Aula 13

Sites Interativos

O que são sites interativos

Sites interativos são páginas da WEB que possuem algo além de apenas o conteúdo do site e que com base na interação do usuário apresentam alguma funcionalidade estética. Podemos citar por exemplo um site onde quando você passa o mouse nas letras, elas mudam de cor.

Os sites interativos tem como principal objetivo se destacar dos demais e entregar uma experiência mais divertida ao usuário.

Testando um site interativo

Vamos testar um site interativo para uma melhor compreensão. Siga os passos abaixo para realizar o teste.

1. Acesse o site: https://www.nike-react.com/.

2. Clique no botão “Go”.

3. Escolha sua três objetos apresentados de customização e clique no botão “next”.

4. .Escolha as demais customizações(próximas duas) e aperte em “next”.

5. Aparecerá um personagem criado com base nas opções de customização que você escolheu. Clique nele para alterar o modo como os elementos foram utilizados em sua construção.

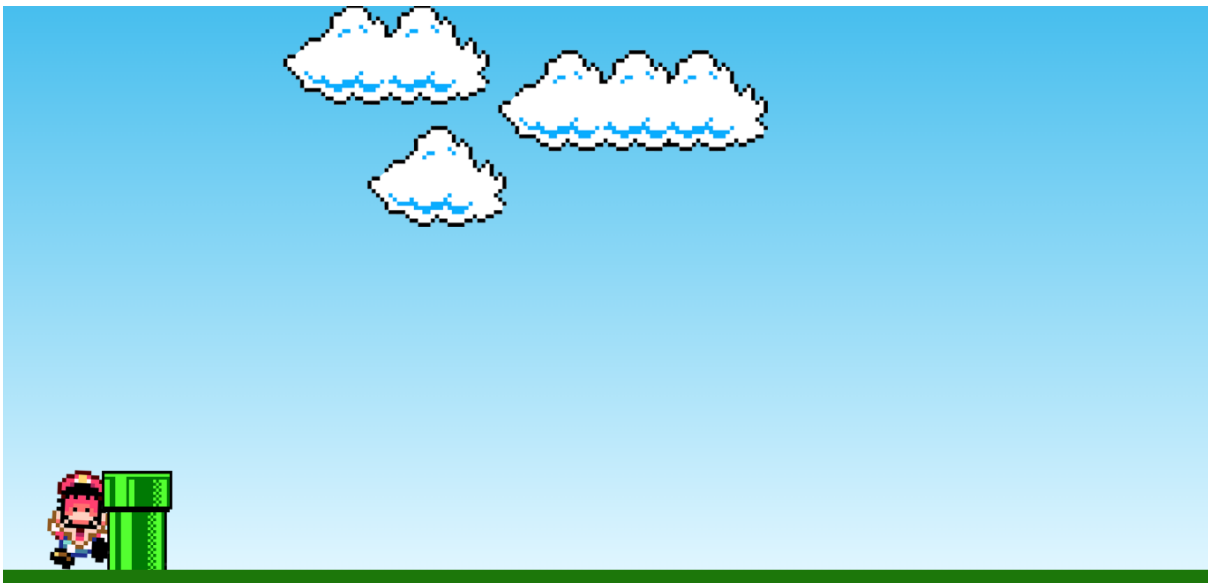
Lista de sites interativos interessantes

Aqui vão alguns sites interativos bastante interessantes, dá para se distrair por algumas boas horas.

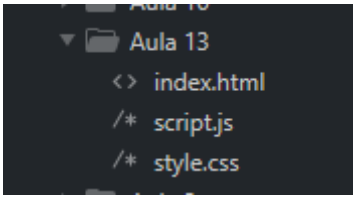
- Find the Invisible Cow
- Hacker Typer
- Staggering Beauty
- Silk
- Binary Music Player
- The Long Doge Challenge
- Checkbox Race!
- Cursor Effects
- drawing.garden
- Koalas to the Max
- JacksonPollock.org

Mario Jump

Na aula de hoje, vamos criar nosso primeiro site interativo. Ele será no formato de um jogo, semelhante ao jogo do dinossauro do Google.



Vamos começar criando a pasta da nossa aula, onde haverá todos os arquivos que vamos precisar para criar o projeto do jogo. Não se esqueça de criar um arquivo html, um css e um javascript, inicialmente vazios.



Estrutura HTML

No arquivo HTML, insira a estrutura básica, e os links para o arquivo CSS e JavaScript. Note que diferente das outras aulas, o script para o JS precisa ser inserido após o Body, pois suas funções serão chamadas posteriormente à página ser carregada. Não se esqueça de adicionar um título.

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- Link do CSS -->
    <link rel="stylesheet" type="text/css" href="style.css">
    <title>Mario Jump</title>
</head>

<body>

</body>

<!-- Script do JavaScript -->
<script src="script.js"></script>
</html>
```

Dentro do Body, adicione uma Div e atribua a ela a classe “game-board”. Essa Div será a delimitação de onde ocorre a tela do jogo. Aproveite para adicionar as 3 principais imagens que vamos usar e suas respectivas classes, como mostra no exemplo abaixo:

```html
<body>
    <div class="game-board">
        <img src="img/clouds.png" class="clouds">
        <img src="img/mario.gif" class="mario">
        <img src="img/pipe.png" class="pipe">
    </div>
</body>
```

Adicionando o CSS

No arquivo CSS é onde definiremos a posição e movimentação das imagens que estão na página. Vamos começar definindo alguns atributos gerais, para isso usamos o símbolo de asterisco, ele faz com que o style seja aplicado a todos os seletores.

Logo em seguida, vamos configurar também a área do jogo, determinando seu tamanho e margem. Observe que há um elemento que ainda não havíamos estudado, o “overflow: hidden”. O Overflow determina que quando um objeto

ultrapassar o limite da borda da nossa Div, ele será dividido, nesse caso, quando passar da borda a parte do objeto para fora ficará “hidden”, isto é, escondida, podemos dizer que se tornará transparente. Desta maneira, quando o cano ou as nuvens passarem pela borda, desaparecerão coluna por coluna de pixels conforme se movimentarem.

