**📄 app\app.component.html (.html)**

<ion-app>

<ion-router-outlet></ion-router-outlet>

</ion-app>

**📄 app\app.component.scss (.scss)**

**📄 app\app.component.spec.ts (.ts)**

import { TestBed } from '@angular/core/testing';

import { provideRouter } from '@angular/router';

import { AppComponent } from './app.component';

describe('AppComponent', () => {

it('should create the app', async () => {

await TestBed.configureTestingModule({

imports: [AppComponent],

providers: [provideRouter([])]

}).compileComponents();

const fixture = TestBed.createComponent(AppComponent);

const app = fixture.componentInstance;

expect(app).toBeTruthy();

});

});

**📄 app\app.component.ts (.ts)**

import { Component } from '@angular/core';

import { IonApp, IonRouterOutlet } from '@ionic/angular/standalone';

@Component({

selector: 'app-root',

templateUrl: 'app.component.html',

imports: [IonApp, IonRouterOutlet],

})

export class AppComponent {

constructor() {}

ngOnInit(): void {

}

}

**📄 app\app.routes.ts (.ts)**

import { Routes } from '@angular/router';

import { HomePage } from './home/home.page';

export const routes: Routes = [

{

path: '',

pathMatch: 'full',

redirectTo: 'home'

},

{

path: 'home',

component: HomePage

}

];

**📄 app\home\home.page.html (.html)**

<canvas #gameCanvas id="gameCanvas"></canvas>

<div id="quizContainer" style="display: none">

<div class="quiz-question" id="questionText"></div>

<div class="choices" id="choicesContainer"></div>

</div>

<button id="startBtn" (click)="startGame()" class="start-btn">Start Game</button>

<button id="pauseBtn" (click)="pauseGame()" class="pause-btn" style="display: none">Pause</button>

<div class="backdrop">

<div class="gradient-top"></div>

<div class="gradient-bottom"></div>

</div>

<div class="score-panel">

<h3 class="score" id="score">Score: 0</h3>

<div class="high-score" id="highScore">High Score: 0</div>

</div>

**📄 app\home\home.page.scss (.scss)**

::ng-deep {

canvas {

display: block;

width: 100vw;

height: 100vh;

touch-action: none;

}

.quiz-question {

position: absolute;

top: 50px;

left: 50%;

transform: translateX(-50%);

font-size: 1rem;

font-weight: bold;

color: #fff;

background: rgba(0, 0, 0, 0.6);

padding: 10px 20px;

border-radius: 10px;

z-index: 10;

}

.choices {

position: absolute;

bottom: 50px;

left: 50%;

transform: translateX(-50%);

display: flex;

gap: 20px;

z-index: 10;

.choice {

background: #222;

color: white;

font-size: 1rem;

padding: 10px 20px;

border: 2px solid #fff;

border-radius: 8px;

cursor: pointer;

user-select: none;

&:hover {

background: #444;

}

}

}

.start-btn {

position: absolute;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

font-size: 1rem;

padding: 15px 30px;

background: #222;

color: white;

border: 2px solid #fff;

border-radius: 10px;

cursor: pointer;

z-index: 20;

}

.pause-btn {

position: absolute;

top: 30px;

// right: 10px;

right: calc(env(safe-area-inset-right, 20px) + 10px);

font-size: 1rem;

padding: 8px 12px;

background: #222;

color: white;

border: 2px solid #fff;

border-radius: 5px;

cursor: pointer;

z-index: 20;

}

.backdrop {

position: fixed;

top: 0;

left: 0;

width: 100%;

height: 100%;

pointer-events: none;

z-index: 5;

}

.gradient-top,

.gradient-bottom {

position: fixed;

left: 0;

width: 100%;

pointer-events: none;

z-index: 10;

}

.gradient-top {

top: 0;

height: 60%;

/\* adjust as needed \*/

background: linear-gradient(to bottom, rgba(0, 0, 0, 0.32), rgba(0, 0, 0, 0));

