

—— ATIVIDADE PRÁTICA – JUMP GAME JS

Jump Game:

- Dentre as possíveis abordagens para a criação de um jogo simples em JS, aqui se escolheu por utilizar o arquivo **JS** fora do arquivo **HTML** principal, da mesma forma que ocorre com o **CSS**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Jump Game</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  
  <div class="game">
    <div id="character"></div>
    <div id="block"></div>
  </div>
  <p>Pontuação: <span id="scoreSpan">0</span></p>
  <script src="script.js"></script>
</body>
</html>
```

—— ATIVIDADE PRÁTICA – JUMP GAME JS

Jump Game:

Passo 1: Estruturando a página com HTML

O HTML serve como a base do jogo, fornecendo a estrutura e os elementos que usaremos.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Jump Game</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  
    <div id="character"></div>
    <div id="block"></div>
  </div>
  <p>Pontuação: <span id="scoreSpan">0</span></p>
  <script src="script.js"></script>
</body>
</html>
```

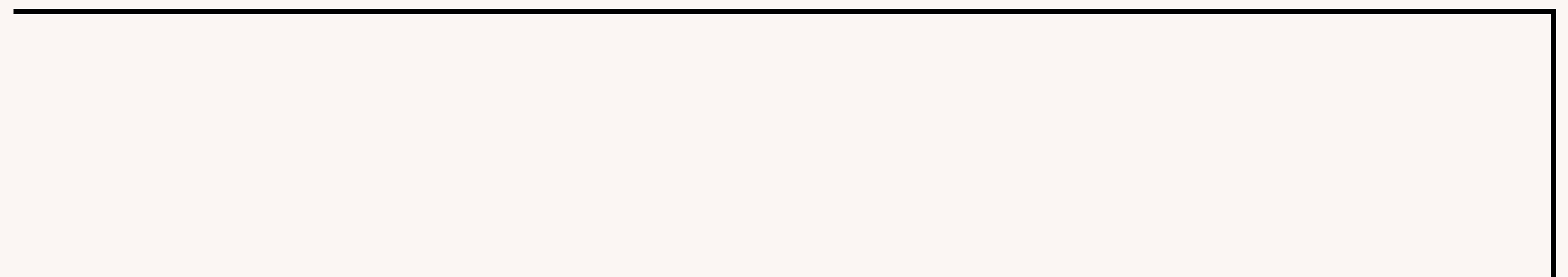
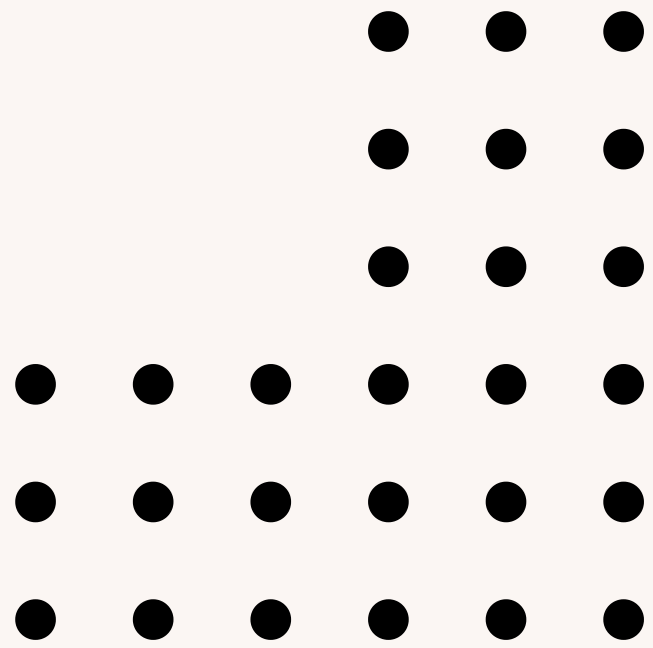
—— ATIVIDADE PRÁTICA – JUMP GAME JS

