**Anotações Projeto Jogo Dinossauro - Celso Henrique**

Dentro do meu **head** tenho algumas tags:

<script src="script.js" charset="UTF-8" defer></script>

**charset UTF-8** é para caso o script tenha caracteres especiais e dessa forma não dê erro, e eles sejam interpretados.

A tag **defer** é pra dizer que o arquivo *script.js* será executado depois que toda a pagina tiver sido carregada pelo navegador

Quero deixar o dino embaixo, vamos usar position: absolute; com bottom: 0 => dessa forma ele vai ficar embaixo, independente do que tiver em volta dele.

**No meu arquivo JS:**

Criando a constante dino para manipular o dino, vamos chamar um método do documento, (o browser disponibiliza uma API são comando para chamar os elementos na tela, manipular, etc.) E no documento (que é a pagina web q estou construindo) ele vai buscar o elemento **dino**. Veja:

**const dino = document.querySelector('.dino');**

Feito isso selecionamos o dino colocando ele dentro da constante dino para conseguir manipular dentro do javascript. O query selector está buscando a classe toda dino, ou seja, ele está retornando a div e tudo mais que tem o dino:

<div class="dino"></div>

Uma das funcionalidades do nosso jogo é fazer o dino pular, pra isso precisamos interceptar o pressionamento de teclas, nesse caso, apertando o botão espaço, vou conseguir fazer ele pular. Vamos usar o **eventListener** pra isso:

document.addEventListener('keyup', function() {

});

**keyup** é quando pressionamos uma tecla e ele gera um evento de keydown e quando soltamos a tecla ele gera um keyup.

Agora vamos colocar ele pra verificar se o usuário apertou a tecla espaço.

Vamos criar uma function **handleKeyUp** com o argumento **event**. O event é enviado para a função toda vez que alguém pressiona uma tecla pelo navegador. Com ele será possível identificar qual a tecla que a pessoa está pressionando

Vamos fazer uma condicional pra isso:

function handleKeyUp(event) {

if (event.keyCode === 32) {

}

}

document.addEventListener('keyup', handleKeyUp);

o keycode 32 é para a tecla espaço!

No site keycode.info vc consegue checar essas informações.

Pronto, já interceptamos o evento de apertar a tecla espaço, agora precisamos fazer o dinossauro pular. Vamos criar a function **jump:**

Vamos criar uma variável **position** com valor 0 usando **let** pois vamos mudar seu valor futuramente. Será a posição inicial do dinossauro.

Para fazer a animação (o movimento) vamos criar outra variável **upInterval**, que será responsável por criar uma repetição para levar o dinossauro para cima usando uma função do javascript **setInterval()** que serve para definir intervalos, e então todo código que estiver dentro dela vai ser executado sem parar no intervalo que for definido. No caso colocamos o intervalo **20**, ou seja, a cada 20 milesegundos será executado o código. Se alterarmos esse intervalo de 20 que colocamos pra menos ou para mais, modificará a velocidade de subida ou descida do dino.

Vamos alterar a posição inicial tb para 20, mas isso ainda não fará o dino pular:

function jump() {

let position = 0;

let upInterval = setInterval(() => {

position += 20;

}, 20);

}

Para fazer o dino pular precisamos aplicar estilos no seu CSS. Vamos pegar a propriedade bottom que colocamos no css e alterar ela:

let upInterval = setInterval(() => {

position += 20;

dino.style.bottom = position + 'px';

}, 20);

Precisamos desse ‘px’ depois para o CSS entender que é 20 px, senão ele não le o valor corretamente.

Agora vamos chamar a função **jump()** no evento que vê se a pessoa pressionou espaço (dentro da function **handleKeyUp**). O código todo ficará assim:

const dino = document.querySelector('.dino');

function handleKeyUp(event) {

if (event.keyCode === 32) {

jump();

}

}

function jump() {

let position = 0;

let upInterval = setInterval(() => {

position += 20;

dino.style.bottom = position + 'px';

}, 20);

}

document.addEventListener('keyup', handleKeyUp);

Com isso tudo que fizemos, já conseguimos fazer o dino pular, mas ele “voa” e não volta.   
Precisamos fazer ele subir até um máximo definido e retornar para a posição 0.

