {
    ctx.beginPath(); // creating new path to draw circle
    ctx.moveTo(prevMouseX, prevMouseY); // moving triangle to the mouse
pointer
    ctx.lineTo(e.offsetX, e.offsetY); // creating first line according to the
mouse pointer
    ctx.lineTo(prevMouseX * 2 - e.offsetX, e.offsetY); // creating bottom line
of triangle
    ctx.closePath(); // closing path of a triangle so the third line draw
automatically
    fillColor.checked ? ctx.fill() : ctx.stroke(); // if fillColor is checked
fill triangle else draw border
}
const startDraw = (e) => {
    isDrawing = true;
    prevMouseX = e.offsetX; // passing current mouseX position as prevMouseX
value
    prevMouseY = e.offsetY; // passing current mouseY position as prevMouseY
value
    ctx.beginPath(); // creating new path to draw
    ctx.lineWidth = brushWidth; // passing brushSize as line width
    ctx.strokeStyle = selectedColor; // passing selectedColor as stroke style
    ctx.fillStyle = selectedColor; // passing selectedColor as fill style
    // copying canvas data & passing as snapshot value.. this avoids dragging
the image
    snapshot = ctx.getImageData(0, 0, canvas.width, canvas.height);
}
const drawing = (e) => {
    if(!isDrawing) return; // if isDrawing is false return from here
    ctx.putImageData(snapshot, 0, 0); // adding copied canvas data on to this
canvas
    if(selectedTool === "brush" || selectedTool === "eraser") {
        // if selected tool is eraser then set strokeStyle to white
        // to paint white color on to the existing canvas content else set the
stroke color to selected color
```

```
        ctx.strokeStyle = selectedTool === "eraser" ? "#fff" : selectedColor;
        ctx.lineTo(e.offsetX, e.offsetY); // creating line according to the
mouse pointer
        ctx.stroke(); // drawing/filling line with color
    } else if(selectedTool === "rectangle"){
        drawRect(e);
    } else if(selectedTool === "circle"){
        drawCircle(e);
    } else {
        drawTriangle(e);
    }
}
toolBtns.forEach(btn => {
    btn.addEventListener("click", () => { // adding click event to all tool
option
        // removing active class from the previous option and adding on
current clicked option
        document.querySelector(".options .active").classList.remove("active");
        btn.classList.add("active");
        selectedTool = btn.id;
    });
});
sizeSlider.addEventListener("change", () => brushWidth = sizeSlider.value); //
passing slider value as brushSize
colorBtns.forEach(btn => {
    btn.addEventListener("click", () => { // adding click event to all color
button
        // removing selected class from the previous option and adding on
current clicked option
        document.querySelector(".options
.selected").classList.remove("selected");
        btn.classList.add("selected");
        // passing selected btn background color as selectedColor value
        selectedColor =
window.getComputedStyle(btn).getPropertyValue("background-color");
    });
});
colorPicker.addEventListener("change", () => {
    // passing picked color value from color picker to last color btn
background
    colorPicker.parentElement.style.background = colorPicker.value;
    colorPicker.parentElement.click();
});
clearCanvas.addEventListener("click", () => {
    ctx.clearRect(0, 0, canvas.width, canvas.height); // clearing whole canvas
    setCanvasBackground();
});
saveImg.addEventListener("click", () => {
```