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Jump Game</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  
  <div class="game">
    <div id="character"></div>
    <div id="block"></div>
  </div>
  <p>Pontuação: <span id="scoreSpan">0</span></p>
  <script src="script.js"></script>
</body>
</html>
```

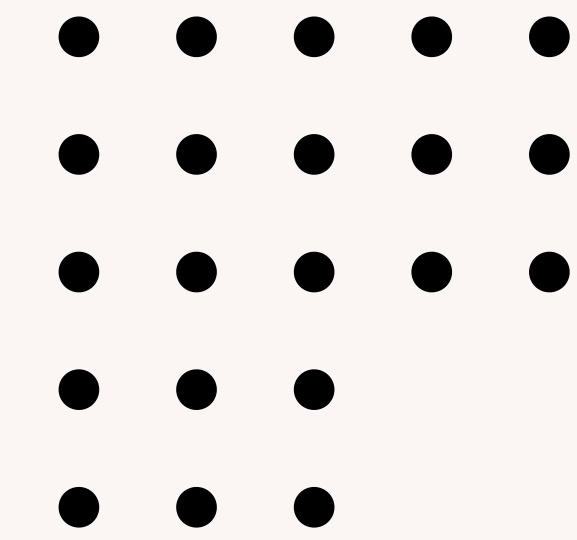
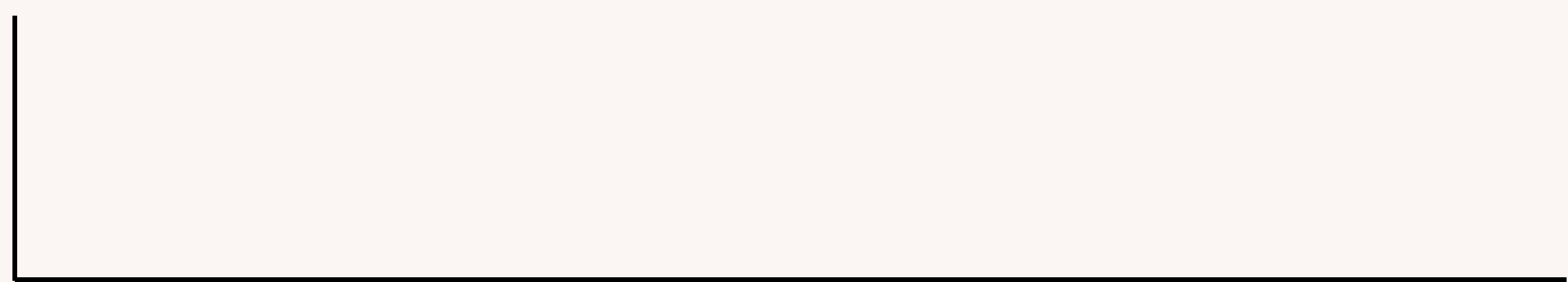
—— ATIVIDADE PRÁTICA – JUMP GAME JS

Jump Game:

- ``: Adiciona a imagem de fundo com a classe `filtered-image`.
- `<div class="game">`: Contêiner principal do jogo.
- `<div id="character"></div>`: Representa o personagem controlado pelo jogador.
- `<div id="block"></div>`: Representa o obstáculo que se move na tela.
- `<p>`: Exibe a pontuação atual do jogador.
- `<script src="script.js"></script>`: Vincula o arquivo JavaScript para adicionar interatividade.



CSS



—— ATIVIDADE PRÁTICA – JUMP GAME JS

Jump Game:

- 1 - O CSS é usado para dar vida ao jogo, definindo o estilo e posicionamento dos elementos. Vamos adicionar a aparência para o personagem, o obstáculo, e a imagem de fundo.

```
1  * {
2    padding: 0;
3    margin: 0;
4    overflow-x: hidden;
5  }
6
7  body {
8    background-color: rgb(0, 0, 0);
9  }
10
11  .filtered-image {
12    mix-blend-mode: hard-light;
13    width: 450px;
14    height: 300px;
15    position: absolute;
16    top: 10px;
17    left: 50%;
18    transform: translateX(-50%);
19  }
20
21  .game {
22    width: 750px;
23    height: 200px;
24    border: 1px solid white;
25    margin: auto;
26    position: absolute;
27    top: 40px;
28    left: 50%;
29    transform: translateX(-50%);
30  }
31
32  #character {
33    width: 20px;
34    height: 50px;
35    background-color: red;
36    position: relative;
37    top: 150px;
38  }
39
40  .animate {
41    animation: jump 0.5s linear;
42  }
43
44  @keyframes jump {
45    0% { top: 150px; }
46    30% { top: 100px; }
47    70% { top: 100px; }
48    100% { top: 150px; }
49  }
50
51  #block {
52    background-color: blue;
53    width: 20px;
54    height: 20px;
55    position: relative;
56    top: 130px;
57    left: 500px;
58    animation: block 3s infinite linear;
59  }
60
61  @keyframes block {
62    0% { left: 750px; }
63    100% { left: -20px; }
64  }
65
66  p {
67    text-align: center;
68    color: #0af7ff;
69    font-size: 28px;
70    font-weight: bold;
71    position: absolute;
72    top: 300px;
73    left: 50%;
74    transform: translateX(-50%);
75  }
```


—— ATIVIDADE PRÁTICA – JUMP GAME JS

1 - Reset de Estilos Globais:

- Remove as margens e preenchimentos padrão de todos os elementos.
- `overflow-x: hidden;`: Evita a barra de rolagem horizontal.

2 - Estilo do Corpo:

- Define a cor de fundo da página como preto.