Para isso vamos adicionar uma condicional dentro da variavel **upInterval:**

Subindo até 150:

let upInterval = setInterval(() => {

if (position >= 150) {

clearInterval(upInterval);

}

position += 20;

dino.style.bottom = position + 'px';

}, 20);

Dessa forma o dino sobe somente até 150 e para.

Agora precisamos fazer ele descer (vamos limpar o intervalo com **clearInterval**)::

if (position >= 150) {

clearInterval(upInterval);

//descendo

let downInterval = setInterval(() => {

position -= 20;

dino.style.bottom = position + 'px';

Mas com isso, o dino está descendo demais. Precisamos limpar o intervalo de descida, fazer com que ele pare de acontecer quando for 0.

Vamos usar novamente a função **clearInterval** do Javascript, ela vai zerar o intervalo que eu quiser. Antes zeramos (limpamos) o upInterval, agora será o downInterval que criamos:

//descendo

let downInterval = setInterval(() => {

if (position <= 0) {

clearInterval(downInterval);

} else {

position -= 20;

dino.style.bottom = position + 'px';

}

}, 20);

Prontinho! Nosso dino já está pulando e o código todo ficou assim:

const dino = document.querySelector('.dino');

function handleKeyUp(event) {

if (event.keyCode === 32) {

jump();

}

}

function jump() {

let position = 0;

let upInterval = setInterval(() => {

if (position >= 150) {

clearInterval(upInterval);

//descendo

let downInterval = setInterval(() => {

if (position <= 0) {

clearInterval(downInterval);

} else {

position -= 20;

dino.style.bottom = position + 'px';

}

}, 20);

} else {

//subindo

position += 20;

dino.style.bottom = position + 'px';

}

}, 20);

}

document.addEventListener('keyup', handleKeyUp);

Se observarmos bem, vamos ver que o dino está travando quando aperto muitas vezes o espaço.

Vamos criar a variável **isJumping** para saber se ele tá pulando ou não, para acabar com essa bugada.

Vamos colocar ela como uma condicional dentro da nossa função **handleKeyUp** e também colocar ela como ativada (true) dentro da função **jump no inicio!**. Assim saberemos se ele está pulando. E vamos colocar como false dentro de **downInterval** para mostrar que não está mais pulando:

const dino = document.querySelector('.dino');

let isJumping = false;

function handleKeyUp(event) {

if (event.keyCode === 32) {

if (!isJumping) {

jump();

}

}

}

function jump() {

let position = 0;

isJumping = true;

let upInterval = setInterval(() => {

if (position >= 150) {

clearInterval(upInterval);

//descendo

let downInterval = setInterval(() => {

if (position <= 0) {

clearInterval(downInterval);

isJumping = false;

} else {

position -= 20;

dino.style.bottom = position + 'px';

}

}, 20);

} else {

//subindo

position += 20;

dino.style.bottom = position + 'px';

}

}, 20);

}

document.addEventListener('keyup', handleKeyUp);

Agora vamos criar os cactus!

Para isso vamos criar uma função **createCactus.** Nela vamos chamar uma função do javascript **document.createElement()**, para criar nossa div com o cactus, assim como fizemos para o dino, só que agora estamos usando javascritp e não html.

function createCactus() {

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

}

Agora vamos adicionar uma classe para o cactus, para depois conseguirmos fazer o CSS dele, colocando estilos:

function createCactus() {

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

cactus.classList.add('cactus');

}

Agora quero inserir o cactus dentro da div background que criamos lá no html no início da aula. Vamos escrever na segunda linha do código:

const background = document.querySelector('.background');

Agora vamos adicionar um filho ao background para colocar ele dentro da nossa div background, usando **appendChild(cactus)**:

function createCactus() {

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

cactus.classList.add('cactus');

background.appendChild(cactus);

}

Agora vamos posicionar nosso cactus à direita:

let cactusPosition = 1000;

cactus.style.left = 1000 + 'px';

Vai ficar assim o código para criação do cactus::

function createCactus() {

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

let cactusPosition = 1000;

cactus.classList.add('cactus');

cactus.style.left = 1000 + 'px';

background.appendChild(cactus);

}

Agora quero que imediatamente na hora que o jogo começar o cactus seja criado, para isso vou chamar a função **createCactus().**

function createCactus() {

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

let cactusPosition = 1000;

cactus.classList.add('cactus');

cactus.style.left = 1000 + 'px';

background.appendChild(cactus);