```
    const link = document.createElement("a"); // creating <a> element
    link.download = `${Date.now()}.jpg`; // passing current date as link
download value
    link.href = canvas.toDataURL(); // passing canvasData as link href value
    link.click(); // clicking link to download image
});
canvas.addEventListener("mousedown", startDraw);
canvas.addEventListener("mousemove", drawing);
canvas.addEventListener("mouseup", () => isDrawing = false);
```

2. Piano hehe game



index.html

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>Playable Piano JavaScript | CodingNepal</title>
    <link rel="stylesheet" href="style.css">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <script src="script.js" defer></script>
  </head>
  <body>
    <div class="wrapper">
      <header>
        <h2>PIANO Hehehe</h2>
        <div class="column volume-slider">
          <span>Volume</span><input type="range" min="0" max="1" value="0.5"
step="any">
        </div>
        <div class="column keys-checkbox">
          <span>Show Keys</span><input type="checkbox" checked>
        </div>
```

```
      </header>
      <ul class="piano-keys">
        <li class="key white" data-key="a"><span>a</span></li>
        <li class="key black" data-key="w"><span>w</span></li>
        <li class="key white" data-key="s"><span>s</span></li>
        <li class="key black" data-key="e"><span>e</span></li>
        <li class="key white" data-key="d"><span>d</span></li>
        <li class="key white" data-key="f"><span>f</span></li>
        <li class="key black" data-key="t"><span>t</span></li>
        <li class="key white" data-key="g"><span>g</span></li>
        <li class="key black" data-key="y"><span>y</span></li>
        <li class="key white" data-key="h"><span>h</span></li>
        <li class="key black" data-key="u"><span>u</span></li>
        <li class="key white" data-key="j"><span>j</span></li>
        <li class="key white" data-key="k"><span>k</span></li>
        <li class="key black" data-key="o"><span>o</span></li>
        <li class="key white" data-key="l"><span>l</span></li>
        <li class="key black" data-key="p"><span>p</span></li>
        <li class="key white" data-key=";"><span>;</span></li>
      </ul>
    </div>
  </body>
</html>
```

style.css

```
import
url('https://fonts.googleapis.com/css2?family=Poppins:wght@400;500;600&display
=swap');
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  font-family: 'Poppins', sans-serif;
}
body {
  display: flex;
  align-items: center;
  justify-content: center;
  min-height: 100vh;
  background: #E3F2FD;
}
.wrapper {
  padding: 35px 40px;
  border-radius: 20px;
  background: #141414;
}
.wrapper header {
```

```
  display: flex;
  color: #B2B2B2;
  align-items: center;
  justify-content: space-between;
}
header h2 {
  font-size: 1.6rem;
}
header .column {
  display: flex;
  align-items: center;
}
header span {
  font-weight: 500;
  margin-right: 15px;
  font-size: 1.19rem;
}
header input {
  outline: none;
  border-radius: 30px;
}
.volume-slider input {
  accent-color: #fff;
}
.keys-checkbox input {
  height: 30px;
  width: 60px;
  cursor: pointer;
  appearance: none;
  position: relative;
  background: #4B4B4B
}
.keys-checkbox input::before {
  content: "";
  position: absolute;
  top: 50%;
  left: 5px;
  width: 20px;
  height: 20px;
  border-radius: 50%;
  background: #8c8c8c;
  transform: translateY(-50%);
  transition: all 0.3s ease;
}
.keys-checkbox input:checked::before {
  left: 35px;
  background: #fff;
}
```