```
1  * {
2      padding: 0;
3      margin: 0;
4      overflow-x: hidden;
5  }
6
7  body {
8      background-color:  rgb(0, 0, 0);
9  }
10
```

—— ATIVIDADE PRÁTICA – JUMP GAME JS

3 - Imagem de Fundo:

- `.filtered-image`: Classe aplicada à imagem de fundo.
- `mix-blend-mode: hard-light`:: Aplica um efeito de mistura de cores.
- `width` e `height`: Define as dimensões da imagem.
- `position: absolute; top: 10px; left: 50%; transform: translateX(-50%);`:: Centraliza a imagem horizontalmente na página.

```
11  .filtered-image {  
12      mix-blend-mode: hard-light;  
13      width: 450px;  
14      height: 300px;  
15      position: absolute;  
16      top: 10px;  
17      left: 50%;  
18      transform: translateX(-50%);  
19  }
```


—— ATIVIDADE PRÁTICA – JUMP GAME JS

4 - Área do Jogo:


- `..game`: Contêiner do jogo.
- `width` e `height`: Define o tamanho da área de jogo.
- `border`: Adiciona uma borda branca ao redor da área de jogo.
- `margin: auto`;: Centraliza o contêiner.
- `position: absolute; top: 40px; left: 50%; transform: translateX(-50%);`: Posiciona o contêiner no centro da página com um deslocamento vertical.

```
21  .game {  
22      width: 750px;  
23      height: 200px;  
24      border: 1px solid white;  
25      margin: auto;  
26      position: absolute;  
27      top: 40px;  
28      left: 50%;  
29      transform: translateX(-50%);  
30  }
```

—— ATIVIDADE PRÁTICA – JUMP GAME JS

5 - Personagem:

- #character: ID aplicado ao personagem.
- width e height: Define o tamanho do personagem.
- background-color: Define a cor do personagem como vermelho.
- position: relative; top: 150px;: Posiciona o personagem na base da área de jogo.

```
32  #character {  
33      width: 20px;  
34      height: 50px;  
35      background-color:  red;  
36      position: relative;  
37      top: 150px;  
38  }  
39
```

—— ATIVIDADE PRÁTICA – JUMP GAME JS

6 - Animação de Salto:

- .animate: Classe que aplica a animação de salto ao personagem.
- @keyframes jump: Define a animação de salto, movendo o personagem para cima e de volta.