}

createCactus();

document.addEventListener('keyup', handleKeyUp);

Vamos estilizar o cactus para ele aparecer na pag da web:

.cactus {

position: absolute;

width: 60px;

height: 60px;

bottom: 0;

background: red;

}

Precisamos fazer o cactus se movimentar agora:

Vou criar uma variavel **leftInterval** para o cactus ir para a esquerda usando novamente **setInterval:**

let leftInterval = setInterval(() => {

cactusPosition -= 10;

cactus.style.left = cactusPosition + 'px';

}, 20)

No cód acima podemos definir a velocidade que ele irá se mover para a esquerda. Se eu colocar 1 ele vai super lento, se colocar 20 vai rapido demais.

Pode ficar aqui como desafio, fazer essa velocidade subir à medida que o jogador fica mais tempo jogando!

Agora já temos um cactus se movendo pela tela e indo embora da direita para a esquerda! MAS precisamos fazer vários cactus irem surgindo na tela!

Primeiro vamos fazer o cactus desaparecer, conferindo se ele saiu totalmente da tela (verificando se sua posição é - 60: **cactusPosition < -60**), (colocamos 60 pois é o seu tamanho! Ele tem 60px por 60px) e vamos usar também o **clearInterval.**

Vamos também precisar usar o **removeChild** para remover o elemento filho do background que é o cactus:

if (cactusPosition < -60) {

clearInterval(leftInterval);

background.removeChild(cactus);

}

Se nao fizermos isso o cactus ia ficar andando pela tela sem parar. Mesmo não vendo ele na nossa tela, ele estaria ali num loop, passando por um processamento desnecessário.

O código ficará assim (se a posição for menor que -60 ele elimina o cactus, caso contrário ele continua se movimentando):

let leftInterval = setInterval(() => {

if (cactusPosition < -60) {

clearInterval(leftInterval);

background.removeChild(cactus);

} else {

cactusPosition -= 10;

cactus.style.left = cactusPosition + 'px';

}

}, 20)

Agora vamos criar um novo cactus.

Eles aparecem de forma aleatória, as vezes vem um só, as vezes tres, etc…

Para isso vamos criar uma variavel chamada **randomTime** e usar nela um método chamado **Math** que é usado para fazer operações matemáticas etc. No caso vamos usa-la para gerar numeros aleatorios atraves do **Math.random**:

function createCactus() {

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

let cactusPosition = 1000;

let randomTime = Math.random() \* 6000;

Vamos multiplicar por 6000 (não entendi pq)

Agora com esses numeros aleatorios vamos gerar novos cactus!

Vamos usar **setTimeout** para executar uma determinada função (sendo este seu primeiro argumento) e depois vamos determinar o tempo de execução dessa função (sendo este seu segundo argumento):

setTimeout(createCactus, randomTime);

O Código ficou assim (para a criação dos cactus):

function createCactus() {

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

let cactusPosition = 1000;

let randomTime = Math.random() \* 6000;

cactus.classList.add('cactus');

cactus.style.left = 1000 + 'px';

background.appendChild(cactus);

let leftInterval = setInterval(() => {

if (cactusPosition < -60) {

clearInterval(leftInterval);

background.removeChild(cactus);

} else {

cactusPosition -= 10;

cactus.style.left = cactusPosition + 'px';

}

}, 20);

setTimeout(createCactus, randomTime);

}

É meio estranho invocar **createCactus** (com o setTimeout) dentro da própria função **createCactus**. Isso se chama recursividade, quando chamamos uma função dentro dela mesma. É como se fosse um efeito do espelho, um espelho olha pro outro e vai gerando imagens e assim por diante.

No nosso caso, estamos invocando a criação de um novo cactus de novo, de maneira infinita, num loop. Estamos usando a recursividade para ir gerando nossos cactus.

Agora vamos colocar no código uma forma de mostrar que o dino entrou em contato com o cactus, fazendo com que tudo pare, dando game over.