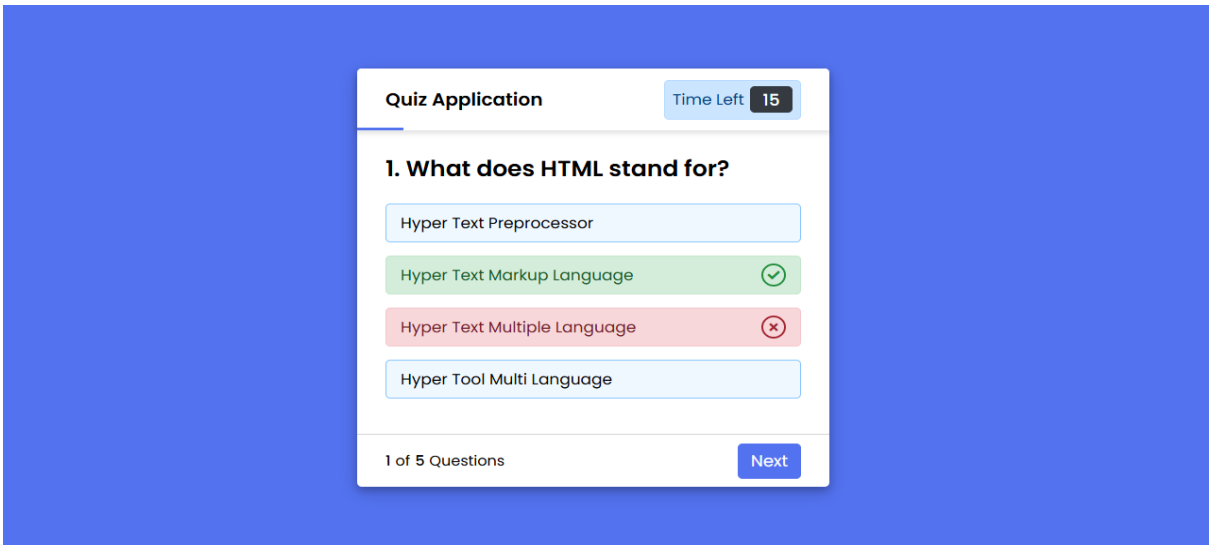
```css
.piano-keys {
  display: flex;
  list-style: none;
  margin-top: 40px;
}
.piano-keys .key {
  cursor: pointer;
  user-select: none;
  position: relative;
  text-transform: uppercase;
}
.piano-keys .black {
  z-index: 2;
  width: 44px;
  height: 140px;
  margin: 0 -22px 0 -22px;
  border-radius: 0 0 5px 5px;
  background: linear-gradient(#333, #000);
}
.piano-keys .black.active {
  box-shadow: inset -5px -10px 10px rgba(255,255,255,0.1);
  background:linear-gradient(to bottom, #000, #434343);
}
.piano-keys .white {
  height: 230px;
  width: 70px;
  border-radius: 8px;
  border: 1px solid #000;
  background: linear-gradient(#fff 96%, #eee 4%);
}
.piano-keys .white.active {
  box-shadow: inset -5px 5px 20px rgba(0,0,0,0.2);
  background:linear-gradient(to bottom, #fff 0%, #eee 100%);
}
.piano-keys .key span {
  position: absolute;
  bottom: 20px;
  width: 100%;
  color: #A2A2A2;
  font-size: 1.13rem;
  text-align: center;
}
.piano-keys .key.hide span {
  display: none;
}
.piano-keys .black span {
  bottom: 13px;
  color: #888888;
```

```
}
@media screen and (max-width: 815px) {
  .wrapper {
    padding: 25px;
  }
  header {
    flex-direction: column;
  }
  header :where(h2, .column) {
    margin-bottom: 13px;
  }
  .volume-slider input {
    max-width: 100px;
  }
  .piano-keys {
    margin-top: 20px;
  }
  .piano-keys .key:where(:nth-child(9), :nth-child(10)) {
    display: none;
  }
  .piano-keys .black {
    height: 100px;
    width: 40px;
    margin: 0 -20px 0 -20px;
  }
  .piano-keys .white {
    height: 180px;
    width: 60px;
  }
}
@media screen and (max-width: 615px) {
  .piano-keys .key:nth-child(13),
  .piano-keys .key:nth-child(14),
  .piano-keys .key:nth-child(15),
  .piano-keys .key:nth-child(16),
  .piano-keys .key :nth-child(17) {
    display: none;
  }
  .piano-keys .white {
    width: 50px;
  }
}
```

