**Структурирана, балансирана презентация за трима души**, така че всеки да говори ~5 минути, да покрие своя основен компонент (**Canvas**, **SVG**, **WebGL**), но и да се включи въведение и заключение без претоварване.

**Общи параметри:**

* Общо време: 15 минути
* 3 участници x 5 минути
* Общо слайдове: около **12 слайда**
* Всеки представя един компонент **+ общи теми разпределени стратегически**

**Разпределение на презентацията**

| **Човек** | **Отговорности** | **Минутно разпределение** | **Слайдове** |
| --- | --- | --- | --- |
| **Човек 1** | Въведение + Canvas | ~5 мин | Слайд 1–4 |
| **Човек 2** | SVG + сравнение | ~5 мин | Слайд 5–8 |
| **Човек 3** | WebGL + заключение | ~5 мин | Слайд 9–12 |

**Детайли по слайдове и говорещи:**

**Човек 1: Въведение + Canvas (~5 мин)**

**Слайд 1: Заглавие и представяне**

* „Canvas, SVG и WebGL – алтернатива на HTML+CSS“

**Слайд 2: Какво представлява традиционният подход (HTML+CSS+DOM)**

* DOM дърво, семантика, достъпност, но проблем при анимации/графики

**Слайд 3: Какво е Canvas?**

* Пикселна графика, JS-рисуване, без DOM елементи
* Употреба: игри, анимации, редактори

**Слайд 4: Live Demo: Canvas**

* Анимирано кръгче (от файла)

**Човек 2: SVG + Сравнение (~5 мин)**

**Слайд 5: Какво е SVG?**

* Векторна, DOM-базирана, стил с CSS
* Подходяща за икони, графики, диаграми

**Слайд 6: Live Demo: SVG**

* SVG кръг с hover ефект

**Слайд 7: Сравнение между Canvas, SVG и WebGL**

* Табличен формат: графика, производителност, контрол

**Слайд 8: Защо да ги изберем пред HTML+CSS?**

* Производителност, визуална свобода, специфични use cases

**Човек 3: WebGL + Заключение (~5 мин)**

**Слайд 9: Какво е WebGL?**

* 3D, базирано на OpenGL, хардуерно ускорено
* Често се използва с Three.js

**Слайд 10: Live Demo: WebGL**

* Въртящ се куб с Three.js

**Слайд 11: Кога да ги използваме?**

* Дашборди, 3D визуализации, игри
* Комбинация между DOM + графични технологии

**Слайд 12: Заключение**

* HTML+CSS ≠ винаги най-доброто
* Canvas/SVG/WebGL = мощни алтернативи при нужда от динамична визуализация

**Допълнително:**

* Всеки презентатор **отваря и показва live demo** за своята част (~1–1.5 мин от времето му)

Включване на **"Разработка на приложения чрез тези технологии като алтернатива на HTML+CSS. Причини за този избор, разлика с DOM-базирани приложения"** в презентацията:

**1. "Разработка на приложения чрез тези технологии като алтернатива на HTML+CSS"**

Включено в:

* **Слайд 2 (Човек 1)** – Обяснение защо стандартният HTML+CSS подход има ограничения
* **Слайд 7–8 (Човек 2)** – Сравнение с DOM, предимства на SVG/Canvas/WebGL
* **Слайд 11 (Човек 3)** – Примери за реални приложения: игри, редактори, визуализации

*Как да се подчертае по-добре:*  
Добави подзаглавие в Слайд 11:  
„📱 Реални приложения с тези технологии“

**2. "Причини за този избор"**

Включено в:

* **Слайд 3 (Canvas)** – Причини: анимации, игри, производителност
* **Слайд 5 (SVG)** – Причини: вектор, интерактивност, responsive
* **Слайд 9 (WebGL)** – Причини: 3D, хардуерно ускорено

*Как да се подчертае по-добре:*  
В отделен **слайд с обобщени причини** преди заключението – например:

**Нов Слайд 11 (старият 11 → 12):**

**„Причини да изберем Canvas/SVG/WebGL пред HTML+CSS“**

* По-добър контрол върху визуализацията
* По-висока производителност
* Подходящи за динамични, графично-интензивни приложения

**3. "Разлика с DOM-базирани приложения"**

Включено в:

* **Слайд 2** – обяснение на DOM, традиционен подход
* **Слайд 7** – Таблично сравнение DOM vs Canvas vs SVG vs WebGL

*Как да се подчертае по-добре:*  
Добавете в таблицата допълнителна колона или ред „**Работи с DOM?**“ – да/не  
Може да се покаже визуално:

* Canvas = пикселно, без DOM
* SVG = в DOM
* WebGL = JS-базирано, WebGL контекст, извън DOM

**Обновено предложение за структура:**

| **Слайд** | **Тема** | **Говорител** | **Бележка** |
| --- | --- | --- | --- |
| 1 | Заглавие | 1 | Начало |
| 2 | HTML+CSS ограничения | 1 | Сравнение с DOM |
| 3 | Canvas теория | 1 | Причини |
| 4 | Canvas demo | 1 | Анимация |
| 5 | SVG теория | 2 | Причини |
| 6 | SVG demo | 2 | Икона |
| 7 | Таблично сравнение | 2 | DOM/Canvas/SVG/WebGL |
| 8 | Предимства спрямо HTML+CSS | 2 | Ключови причини |
| 9 | WebGL теория | 3 | 3D графика |
| 10 | WebGL demo | 3 | Въртящ се куб |
| 11 | Реални приложения | 3 | Примери за app разработки |
| 12 | Заключение | 3 | Сумарно |