}

.gradient-bottom {

bottom: 0;

height: 60%;

/\* adjust as needed \*/

background: linear-gradient(to top, rgba(0, 0, 0, 0.32), rgba(0, 0, 0, 0));

}

.score-panel {

position: absolute;

top: 30px;

left: 10px;

font-size: 1.5rem;

color: white;

z-index: 20;

}

.score {

margin: 0;

}

.high-score {

font-size: 1rem;

font-weight: bold;

}

}

**📄 app\home\home.page.spec.ts (.ts)**

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { HomePage } from './home.page';

describe('HomePage', () => {

let component: HomePage;

let fixture: ComponentFixture<HomePage>;

beforeEach(async () => {

fixture = TestBed.createComponent(HomePage);

component = fixture.componentInstance;

fixture.detectChanges();

});

it('should create', () => {

expect(component).toBeTruthy();

});

});

**📄 app\home\home.page.ts (.ts)**

import { HttpClient, HttpClientModule } from '@angular/common/http';

import { AfterViewInit, Component, ElementRef, OnInit, ViewChild } from '@angular/core';

import { IonHeader, IonToolbar, IonTitle, IonContent } from '@ionic/angular/standalone';

import { ScreenOrientation } from '@capacitor/screen-orientation';

import { Capacitor } from '@capacitor/core';

import { Device } from '@capacitor/device';

import { App } from '@capacitor/app';

import { AlertController } from '@ionic/angular';

import { Howl } from 'howler';

@Component({

selector: 'app-home',

templateUrl: 'home.page.html',

styleUrls: ['home.page.scss'],

imports: [IonHeader, IonToolbar, IonTitle, IonContent, HttpClientModule],

})

export class HomePage implements OnInit, AfterViewInit {

@ViewChild('gameCanvas', { static: false }) canvasRef!: ElementRef;

questions: any[] = [];

canvas!: HTMLCanvasElement;

ctx!: CanvasRenderingContext2D;

highScore = 0;

score = 0;

consecutiveCorrect = 0;

questionStartTime = 0;

nextQuestionAt: number | null = null;

questionTimeout = 5000;

SPEED = 0.3;

GRAVITY = 0.5;

FRICTION = 0.98;

TERRAIN\_AMPLITUDE = 80;

CONTACT\_THRESHOLD = 0;

paused = true;

gameOver = false;

quizReady = false;

cliff: any = null;

bridgeShownFor: any = null;

vehicle: any = {

x: 0,

y: 0,

vx: this.SPEED,

vy: 0,

angle: 0,

width: 50,

height: 25,

wheelBase: 25,

wheelRadius: 10

};

soundBg = new Howl({

src: ['assets/audio/background.mp3'],

loop: true,

volume: 0.2

});

soundCorrectAnswer = new Howl({

src: ['assets/audio/correct-answer.mp3'],

loop: false,

volume: 0.4,

autoplay: false

});

soundWrongAnswer = new Howl({

src: ['assets/audio/wrong-answer.mp3'],

loop: false,

volume: 0.4,

autoplay: false

});

soundFalling = new Howl({

src: ['assets/audio/falling.mp3'],

loop: false,

volume: 0.4,

autoplay: false

});

soundGameOver = new Howl({

src: ['assets/audio/game-over.mp3'],

loop: false,

volume: 0.4,

autoplay: false

});

soundEngineStart = new Howl({

src: ['assets/audio/engine-start.mp3'],

loop: false,

volume: 0.4,

autoplay: false

});

soundGo = new Howl({

src: ['assets/audio/go.mp3'],

loop: false,

volume: 0.4,

autoplay: false

});

soundStartPause = new Howl({

src: ['assets/audio/start-pause.mp3'],

loop: false,

volume: 1,

autoplay: false

});

soundNextQuestion = new Howl({

src: ['assets/audio/next-question.mp3'],

loop: false,

volume: 1,

autoplay: false

});

constructor(private http: HttpClient) {

}

ngOnInit(): void {

App.addListener('pause', () => {

if (this.soundBg.playing()) {

this.soundBg.stop();