script.js

```
const pianoKeys = document.querySelectorAll(".piano-keys .key"),
volumeSlider = document.querySelector(".volume-slider input"),
keysCheckbox = document.querySelector(".keys-checkbox input");
let allKeys = [],
audio = new Audio(`tunes/a.wav`); // by default, audio src is "a" tune
const playTune = (key) => {
    audio.src = `tunes/${key}.wav`; // passing audio src based on key pressed
    audio.play(); // playing audio
    const clickedKey = document.querySelector(`[data-key="${key}"]`); //
getting clicked key element
    clickedKey.classList.add("active"); // adding active class to the clicked
key element
    setTimeout(() => { // removing active class after 150 ms from the clicked
key element
        clickedKey.classList.remove("active");
    }, 150);
}
pianoKeys.forEach(key => {
    allKeys.push(key.dataset.key); // adding data-key value to the allKeys
array
    // calling playTune function with passing data-key value as an argument
    key.addEventListener("click", () => playTune(key.dataset.key));
});
const handleVolume = (e) => {
    audio.volume = e.target.value; // passing the range slider value as an
audio volume
}
const showHideKeys = () => {
    // toggling hide class from each key on the checkbox click
    pianoKeys.forEach(key => key.classList.toggle("hide"));
}
const pressedKey = (e) => {
    // if the pressed key is in the allKeys array, only call the playTune
function
    if(allKeys.includes(e.key)) playTune(e.key);
}
keysCheckbox.addEventListener("click", showHideKeys);
volumeSlider.addEventListener("input", handleVolume);
document.addEventListener("keydown", pressedKey);
```

3. Quiz Game



index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Awesome Quiz App | CodingNepal</title>
  <!-- Linking CSS for styling the quiz app -->
  <link rel="stylesheet" href="style.css">
  <!-- FontAwesome CDN Link for Icons -->
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.3/css/all.min.css" />
</head>
<body>
  <!-- Start Quiz Button -->
  <div class="start_btn"><button>Start Quiz</button></div>

  <!-- Info Box (displayed when the quiz starts) -->
  <div class="info_box">
    <div class="info-title"><span>Some Rules of this Quiz</span></div>
    <div class="info-list">
      <div class="info">1. You will have only <span>15 seconds</span> per each
question.</div>
      <div class="info">2. Once you select your answer, it can't be
undone.</div>
      <div class="info">3. You can't select any option once time goes
off.</div>
      <div class="info">4. You can't exit from the Quiz while you're
playing.</div>
      <div class="info">5. You'll get points on the basis of your correct
answers.</div>
```

```
    </div>
    <div class="buttons">
      <button class="quit">Exit Quiz</button>
      <button class="restart">Continue</button>
    </div>
  </div>

  <!-- Quiz Box (main container for the quiz) -->
  <div class="quiz_box">
    <header>
      <div class="title">Quiz Application</div>
      <div class="timer">
        <div class="time_left_txt">Time Left</div>
        <div class="timer_sec">15</div>
      </div>
      <div class="time_line"></div>
    </header>
    <section>
      <div class="que_text">
        <!-- Question text will be inserted here by JavaScript -->
      </div>
      <div class="option_list">
        <!-- Options will be inserted here by JavaScript -->
      </div>
    </section>

    <!-- Footer of Quiz Box -->
    <footer>
      <div class="total_que">
        <!-- Question count number will be inserted here by JavaScript -->
      </div>
      <button class="next_btn">Next</button>
    </footer>
  </div>

  <!-- Result Box (displayed after completing the quiz) -->
  <div class="result_box">
    <div class="icon">
      <i class="fas fa-crown"></i>
    </div>
    <div class="complete_text">You've completed the Quiz!</div>
    <div class="score_text">
      <!-- Score result will be inserted here by JavaScript -->
    </div>
    <div class="buttons">
      <button class="restart">Replay Quiz</button>
      <button class="quit">Quit Quiz</button>
    </div>
```

```
  </div>

  <!-- JavaScript file for managing questions and options -->
  <script src="js/questions.js"></script>

  <!-- JavaScript file containing all quiz logic -->
  <script src="js/script.js"></script>
</body>
</html>
```

style.css

```
@import
url('https://fonts.googleapis.com/css2?family=Poppins:wght@200;300;400;500;600
;700&display=swap');

* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  font-family: 'Poppins', sans-serif;
}

body {
  overflow-x: hidden;
  background: #5372F0;
}

.start_btn,
.info_box,
.quiz_box,
.result_box {
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2),
    0 6px 20px 0 rgba(0, 0, 0, 0.19);
}