```
40  .animate {  
41  |    animation: jump 0.5s linear;  
42  }  
43  
44  @keyframes jump {  
45  |    0% { top: 150px; }  
46  |    30% { top: 100px; }  
47  |    70% { top: 100px; }  
48  |    100% { top: 150px; }  
49  }  
50
```

—— ATIVIDADE PRÁTICA – JUMP GAME JS

7 - Obstáculo (Bloco):


- #block: ID aplicado ao obstáculo.
- background-color: Define a cor do obstáculo como azul.
- width e height: Define o tamanho do obstáculo.
- position: relative; top: 130px; left: 500px;: Posiciona o obstáculo na área de jogo.
- animation: block 3s infinite linear;: Anima o obstáculo da direita para a esquerda continuamente

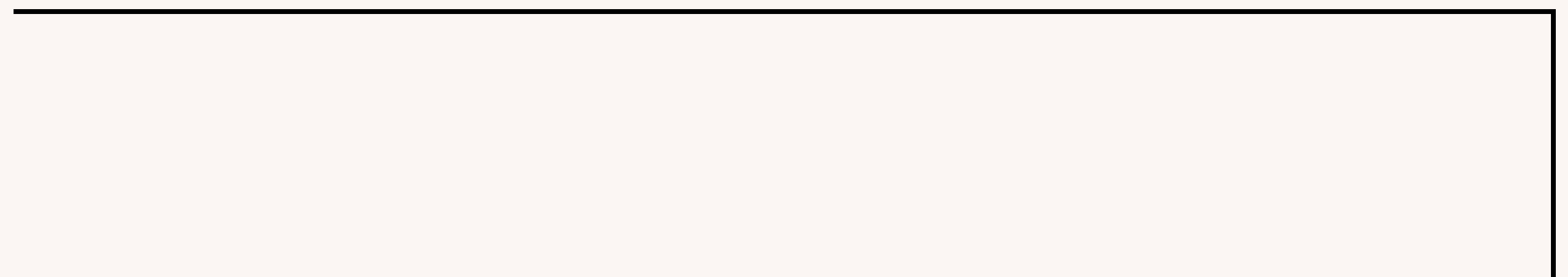
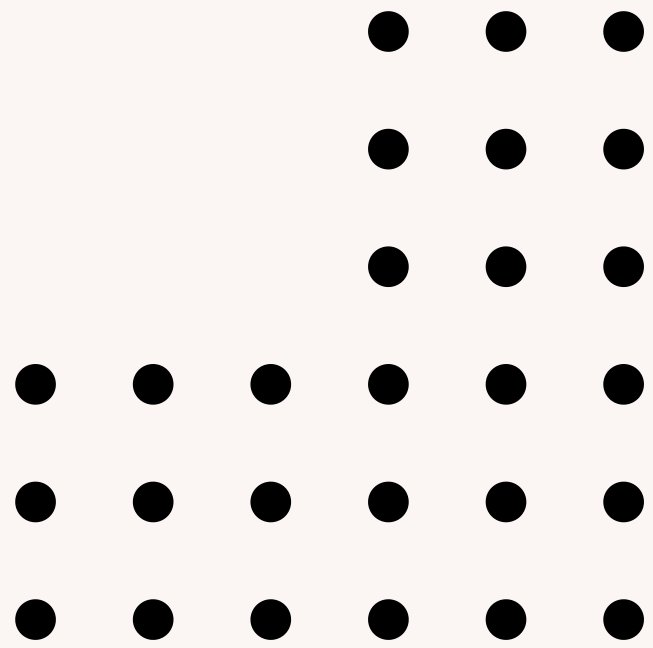
```
50
51  #block {
52      background-color: blue;
53      width: 20px;
54      height: 20px;
55      position: relative;
56      top: 130px;
57      left: 500px;
58      animation: block 3s infinite linear;
59  }
60
61  @keyframes block {
62      0% { left: 750px; }
63      100% { left: -20px; }
64  }
```

—— ATIVIDADE PRÁTICA – JUMP GAME JS

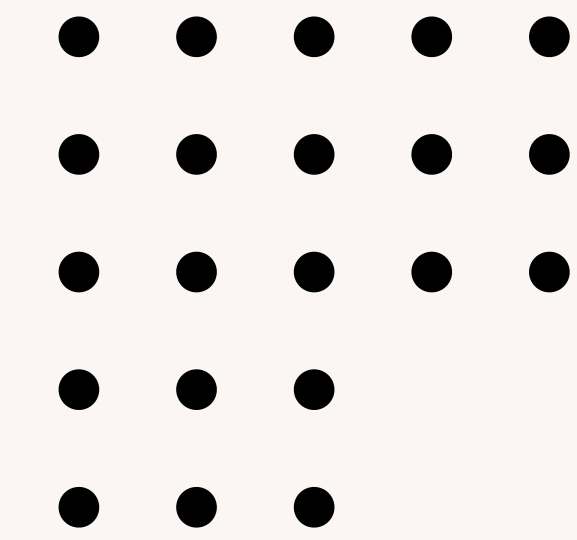
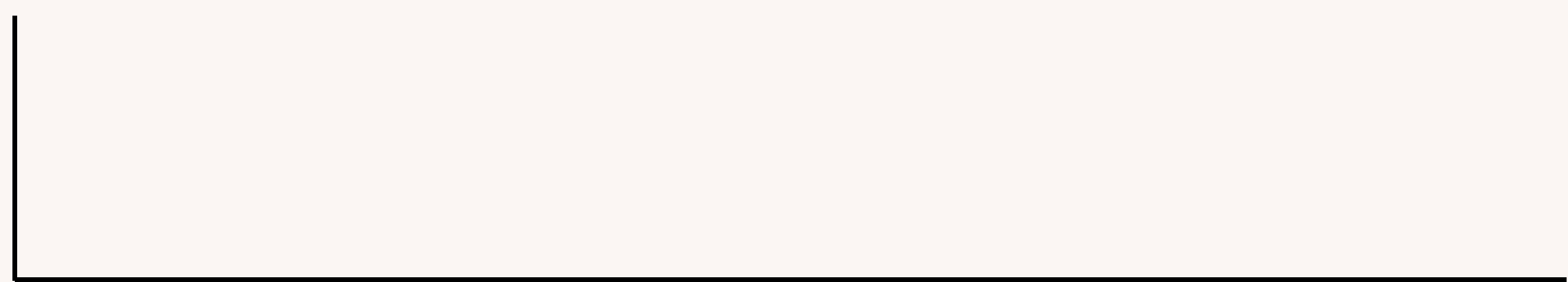
8 - Pontuação:

- Estiliza o parágrafo que exibe a pontuação.
- Centraliza o texto e define a cor, tamanho e peso da fonte.
- Posiciona a pontuação abaixo da área de jogo.