}

this.pauseGame();

});

App.addListener('resume', () => {

if (!this.soundBg.playing()) {

this.soundBg.play();

}

});

if (!this.soundBg.playing()) {

this.soundBg.play();

}

document.addEventListener("backbutton", async (e) => {

e.preventDefault(); // Prevent default back action

if (!this.paused && !this.gameOver) {

this.pauseGame();

} else {

const confirmExit = confirm("Are you sure you want to exit the game?");

if (confirmExit) {

this.soundBg.stop();

await App.exitApp();

}

}

}, false);

}

async ngAfterViewInit() {

window.addEventListener("resize", () => { this.resizeCanvas() });

window.addEventListener("orientationchange", () => { this.resizeCanvas() });

await this.lockLandscapeIfMobile();

await this.loadQuestions();

// Wait for canvas to be present in DOM and visible

await this.waitForCanvasReady();

this.canvas = this.canvasRef.nativeElement;

this.resizeCanvas(); // initial

this.ctx = this.canvas.getContext('2d')!;

this.canvas.width = window.innerWidth;

this.canvas.height = window.innerHeight;

this.initVehicleY();

this.setHighScore(

localStorage.getItem('highScore') ? +localStorage.getItem('highScore')! : 0

);

this.setScore(0);

this.loop();

const navHeight = this.getNavigationBarHeight();

if (navHeight > 0) {

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

pauseBtn!.style.marginRight = `${navHeight + 10}px`;

}

}

getNavigationBarHeight(): number {

const screenHeight = window.screen.height;

const innerHeight = window.innerHeight;

const navBarHeight = screenHeight - innerHeight;

return navBarHeight > 0 ? navBarHeight : 0;

}

resizeCanvas() {

console.log("this.canvas", this.canvas);

console.log("window.innerWidth", window.innerWidth);

console.log("window.innerHeight", window.innerHeight);

this.canvas.width = window.innerWidth;

this.canvas.height = window.innerHeight;

}

private waitForCanvasReady(): Promise<void> {

return new Promise((resolve) => {

const check = () => {

const canvas = this.canvasRef?.nativeElement;

if (canvas && canvas.offsetWidth > 0 && canvas.offsetHeight > 0) {

resolve();

} else {

requestAnimationFrame(check);

}

};

check();

});

}

async lockLandscapeIfMobile() {

const info = await Device.getInfo();

const isMobile = info.platform === 'ios' || info.platform === 'android';

const isNative = Capacitor.isNativePlatform();

if (isMobile && isNative) {

try {

await ScreenOrientation.lock({ orientation: 'landscape' });

console.log('🔒 Screen locked to landscape');

} catch (err) {

console.warn('⚠️ Failed to lock screen orientation:', err);

}

} else {

console.log('💻 Skipping orientation lock (not native mobile)');

}

}

async loadQuestions() {

const data: any = await this.http.get<any[]>('assets/questions.json').toPromise();

this.questions = data.map((q: any) => ({ ...q, done: false }));

}

initVehicleY() {

const frontX = this.vehicle.x + this.vehicle.wheelBase / 2;

const backX = this.vehicle.x - this.vehicle.wheelBase / 2;

const frontY = this.getTerrainY(frontX);

const backY = this.getTerrainY(backX);

const avgY = (frontY + backY) / 2;

this.vehicle.y = avgY - this.vehicle.wheelRadius - this.vehicle.height / 2;

}

startGame() {

this.soundStartPause.play();

this.paused = false;

this.gameOver = false;

document.getElementById('startBtn')!.style.display = 'none';

document.getElementById('pauseBtn')!.style.display = 'block';

if (!this.cliff) {

this.nextQuestionAt = Date.now() + 5000;

} else if (this.quizReady) {

document.getElementById('quizContainer')!.style.display = 'block';