.info_box.activeInfo,
.quiz_box.activeQuiz,
.result_box.activeResult {
  opacity: 1;
  z-index: 5;
  pointer-events: auto;
  transform: translate(-50%, -50%) scale(1);
}
```

```css
.start_btn button {
  font-size: 25px;
  font-weight: 500;
  color: #5372F0;
  padding: 15px 30px;
  outline: none;
  border: none;
  border-radius: 5px;
  background: #fff;
  cursor: pointer;
}

.info_box {
  max-width: 500px;
  width: 95%;
  background: #fff;
  border-radius: 5px;
  transform: translate(-50%, -50%) scale(0.9);
  opacity: 0;
  pointer-events: none;
  transition: all 0.3s ease;
}

.info_box .info-title {
  height: 60px;
  width: 100%;
  border-bottom: 1px solid lightgrey;
  display: flex;
  align-items: center;
  padding: 0 30px;
  border-radius: 5px 5px 0 0;
  font-size: 20px;
  font-weight: 600;
}

.info_box .info-list {
  padding: 15px 30px;
}

.info_box .info-list .info {
  margin: 5px 0;
  font-size: 17px;
}

.info_box .info-list .info span {
  font-weight: 600;
  color: #5372F0;
```

```css
}

.info_box .buttons {
  height: 60px;
  display: flex;
  align-items: center;
  justify-content: flex-end;
  padding: 0 30px;
  border-top: 1px solid lightgrey;
}

.info_box .buttons button {
  margin: 0 5px;
  height: 40px;
  width: 100px;
  font-size: 16px;
  font-weight: 500;
  cursor: pointer;
  border: none;
  outline: none;
  border-radius: 5px;
  border: 1px solid #5372F0;
  transition: all 0.3s ease;
}

.quiz_box {
  max-width: 500px;
  width: 95%;
  background: #fff;
  border-radius: 5px;
  transform: translate(-50%, -50%) scale(0.9);
  opacity: 0;
  pointer-events: none;
  transition: all 0.3s ease;
}

.quiz_box header {
  position: relative;
  z-index: 2;
  height: 70px;
  padding: 0 30px;
  background: #fff;
  border-radius: 5px 5px 0 0;
  display: flex;
  align-items: center;
  justify-content: space-between;
  box-shadow: 0px 3px 5px 1px rgba(0, 0, 0, 0.1);
}
```

```css
.quiz_box header .title {
  font-size: 20px;
  font-weight: 600;
}

.quiz_box header .timer {
  color: #004085;
  background: #cce5ff;
  border: 1px solid #b8daff;
  height: 45px;
  padding: 0 8px;
  border-radius: 5px;
  display: flex;
  align-items: center;
  justify-content: space-between;
  width: 145px;
}

.quiz_box header .timer .time_left_txt {
  font-weight: 400;
  font-size: 17px;
  user-select: none;
}

.quiz_box header .timer .timer_sec {
  font-size: 18px;
  font-weight: 500;
  height: 30px;
  width: 45px;
  color: #fff;
  border-radius: 5px;
  line-height: 30px;
  text-align: center;
  background: #343a40;
  border: 1px solid #343a40;
  user-select: none;
}

.quiz_box header .time_line {
  position: absolute;
  bottom: 0px;
  left: 0px;
  height: 3px;
  background: #5372F0;
}

section {
```

```
  padding: 25px 30px 20px 30px;
  background: #fff;
}

section .que_text {
  font-size: 25px;
  font-weight: 600;
}

section .option_list {
  padding: 20px 0px;
  display: block;
}

section .option_list .option {
  background: aliceblue;
  border: 1px solid #84c5fe;
  border-radius: 5px;
  padding: 8px 15px;
  font-size: 17px;
  margin-bottom: 15px;
  cursor: pointer;
  transition: all 0.3s ease;
  display: flex;
  align-items: center;
  justify-content: space-between;
}

section .option_list .option:last-child {
  margin-bottom: 0px;
}

section .option_list .option:hover {
  color: #004085;
  background: #cce5ff;
  border: 1px solid #b8daff;
}

section .option_list .option.correct {
  color: #155724;
  background: #d4edda;
  border: 1px solid #c3e6cb;
}

section .option_list .option.incorrect {
  color: #721c24;
  background: #f8d7da;
  border: 1px solid #f5c6cb;
```