**Инструкции: Как да направите демонстрациите стъпка по стъпка**

**1. Софтуер:**

* **Не е нужно нищо специално**, само:
  + Текстов редактор като **VS Code**, **Notepad++**, или дори **Notepad**
  + Модерен браузър (**Chrome**, **Firefox**, **Edge**) за отваряне на HTML файловете

**2. Стъпки за подготовка на всяко демо:**

**Canvas Demo:**

Файл: canvas.html  
Цел: Анимирано движещо се кръгче

**Стъпки:**

1. Отвори canvas.html с текстов редактор
2. Прегледай кода — canvas елемент и JS за рисуване и анимация
3. Отвори файла в браузър → ще видиш движещо се кръгче
4. По желание: промени цвета или размера на кръгчето

**SVG Demo:**

Файл: svg.html  
Цел: Векторен кръг, който сменя цвят при hover

**Стъпки:**

1. Отвори svg.html с текстов редактор
2. Прегледай SVG елемента (circle) и CSS hover стил
3. Отвори файла в браузър → задръж мишката върху кръга, ще смени цвят
4. По желание: добави още SVG фигури или анимация

**WebGL Demo (с Three.js):**

Файл: webgl.html  
Цел: Въртящ се 3D куб

**Стъпки:**

1. Отвори webgl.html с текстов редактор
2. Виж как се използва Three.js (зареден от CDN линк)
3. Отвори в браузър → ще се появи въртящ се 3D куб
4. Можеш да промениш цветовете, размера, или да добавиш още геометрии

**Бележки за говорене:**

**Човек 1 (Canvas):**

* „Canvas работи на принципа на пикселна графика – нямаме DOM елементи, а директно рисуваме“
* „Подходящо е за игри, анимации и визуализации“
* „В демонстрацията виждаме как едно кръгче се движи с помощта на requestAnimationFrame()“

**Човек 2 (SVG):**

* „SVG е векторна графика – тя е част от DOM, което позволява интерактивност и стилове чрез CSS“
* „Подходящо е за икони, диаграми, графики“
* „В нашия пример имаме кръг, който сменя цвят при задържане на мишката“

**Човек 3 (WebGL):**

* „WebGL дава достъп до хардуерно ускорена 3D графика – базирана на OpenGL“
* „Използваме Three.js – популярен библиотечен wrapper“
* „Демонстрираме въртящ се куб – типично начало на 3D сцена“

**Code for Demo 1 – Canvas:**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>Canvas Demo</title>

  <style>

    body { background: #f0f0f0; text-align: center; padding: 40px; font-family: sans-serif; }

    canvas { border: 1px solid #ccc; }

  </style>

</head>

<body>

  <h2>Canvas Demo – Анимирано кръгче</h2>

  <canvas id="myCanvas" width="400" height="200"></canvas>

  <script>

    const canvas = document.getElementById("myCanvas");

    const ctx = canvas.getContext("2d");

    let x = 50;

    function draw() {

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

      ctx.beginPath();

      ctx.arc(x, 100, 30, 0, Math.PI \* 2);

      ctx.fillStyle = "#3498db";

      ctx.fill();

      x += 2;

      if (x > canvas.width) x = -30;

      requestAnimationFrame(draw);

    }

    draw();

  </script>

</body>

</html>

**Code for Demo 2 – SVG:**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>SVG Demo</title>

  <style>

    body { background: #f0f0f0; text-align: center; padding: 40px; font-family: sans-serif; }

    #circle:hover { fill: crimson; cursor: pointer; }

  </style>

</head>

<body>

  <h2>SVG Demo – Интерактивна икона</h2>

  <svg width="120" height="120">

    <circle id="circle" cx="60" cy="60" r="50" fill="orange" />

  </svg>

</body>

</html>

**Code for Demo 3 – WebGL:**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>WebGL Demo</title>

  <style>

    body { background: #f0f0f0; text-align: center; padding: 40px; font-family: sans-serif; }

    #container { margin: 0 auto; width: 400px; height: 300px; }

  </style>

</head>

<body>

  <h2>WebGL Demo – Въртящ се 3D куб</h2>

  <div id="container"></div>

  <script src="https://cdnjs.cloudflare.com/ajax/libs/three.js/r134/three.min.js"></script>

  <script>

    const scene = new THREE.Scene();

    const camera = new THREE.PerspectiveCamera(75, 400/300, 0.1, 1000);

    const renderer = new THREE.WebGLRenderer();

    renderer.setSize(400, 300);

    document.getElementById("container").appendChild(renderer.domElement);

    const geometry = new THREE.BoxGeometry();

    const material = new THREE.MeshNormalMaterial();

    const cube = new THREE.Mesh(geometry, material);

    scene.add(cube);

    camera.position.z = 3;

    function animate() {

      requestAnimationFrame(animate);

      cube.rotation.x += 0.01;

      cube.rotation.y += 0.01;

      renderer.render(scene, camera);

    }

    animate();

  </script>

</body>

</html>