}

if (document.getElementById('startBtn')?.innerText === "Start Game" || document.getElementById('startBtn')?.innerText === "Play Again") {

setTimeout(() => {

this.soundGo.play();

this.soundEngineStart.play();

}, 500);

}

}

pauseGame() {

this.soundStartPause.play();

this.paused = true;

document.getElementById('pauseBtn')!.style.display = 'none';

const startBtn = document.getElementById('startBtn')!;

startBtn.style.display = 'block';

startBtn.innerText = 'Resume Game';

document.getElementById('quizContainer')!.style.display = 'none';

}

getRandomQuestion() {

const available = this.questions.filter(q => !q.done);

if (available.length === 0) {

this.gameOver = true;

document.getElementById('quizContainer')!.style.display = 'none';

document.getElementById('pauseBtn')!.style.display = 'none';

setTimeout(() => this.resetGame(), 5000);

return null;

}

const idx = Math.floor(Math.random() \* available.length);

const question = available[idx];

this.questions = this.questions.map(q => q.id === question.id ? { ...q, done: true } : q);

return question;

}

showQuestion() {

if (this.paused || this.gameOver) return;

const question = this.getRandomQuestion();

if (!question) return;

this.soundNextQuestion.play();

const quizContainer = document.getElementById('quizContainer')!;

const questionText = document.getElementById('questionText')!;

const choicesContainer = document.getElementById('choicesContainer')!;

quizContainer.style.display = 'block';

questionText.textContent = question.question;

choicesContainer.innerHTML = '';

this.quizReady = true;

this.questionStartTime = Date.now();

question.choices.forEach((choice: string, idx: number) => {

const btn = document.createElement('div');

btn.className = 'choice';

btn.textContent = choice;

btn.onclick = () => this.handleAnswer(idx, question);

choicesContainer.appendChild(btn);

});

}

handleAnswer(idx: number, question: any) {

if (!this.quizReady) return;

this.quizReady = false;

const timeTaken = Date.now() - this.questionStartTime;

if (idx === question.correct) {

document.getElementById('quizContainer')!.style.display = 'none';

this.cliff.bridgeShown = true;

this.SPEED += 0.1;

this.vehicle.height \*= 0.99;

this.vehicle.width \*= 0.99;

this.vehicle.wheelBase \*= 0.99;

this.vehicle.wheelRadius \*= 0.99;

this.questionTimeout \*= 0.995;

let points = 100;

if (this.SPEED > 0.5) points += 10;

if (timeTaken < 2000) points += 50;

this.consecutiveCorrect++;

if (this.consecutiveCorrect % 3 === 0) points += 50;

this.updateScore(points);

this.nextQuestionAt = Date.now() + this.questionTimeout;

this.soundCorrectAnswer.play();

} else {

document.getElementById('quizContainer')!.style.display = 'none';

document.getElementById('pauseBtn')!.style.display = 'none';

this.SPEED = 5;

this.updateScore(-50);

this.consecutiveCorrect = 0;

this.soundWrongAnswer.play();

}

}

getTerrainY(x: number): number {

if (this.cliff && x > this.cliff.x && x < this.cliff.x + this.cliff.width && !this.cliff.bridgeShown) {

return this.canvas.height + 1000;

}

return this.canvas.height - 120 - Math.sin(x \* 0.005) \* this.TERRAIN\_AMPLITUDE - Math.cos(x \* 0.02) \* 30;

}

spawnCliff() {

this.cliff = {

x: this.vehicle.x + this.canvas.width,

width: 400,

active: true,

bridgeShown: false,

scheduledNext: false

};

this.bridgeShownFor = null;