```css
}

section .option_list .option.disabled {
  pointer-events: none;
}

section .option_list .option .icon {
  height: 26px;
  width: 26px;
  border: 2px solid transparent;
  border-radius: 50%;
  text-align: center;
  font-size: 13px;
  pointer-events: none;
  transition: all 0.3s ease;
  line-height: 24px;
}

.option_list .option .icon.tick {
  color: #23903c;
  border-color: #23903c;
  background: #d4edda;
}

.option_list .option .icon.cross {
  color: #a42834;
  background: #f8d7da;
  border-color: #a42834;
}

footer {
  height: 60px;
  padding: 0 30px;
  display: flex;
  align-items: center;
  justify-content: space-between;
  border-top: 1px solid lightgrey;
}

footer .total_que span {
  display: flex;
  user-select: none;
}

footer .total_que span p {
  font-weight: 500;
  padding: 0 5px;
}
```

```css
footer .total_que span p:first-child {
  padding-left: 0px;
}

footer button {
  height: 40px;
  padding: 0 13px;
  font-size: 18px;
  font-weight: 400;
  cursor: pointer;
  border: none;
  outline: none;
  color: #fff;
  border-radius: 5px;
  background: #5372F0;
  border: 1px solid #5372F0;
  line-height: 10px;
  opacity: 0;
  pointer-events: none;
  transform: scale(0.95);
  transition: all 0.3s ease;
}

footer button:hover {
  background: #0263ca;
}

footer button.show {
  opacity: 1;
  pointer-events: auto;
  transform: scale(1);
}

.result_box {
  background: #fff;
  border-radius: 5px;
  display: flex;
  padding: 25px 30px;
  max-width: 400px;
  width: 95%;
  align-items: center;
  flex-direction: column;
  justify-content: center;
  transform: translate(-50%, -50%) scale(0.9);
  opacity: 0;
  pointer-events: none;
  transition: all 0.3s ease;
```

```css
}

.result_box .icon {
  font-size: 100px;
  color: #5372F0;
  margin-bottom: 10px;
}

.result_box .complete_text {
  font-size: 20px;
  font-weight: 500;
}

.result_box .score_text span {
  display: flex;
  margin: 10px 0;
  font-size: 18px;
  font-weight: 500;
}

.result_box .score_text span p {
  padding: 0 4px;
  font-weight: 600;
}

.result_box .buttons {
  display: flex;
  margin: 20px 0;
}

.result_box .buttons button {
  margin: 0 10px;
  height: 45px;
  padding: 0 20px;
  font-size: 18px;
  font-weight: 500;
  cursor: pointer;
  border: none;
  outline: none;
  border-radius: 5px;
  border: 1px solid #5372F0;
  transition: all 0.3s ease;
}

.buttons button.restart {
  color: #fff;
  background: #5372F0;
}
```

```css
.buttons button.restart:hover {
  background: #0263ca;
}

.buttons button.quit {
  color: #5372F0;
  background: #fff;
}

.buttons button.quit:hover {
  color: #fff;
  background: #5372F0;
}

/* Responsive media query code for small devices */
@media (max-width: 768px) {
  section {
    padding: 25px 15px 20px 15px;
  }

  .quiz_box header,
  .info_box .info-title,
  .info_box .buttons {
    padding: 0 15px;
  }

  .result_box {
    padding: 25px 10px;
  }

  .info_box .info-list {
    padding: 15px;
  }

  .start_btn button {
    font-size: 20px;
    padding: 10px 25px;
  }
}
```

script.js