```
66  p {  
67      text-align: center;  
68      color:  #0af7ff;  
69      font-size: 28px;  
70      font-weight: bold;  
71      position: absolute;  
72      top: 300px;  
73      left: 50%;  
74      transform: translateX(-50%);  
75  }
```



JS



— ATIVIDADE PRÁTICA – JUMP GAME JS

Jump Game:

1 - O JavaScript adiciona a funcionalidade de salto, detecção de colisão, controle de pontuação e lógica de fim de jogo.

```
1  const character = document.getElementById("character");
2  const block = document.getElementById("block");
3  let counter = 0;
4  let isJumping = false;
5
6  document.addEventListener("keydown", function(event) {
7      if (event.code === "Space") {
8          jump();
9      }
10 });
11
12 function jump() {
13     if (!isJumping) {
14         isJumping = true;
15         character.classList.add("animate");
16         setTimeout(() => {
17             character.classList.remove("animate");
18             isJumping = false;
19         }, 500);
20     }
21 }
22
23 const checkDead = setInterval(() => {
24     const characterTop = parseInt(window.getComputedStyle(character).getPropertyValue("top"));
25     const blockLeft = parseInt(window.getComputedStyle(block).getPropertyValue("left"));
26
27     if (blockLeft < 20 && blockLeft > 0 && characterTop >= 130) {
28         block.style.animation = "none";
29         alert("Fim de jogo. Pontuação: " + Math.floor(counter / 100));
30         counter = 0;
31         block.style.animation = "block 3s infinite linear";
32     } else {
33         counter++;
34         document.getElementById("scoreSpan").innerText = Math.floor(counter / 100);
35     }
36 }, 10);
37
```

—— ATIVIDADE PRÁTICA – JUMP GAME JS

1-Seleção de Elementos e Variáveis:

- character: Seleciona o elemento do personagem.
- block: Seleciona o elemento do obstáculo.
- counter: Armazena a pontuação do jogador.
- isJumping: Flag para indicar se o personagem está saltando, evitando múltiplos saltos simultâneos.

```
1  const character = document.getElementById("character");  
2  const block = document.getElementById("block");  
3  let counter = 0;  
4  let isJumping = false;
```


—— ATIVIDADE PRÁTICA – JUMP GAME JS

2-Event Listener para o Salto:

- Adiciona um ouvinte de evento para detectar quando a barra de espaço é pressionada.
- Quando a barra de espaço (Space) é pressionada, chama a função jump().

```
6 document.addEventListener("keydown", function(event) {  
7     if (event.code === "Space") {  
8         jump();  
9     }  
10 });
```

—— ATIVIDADE PRÁTICA – JUMP GAME JS

3-Função de Salto:

- Verifica se o personagem já está saltando para evitar saltos consecutivos.
- Adiciona a classe animate para iniciar a animação de salto.
- Usa setTimeout para remover a classe após 500ms (duração da animação), permitindo que o personagem volte ao estado original e permita novos saltos.