}

updateVehicle() {

this.vehicle.vy += this.GRAVITY;

const frontX = this.vehicle.x + this.vehicle.wheelBase / 2;

const backX = this.vehicle.x - this.vehicle.wheelBase / 2;

const frontY = this.getTerrainY(frontX);

const backY = this.getTerrainY(backX);

const dx = frontX - backX;

const dy = frontY - backY;

const slopeAngle = Math.atan2(dy, dx);

this.vehicle.vx += this.SPEED \* Math.cos(slopeAngle) \* 0.1;

this.vehicle.vy -= this.SPEED \* Math.sin(slopeAngle) \* 0.1;

this.vehicle.vx \*= this.FRICTION;

this.vehicle.vy \*= this.FRICTION;

this.vehicle.x += this.vehicle.vx;

this.vehicle.y += this.vehicle.vy;

const expectedY = (frontY + backY) / 2 - this.vehicle.wheelRadius - this.vehicle.height / 2;

const belowGround = this.vehicle.y > expectedY + this.CONTACT\_THRESHOLD;

const aboveGround = this.vehicle.y < expectedY - this.CONTACT\_THRESHOLD;

if (!aboveGround && belowGround) {

this.vehicle.y = expectedY;

this.vehicle.vy = 0;

this.vehicle.angle = slopeAngle;

} else if (!aboveGround) {

const springStrength = 0.2;

const damping = 0.9;

const displacement = expectedY - this.vehicle.y;

this.vehicle.vy += displacement \* springStrength;

this.vehicle.vy \*= damping;

this.vehicle.angle = slopeAngle;

}

}

drawTerrain(camX: number) {

this.ctx.beginPath();

this.ctx.moveTo(0, this.canvas.height);

for (let x = 0; x <= this.canvas.width; x += 10) {

const worldX = camX + x;

const y = this.getTerrainY(worldX);

this.ctx.lineTo(x, y);

}

this.ctx.lineTo(this.canvas.width, this.canvas.height);

this.ctx.fillStyle = '#654321';

this.ctx.fill();

}

drawVehicle(camX: number) {

const { x, y, angle, width, height, wheelRadius, wheelBase } = this.vehicle;

this.ctx.save();

this.ctx.translate(x - camX, y);

this.ctx.rotate(angle);

this.ctx.fillStyle = 'red';

this.ctx.fillRect(-width / 2, -height / 2, width, height);

const drawWheel = (offsetX: number) => {

this.ctx.beginPath();

this.ctx.arc(offsetX, height / 2, wheelRadius, 0, Math.PI \* 2);

this.ctx.fillStyle = 'black';

this.ctx.fill();

};

drawWheel(-wheelBase / 2);

drawWheel(wheelBase / 2);

this.ctx.restore();

}

resetGame() {

this.questions = this.questions.map(q => ({ ...q, done: false }));

this.SPEED = 0.3;

Object.assign(this.vehicle, {

x: 0, y: 0, vx: this.SPEED, vy: 0,

width: 50, height: 25, wheelBase: 25, wheelRadius: 10

});

this.bridgeShownFor = null;

this.score = 0;

this.consecutiveCorrect = 0;

this.setScore(0);

this.initVehicleY();

this.paused = true;

document.getElementById('quizContainer')!.style.display = 'none';

document.getElementById('pauseBtn')!.style.display = 'none';

const startBtn = document.getElementById('startBtn')!;

startBtn.style.display = 'block';

startBtn.innerText = 'Play Again';

this.cliff = null;

this.quizReady = false;

this.gameOver = false;

this.nextQuestionAt = null;

}

loop() {

requestAnimationFrame(() => this.loop());

if (this.gameOver || this.paused) return;

this.updateVehicle();

const camX = this.vehicle.x - this.canvas.width \* 0.1;

this.ctx.clearRect(0, 0, this.canvas.width, this.canvas.height);

this.drawTerrain(camX);

this.drawVehicle(camX);

if (this.nextQuestionAt && Date.now() >= this.nextQuestionAt) {

this.spawnCliff();

this.showQuestion();

this.nextQuestionAt = null;

}

if (this.cliff && this.cliff.active && !this.cliff.bridgeShown) {

const frontWheelX = this.vehicle.x + this.vehicle.wheelBase / 2;

const pastCliff = frontWheelX > this.cliff.x + this.cliff.width;

if (frontWheelX > this.cliff.x + this.cliff.width + 100) {

this.quizReady = false;

document.getElementById('quizContainer')!.style.display = 'none';

document.getElementById('pauseBtn')!.style.display = 'none';