```
// Selecting all required elements
const startBtn = document.querySelector(".start_btn button");
const infoBox = document.querySelector(".info_box");
const exitBtn = infoBox.querySelector(".buttons .quit");
const continueBtn = infoBox.querySelector(".buttons .restart");
const quizBox = document.querySelector(".quiz_box");
const resultBox = document.querySelector(".result_box");
const optionList = document.querySelector(".option_list");
const timeLine = document.querySelector("header .time_line");
const timeText = document.querySelector(".timer .time_left_txt");
const timeCount = document.querySelector(".timer .timer_sec");

let timeValue = 15;
let queCount = 0;
let queNumb = 1;
let userScore = 0;
let counter;
let counterLine;
let widthValue = 0;

const restartQuizBtn = resultBox.querySelector(".buttons .restart");
const quitQuizBtn = resultBox.querySelector(".buttons .quit");
const nextBtn = document.querySelector("footer .next_btn");
const bottomQuesCounter = document.querySelector("footer .total_que");

// Show info box when start button is clicked
startBtn.onclick = () => {
  infoBox.classList.add("activeInfo");
}

// Hide info box when exit button is clicked
exitBtn.onclick = () => {
  infoBox.classList.remove("activeInfo");
}

// Start quiz when continue button is clicked
continueBtn.onclick = () => {
  infoBox.classList.remove("activeInfo");
  quizBox.classList.add("activeQuiz");
  initializeQuiz();
}

// Restart quiz when restart button is clicked
restartQuizBtn.onclick = () => {
  resultBox.classList.remove("activeResult");
  quizBox.classList.add("activeQuiz");
  resetQuiz();
```

```
  initializeQuiz();
}

// Reload page when quit button is clicked
quitQuizBtn.onclick = () => {
  window.location.reload();
}

// Show next question when next button is clicked
nextBtn.onclick = () => {
  if (queCount < questions.length - 1) {
    queCount++;
    queNumb++;
    updateQuiz();
  } else {
    clearInterval(counter);
    clearInterval(counterLine);
    showResult();
  }
}

// Initialize the quiz with the first question and timers
function initializeQuiz() {
  showQuestions(queCount);
  queCounter(queNumb);
  startTimer(timeValue);
  startTimerLine(widthValue);
}

// Reset quiz variables
function resetQuiz() {
  timeValue = 15;
  queCount = 0;
  queNumb = 1;
  userScore = 0;
  widthValue = 0;
}

// Update the quiz with the next question and reset timers
function updateQuiz() {
  showQuestions(queCount);
  queCounter(queNumb);
  clearInterval(counter);
  clearInterval(counterLine);
  startTimer(timeValue);
  startTimerLine(widthValue);
  timeText.textContent = "Time Left";
  nextBtn.classList.remove("show");
```

```
}

// Show questions and options
function showQuestions(index) {
  const queText = document.querySelector(".que_text");
  let queTag = `<span>${questions[index].numb}.
${questions[index].question}</span>`;
  let optionTag = questions[index].options.map(option => `<div
class="option"><span>${option}</span></div>`).join('');
  queText.innerHTML = queTag;
  optionList.innerHTML = optionTag;

  optionList.querySelectorAll(".option").forEach(option => {
    option.onclick = () => optionSelected(option);
  });
}

// Handle option selection
function optionSelected(answer) {
  clearInterval(counter);
  clearInterval(counterLine);
  let userAns = answer.textContent;
  let correctAns = questions[queCount].answer;
  let allOptions = optionList.children.length;

  if (userAns === correctAns) {
    userScore++;
    answer.classList.add("correct");
    answer.insertAdjacentHTML("beforeend", tickIconTag);
  } else {
    answer.classList.add("incorrect");
    answer.insertAdjacentHTML("beforeend", crossIconTag);
    highlightCorrectAnswer(correctAns);
  }

  disableOptions();
  nextBtn.classList.add("show");
}

// Highlight the correct answer
function highlightCorrectAnswer(correctAns) {
  for (let i = 0; i < optionList.children.length; i++) {
    if (optionList.children[i].textContent === correctAns) {
      optionList.children[i].classList.add("correct");
      optionList.children[i].insertAdjacentHTML("beforeend", tickIconTag);
    }
  }
}
```