```
12 function jump() {  
13     if (!isJumping) {  
14         isJumping = true;  
15         character.classList.add("animate");  
16         setTimeout(() => {  
17             character.classList.remove("animate");  
18             isJumping = false;  
19         }, 500);  
20     }  
21 }
```

—— ATIVIDADE PRÁTICA – JUMP GAME JS

4-Detecção de Colisão e Pontuação:

- setInterval: Executa a função a cada 10ms para verificar constantemente a posição do personagem e do obstáculo.
- characterTop: Obtém a posição vertical atual do personagem.
- blockLeft: Obtém a posição horizontal atual do obstáculo.

```
const checkDead = setInterval(() => {
  const characterTop = parseInt(window.getComputedStyle(character).getPropertyValue("top"));
  const blockLeft = parseInt(window.getComputedStyle(block).getPropertyValue("left"));

  if (blockLeft < 20 && blockLeft > 0 && characterTop >= 130) {
    block.style.animation = "none";
    alert("Fim de jogo. Pontuação: " + Math.floor(counter / 100));
    counter = 0;
    block.style.animation = "block 3s infinite linear";
  } else {
    counter++;
    document.getElementById("scoreSpan").innerText = Math.floor(counter / 100);
  }
}, 10);
```

—— ATIVIDADE PRÁTICA – JUMP GAME JS

4-Detecção de Colisão e Pontuação:

- **Condição de Colisão:**

- `blockLeft < 20 && blockLeft > 0`: Verifica se o obstáculo está na mesma posição horizontal que o personagem.
- `characterTop >= 130`: Verifica se o personagem não está no ar (indicando uma colisão).

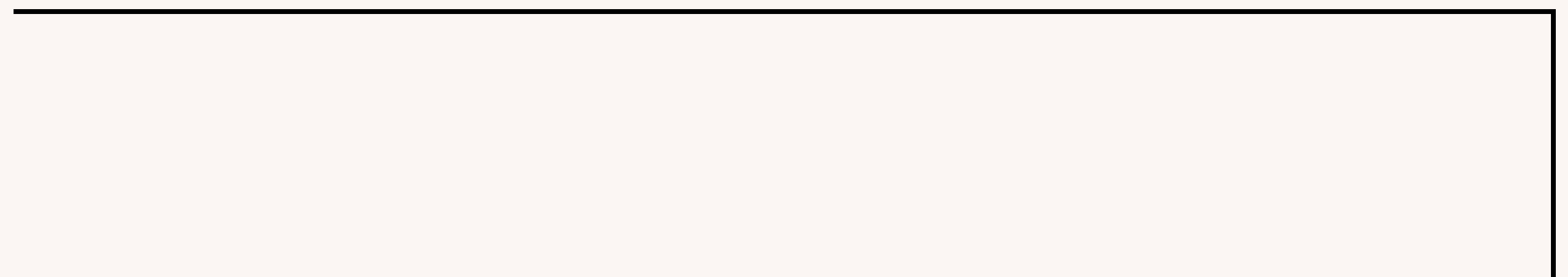
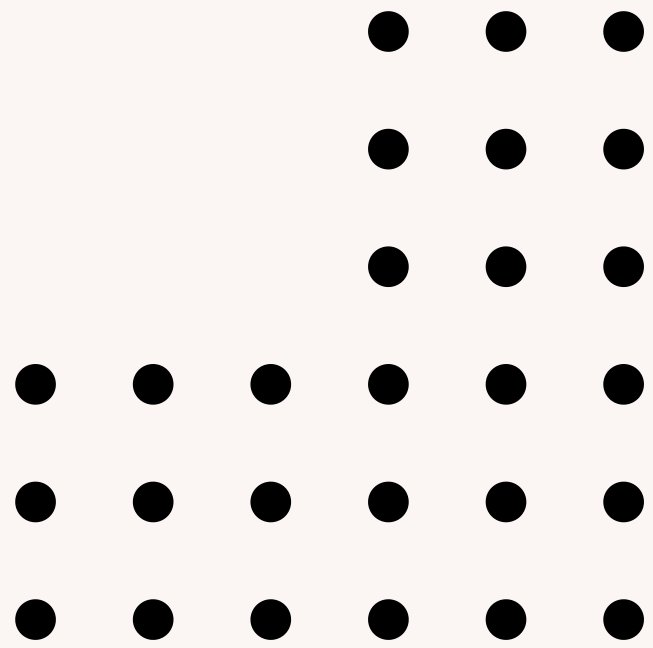
```
const checkDead = setInterval(() => {  
  const characterTop = parseInt(window.getComputedStyle(character).getPropertyValue("top"));  
  const blockLeft = parseInt(window.getComputedStyle(block).getPropertyValue("left"));  
  