}

if (pastCliff || pastCliff && this.vehicle.y > this.getTerrainY(this.vehicle.x) + 100) {

this.SPEED = 0;

this.vehicle.vx = 0;

this.vehicle.vy = 0;

this.quizReady = false;

document.getElementById('quizContainer')!.style.display = 'none';

document.getElementById('pauseBtn')!.style.display = 'none';

this.gameOver = true;

this.cliff = null;

this.soundFalling.play();

setTimeout(() => {

this.soundGameOver.play();

this.resetGame();

}, 5000);

}

}

}

setHighScore(score: number) {

this.highScore = score;

localStorage.setItem('highScore', score.toString());

document.getElementById('highScore')!.textContent = `High Score: ${score}`;

}

setScore(score: number) {

this.score = score > 0 ? score : 0;

document.getElementById('score')!.textContent = `Score: ${this.score}`;

if (score > this.highScore) {

this.setHighScore(score);

}

}

updateScore(points: number) {

this.score += points;

this.setScore(this.score);

}

}

**📄 environments\environment.prod.ts (.ts)**

export const environment = {

production: true

};

**📄 environments\environment.ts (.ts)**

// This file can be replaced during build by using the `fileReplacements` array.

// `ng build` replaces `environment.ts` with `environment.prod.ts`.

// The list of file replacements can be found in `angular.json`.

export const environment = {

production: false

};

/\*

\* For easier debugging in development mode, you can import the following file

\* to ignore zone related error stack frames such as `zone.run`, `zoneDelegate.invokeTask`.

\*

\* This import should be commented out in production mode because it will have a negative impact

\* on performance if an error is thrown.

\*/

// import 'zone.js/plugins/zone-error'; // Included with Angular CLI.

**📄 global.scss (.scss)**

/\*

\* App Global CSS

\* ----------------------------------------------------------------------------

\* Put style rules here that you want to apply globally. These styles are for

\* the entire app and not just one component. Additionally, this file can be

\* used as an entry point to import other CSS/Sass files to be included in the

\* output CSS.

\* For more information on global stylesheets, visit the documentation:

\* https://ionicframework.com/docs/layout/global-stylesheets

\*/

/\* Core CSS required for Ionic components to work properly \*/

@import "@ionic/angular/css/core.css";

/\* Basic CSS for apps built with Ionic \*/

@import "@ionic/angular/css/normalize.css";

@import "@ionic/angular/css/structure.css";

@import "@ionic/angular/css/typography.css";

@import "@ionic/angular/css/display.css";

/\* Optional CSS utils that can be commented out \*/

@import "@ionic/angular/css/padding.css";

@import "@ionic/angular/css/float-elements.css";

@import "@ionic/angular/css/text-alignment.css";

@import "@ionic/angular/css/text-transformation.css";

@import "@ionic/angular/css/flex-utils.css";

/\*\*

\* Ionic Dark Mode

\* -----------------------------------------------------

\* For more info, please see:

\* https://ionicframework.com/docs/theming/dark-mode

\*/

/\* @import "@ionic/angular/css/palettes/dark.always.css"; \*/

/\* @import "@ionic/angular/css/palettes/dark.class.css"; \*/

@import '@ionic/angular/css/palettes/dark.system.css';

html,

body {

margin: 0;

background: #FFEBC6;

font-family: sans-serif;

touch-action: manipulation;

height: 100%;

}

**📄 index.html (.html)**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<title>Ionic App</title>

<base href="/" />

<meta name="color-scheme" content="light dark" />

<meta name="viewport" content="viewport-fit=cover, width=device-width, initial-scale=1.0, minimum-scale=1.0, maximum-scale=1.0, user-scalable=no" />