```
// Disable all options
function disableOptions() {
  for (let i = 0; i < optionList.children.length; i++) {
    optionList.children[i].classList.add("disabled");
  }
}

// Show result box
function showResult() {
  infoBox.classList.remove("activeInfo");
  quizBox.classList.remove("activeQuiz");
  resultBox.classList.add("activeResult");
  const scoreText = resultBox.querySelector(".score_text");
  let scoreTag = '';

  if (userScore > 3) {
    scoreTag = `<span>and congrats! 🎉, You got <p>${userScore}</p> out of
<p>${questions.length}</p></span>`;
  } else if (userScore > 1) {
    scoreTag = `<span>and nice 😎, You got <p>${userScore}</p> out of
<p>${questions.length}</p></span>`;
  } else {
    scoreTag = `<span>and sorry 😐, You got only <p>${userScore}</p> out of
<p>${questions.length}</p></span>`;
  }

  scoreText.innerHTML = scoreTag;
}

// Start the timer for the quiz
function startTimer(time) {
  counter = setInterval(() => {
    timeCount.textContent = time > 9 ? time : `0${time}`;
    time--;
    if (time < 0) {
      clearInterval(counter);
      timeText.textContent = "Time Off";
      highlightCorrectAnswer(questions[queCount].answer);
      disableOptions();
      nextBtn.classList.add("show");
    }
  }, 1000);
}

function startTimerLine(time) {
  const totalTime = 550; // Total time for the timer in milliseconds
  counterLine = setInterval(() => {
```

```
    time += 1;
    let progressPercentage = (time / totalTime) * 100;
    timeLine.style.width = `${progressPercentage}%`;
    if (time >= totalTime) {
      clearInterval(counterLine);
    }
  }, 29);
}

// Update the question counter
function queCounter(index) {
  let totalQueCounTag = `<span><p>${index}</p> of <p>${questions.length}</p>
Questions</span>`;
  bottomQuesCounter.innerHTML = totalQueCounTag;
}

// Tick and cross icons
const tickIconTag = '<div class="icon tick"><i class="fas fa-
check"></i></div>';
const crossIconTag = '<div class="icon cross"><i class="fas fa-
times"></i></div>';
```

questions.js

```
// Questions array
const questions = [
  {
    numb: 1,
    question: "What does HTML stand for?",
    answer: "Hyper Text Markup Language",
    options: [
      "Hyper Text Preprocessor",
      "Hyper Text Markup Language",
      "Hyper Text Multiple Language",
      "Hyper Tool Multi Language"
    ]
  },
  {
    numb: 2,
    question: "What does CSS stand for?",
    answer: "Cascading Style Sheet",
    options: [
      "Common Style Sheet",
      "Colorful Style Sheet",
      "Computer Style Sheet",
      "Cascading Style Sheet"
    ]
  },
```

```
  {
    numb: 3,
    question: "What does PHP stand for?",
    answer: "Hypertext Preprocessor",
    options: [
      "Hypertext Preprocessor",
      "Hypertext Programming",
      "Hypertext Preprogramming",
      "Hometext Preprocessor"
    ]
  },
  {
    numb: 4,
    question: "What does SQL stand for?",
    answer: "Structured Query Language",
    options: [
      "Stylish Question Language",
      "Stylesheet Query Language",
      "Statement Question Language",
      "Structured Query Language"
    ]
  },
  {
    numb: 5,
    question: "What does XML stand for?",
    answer: "eXtensible Markup Language",
    options: [
      "eXtensible Markup Language",
      "eXecutable Multiple Language",
      "eXTra Multi-Program Language",
      "eXamine Multiple Language"
    ]
  },
];
```