```css
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

.game-board {
    width: 100%;
    height: 100vh;
    border-bottom: 16px solid rgb(28, 119, 10);
    margin: 0 auto;
    position: relative;
    overflow: hidden;
    background: linear-gradient(#46beee, #e0f6ff);
}
```

A configuração do cano já introduz elementos de animação. Para que funcione corretamente, precisamos definir a posição como absoluta, isto é, ao invés de se alinhar de acordo com o documento inteiro, ele se vai se alinhar com seu objeto pai (nesse caso a div game-board), podendo até sobrepor outros objetos. Por isso é importante que no .game-board tenha o atributo “position: relative”.

Vamos então criar a Animação do cano. Ele vai se movimentar da direita para a esquerda. A sequência dos frames da animação é definida na imagem seguinte. Insira o nome da animação no atributo Animation e defina o tempo de execução. “Infinite” determina que animação será infinita e “linear” que os frames passarão de forma linear, sem haver aceleração.

```css
.pipe {
    position: absolute;
    bottom: 0;
    width: 80px;
    animation: pipe-animation 1s infinite linear;
    height: 112px;
}
```

Note que o “from” inicia em um valor negativo, usamos assim para que o cano não surja de repente no início da tela. teste com o valor em 0 para observar o que ocorre.

```css
@keyframes pipe-animation {
    from {
        right: -80px;
    }
    to {
        right: 100%
    }
}
```

Agora, vamos configurar nosso protagonista. Assim como o cano, ele terá posição absoluta, e definimos o bottom como zero, para que fique colado ao chão.

```css
.mario {
    position: absolute;
    bottom: 0;
    width: 150px;
}
```

A animação do pulo também precisa ser configurada, para isso, inserimos o nome da animação, tempo de duração. “Ease-out” é um tipo de transição de animação do CSS, que define uma animação mais abrupta no início e mais lenta no final.

```css
.jump {
    animation: jump 500ms infinite ease-out;
}
```

```css
@keyframes jump {
    0% {
        bottom: 0;
    }
    40% {
        bottom: 180px;
    }
    50% {
        bottom: 180px;
    }
    60% {
        bottom: 180px;
    }
    100% {
        bottom: 0;
    }
}
```

Para a animação das nuvens, faremos da mesma maneira: a posição precisa ser absoluta, e configura-se os frames da direita para a esquerda, com duração de 20 segundos, de modo infinito e velocidade linear.

```css
.clouds {
    position: absolute;
    width: 550px;
    animation: clouds-animation 20s infinite linear;
}
```

```css
@keyframes clouds-animation {
    from {
        right: -550px;
    }
    to {
        right: 100%
    }
}
```

Configurando o JavaScript

No arquivo de javascript, vamos utilizar inicialmente 2 variáveis constantes. O valor atribuído a elas, são as classes que definimos no nosso CSS. Para que um arquivo JS tenha acesso a um elemento CSS, utilizamos “document.querySelector”.

```js
const mario = document.querySelector('.mario');
const pipe = document.querySelector('.pipe');
```

Para que o pulo do personagem funcione, é necessário adicionar e remover a classe “jump” toda vez que formos pular.

```js
const jump = () => {
    mario.classList.add('jump');

    setTimeout(()=> {mario.classList.remove('jump');}, 500);
}
```

A próxima função, vai verificar se houve colisão entre o personagem e o cano. Para que isso seja possível, é necessário verificar a posição do cano, então cria-se uma variável “pipePosition” onde seu valor é a posição em relação ao eixo esquerdo da Div.

Também pegamos com uma variável a posição vertical (bottom) do personagem, para determinar se ele passou por cima do cano ou não.

Com essas duas variáveis, utilizaremos uma estrutura condicional If, para testar se houve colisão ou não. Na condição podemos ler: “Se a posição do cano em relação ao eixo esquerdo for menor ou igual a 120px E maior do que zero E a posição vertical do mario em relação ao chão for menor do que 112px”.

Caso a condição esteja correta, executa-se a sequência de códigos que exibirão a imagem de game-over e configuraram para que o mario pare de pular e o cano de se movimentar.

```
const loop = setInterval(() => {
    const pipePosition = pipe.offsetLeft;
    const marioPosition = +window.getComputedStyle(mario).bottom.replace('px', '');

    if (pipePosition <= 120 && pipePosition > 0 && marioPosition < 112) {
        pipe.style.animation = 'none';
        pipe.style.left = `${pipePosition}px`;

        mario.style.animation = 'none';
        mario.style.bottom = `${marioPosition}px`;

        mario.src = 'img/game-over.png';
        mario.style.width = '75px';
        mario.style.marginLeft = '50px';

        clearInterval(loop); //para o loop
    }
}, 10)
```

Por fim, adicionamos um Listener, que detectar quando a tecla for pressionada.

```
document.addEventListener('keydown', jump);
```

Tarefa de Casa

Escolha 1 site interativo e inspecione a página. Seu objetivo é procurar e identificar estruturas HTML, animações e estilos CSS, funções em JavaScript, etc, que já tenhamos utilizado em aula. Tire print e explique o trecho identificado. Apresente no mínimo 3 linhas de código para explicar (podem ser se partes diferentes do código).