<meta name="format-detection" content="telephone=no" />

<meta name="msapplication-tap-highlight" content="no" />

<link rel="icon" type="image/png" href="assets/icon/favicon.png" />

<!-- add to homescreen for ios -->

<meta name="mobile-web-app-capable" content="yes" />

<meta name="apple-mobile-web-app-status-bar-style" content="black" />

</head>

<body>

<app-root></app-root>

</body>

</html>

**📄 main.ts (.ts)**

import { bootstrapApplication } from '@angular/platform-browser';

import { RouteReuseStrategy, provideRouter, withPreloading, PreloadAllModules } from '@angular/router';

import { IonicRouteStrategy, provideIonicAngular } from '@ionic/angular/standalone';

import { routes } from './app/app.routes';

import { AppComponent } from './app/app.component';

bootstrapApplication(AppComponent, {

providers: [

{ provide: RouteReuseStrategy, useClass: IonicRouteStrategy },

provideIonicAngular(),

provideRouter(routes, withPreloading(PreloadAllModules)),

],

});

**📄 polyfills.ts (.ts)**

/\*\*

\* This file includes polyfills needed by Angular and is loaded before the app.

\* You can add your own extra polyfills to this file.

\*

\* This file is divided into 2 sections:

\* 1. Browser polyfills. These are applied before loading ZoneJS and are sorted by browsers.

\* 2. Application imports. Files imported after ZoneJS that should be loaded before your main

\* file.

\*

\* The current setup is for so-called "evergreen" browsers; the last versions of browsers that

\* automatically update themselves. This includes recent versions of Safari, Chrome (including

\* Opera), Edge on the desktop, and iOS and Chrome on mobile.

\*

\* Learn more in https://angular.io/guide/browser-support

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* BROWSER POLYFILLS

\*/

/\*\*

\* By default, zone.js will patch all possible macroTask and DomEvents

\* user can disable parts of macroTask/DomEvents patch by setting following flags

\* because those flags need to be set before `zone.js` being loaded, and webpack

\* will put import in the top of bundle, so user need to create a separate file

\* in this directory (for example: zone-flags.ts), and put the following flags

\* into that file, and then add the following code before importing zone.js.

\* import './zone-flags';

\*

\* The flags allowed in zone-flags.ts are listed here.

\*

\* The following flags will work for all browsers.

\*

\* (window as any).\_\_Zone\_disable\_requestAnimationFrame = true; // disable patch requestAnimationFrame

\* (window as any).\_\_Zone\_disable\_on\_property = true; // disable patch onProperty such as onclick

\* (window as any).\_\_zone\_symbol\_\_UNPATCHED\_EVENTS = ['scroll', 'mousemove']; // disable patch specified eventNames

\*

\* in IE/Edge developer tools, the addEventListener will also be wrapped by zone.js

\* with the following flag, it will bypass `zone.js` patch for IE/Edge

\*

\* (window as any).\_\_Zone\_enable\_cross\_context\_check = true;

\*

\*/

import './zone-flags';

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Zone JS is required by default for Angular itself.

\*/

import 'zone.js'; // Included with Angular CLI.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* APPLICATION IMPORTS

\*/

**📄 test.ts (.ts)**

// This file is required by karma.conf.js and loads recursively all the .spec and framework files

import 'zone.js/testing';

import { getTestBed } from '@angular/core/testing';

import {

BrowserDynamicTestingModule,

platformBrowserDynamicTesting

} from '@angular/platform-browser-dynamic/testing';

// First, initialize the Angular testing environment.

getTestBed().initTestEnvironment(

BrowserDynamicTestingModule,

platformBrowserDynamicTesting(),

);

**📄 theme\variables.scss (.scss)**

// For information on how to create your own theme, please see:

// http://ionicframework.com/docs/theming/

**📄 zone-flags.ts (.ts)**

/\*\*

\* Prevents Angular change detection from

\* running with certain Web Component callbacks

\*/

// eslint-disable-next-line no-underscore-dangle

(window as any).\_\_Zone\_disable\_customElements = true;