  if (blockLeft < 20 && blockLeft > 0 && characterTop >= 130) {  
    block.style.animation = "none";  
    alert("Fim de jogo. Pontuação: " + Math.floor(counter / 100));  
    counter = 0;  
    block.style.animation = "block 3s infinite linear";  
  } else {  
    counter++;  
    document.getElementById("scoreSpan").innerText = Math.floor(counter / 100);  
  }  
}, 10);
```

—— ATIVIDADE PRÁTICA – JUMP GAME JS

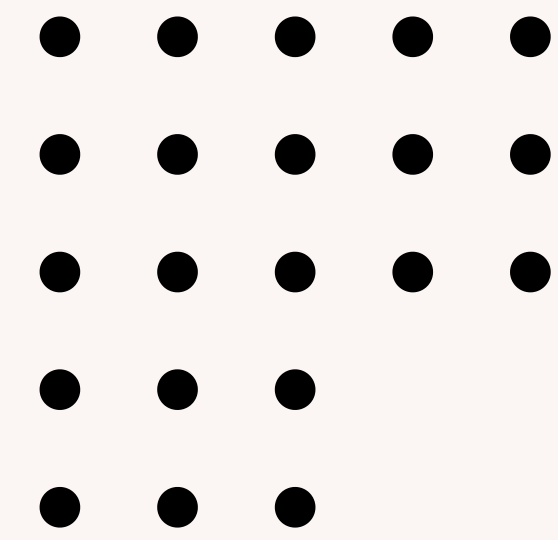
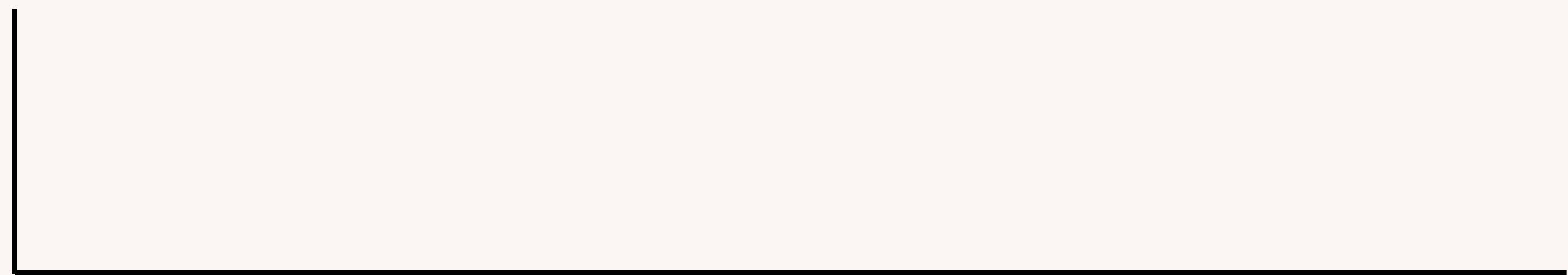
4-Detecção de Colisão e Pontuação:

- **Ação em Caso de Colisão:**
 - Para a animação do obstáculo (block.style.animation = "none";).
 - Exibe um alerta com a pontuação final.
 - Reseta a pontuação e reinicia a animação do obstáculo.
- **Atualização da Pontuação:**
 - Incrementa o contador a cada intervalo.
 - Atualiza o elemento scoreSpan para exibir a pontuação atualizada.

```
const checkDead = setInterval(() => {  
  const characterTop = parseInt(window.getComputedStyle(character).getPropertyValue("top"));  
  const blockLeft = parseInt(window.getComputedStyle(block).getPropertyValue("left"));  
  
  if (blockLeft < 20 && blockLeft > 0 && characterTop >= 130) {  
    block.style.animation = "none";  
    alert("Fim de jogo. Pontuação: " + Math.floor(counter / 100));  
    counter = 0;  
    block.style.animation = "block 3s infinite linear";  
  } else {  
    counter++;  
    document.getElementById("scoreSpan").innerText = Math.floor(counter / 100);  
  }  
}, 10);
```



ATIVIDADE



ATIVIDADE

Com base no “jogo” apresentado, façam as seguintes modificações e adições:

- **Adicionar função de pausa**
- **Adicionar movimentação horizontal para o “player”**
- **Adicionar mais de um elemento de obstáculo no cenário, com variação de altura e cor**