Vamos colocar mais uma condicional dentro de **leftInterval** verificando se o cactus ainda está na tela (ou seja, cactusPosition > 0) e está no mesmo espaço que o dino ou seja cactusPosition < 60, (60 é o tamanho do dino então se o cactus está entre 0 e 60 significa q ele está na mesma posição que ele).

else if {cactusPosition > 0 && cactusPosition < 60)

A ação para essa condicional acima é dar game over, fazer com que o cactus pare de ir pra esquerda, para isso vamos limpar o intervalo da esquerda (leftInterval):

else if (cactusPosition > 0 && cactusPosition < 60) {

clearInterval(leftInterval);

}

Agora vou acessar a tag **body** do meu html e colocar um h1 para mostrar o texto “fim de jogo”. Então, agora quando o cactus encosta no dino ele limpa a tela toda,ela fica em branco e mostra o texto de fim de jogo:

else if (cactusPosition > 0 && cactusPosition < 60) {

clearInterval(leftInterval);

document.body.innerHTML = '<h1 class="game-over">Fim de jogo</h1>';

}

Mas agora temos um problema, pois mesmo pulando e fugindo do cactus ele da fim de jogo.

Então precisamos calcular se o dino está pulando ou não.

Através da variavel **position** sabemos onde ele está, então vamos tirar ela de dentro da function jump e colocar ela externa para se tornar acessível (isso se chama closure em javascript, as variaveis ficam disponiveis somente dentro de onde elas foram criadas, entao criando fora ela fica global).

O inicio do codigo ficou assim agora:

const dino = document.querySelector('.dino');

const background = document.querySelector('.background');

let isJumping = false;

let position = 0;

Agora vamos acessar a posição do pulo dentro de **leftInterval.**

Se o dino não estiver pulando então a position dele é menor que 60 (que é seu tamanho). Se ele estiver com mais de 60 ele está pulando:

else if (cactusPosition > 0 && cactusPosition < 60 && position < 60)

Nossa **leftInterval** ficou assim:

let leftInterval = setInterval(() => {

if (cactusPosition < -60) {

clearInterval(leftInterval);

background.removeChild(cactus);

} else if (cactusPosition > 0 && cactusPosition < 60 && position < 60) {

//Game over

clearInterval(leftInterval);

document.body.innerHTML = '<h1 class="game-over">Fim de jogo</h1>';

} else {

cactusPosition -= 10;

cactus.style.left = cactusPosition + 'px';

}

}, 20);

O jogo está pronto e seu código completo é esse:

const dino = document.querySelector('.dino');

const background = document.querySelector('.background');

let isJumping = false;

let position = 0;

function handleKeyUp(event) {

if (event.keyCode === 32) {

if (!isJumping) {

jump();

}

}

}

function jump() {

isJumping = true;

let upInterval = setInterval(() => {

if (position >= 150) {

clearInterval(upInterval);

//descendo

let downInterval = setInterval(() => {

if (position <= 0) {

clearInterval(downInterval);

isJumping = false;

} else {

position -= 20;

dino.style.bottom = position + 'px';

}

}, 20);

} else {

//subindo

position += 20;

dino.style.bottom = position + 'px';

}

}, 20);

}

function createCactus() {

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

let cactusPosition = 1000;

let randomTime = Math.random() \* 6000;

cactus.classList.add('cactus');

cactus.style.left = 1000 + 'px';

background.appendChild(cactus);

let leftInterval = setInterval(() => {

if (cactusPosition < -60) {

clearInterval(leftInterval);

background.removeChild(cactus);

} else if (cactusPosition > 0 && cactusPosition < 60 && position < 60) {

//Game over

clearInterval(leftInterval);

document.body.innerHTML = '<h1 class="game-over">Fim de jogo</h1>';

} else {

cactusPosition -= 10;

cactus.style.left = cactusPosition + 'px';

}

}, 20);

setTimeout(createCactus, randomTime);

}

createCactus();

document.addEventListener('keyup', handleKeyUp);

Agora vamos estilizar nosso jogo:

Colocar um cinza claro no background do jogo (#fafafa):

body {

background: #fafafa;

}

Colocando a imagem do dino:

background-image: url(dino.png);

Agora do cactus:

background-image: url(cactus.png);

Vamos fazer um deserto em movimento, usando animação em CSS com @keyframes.

Vou chamar de slideright e colocar sua posição inicial como 70000% (bem na esquerda).

indo para zero:

@keyframes slideright {

from {

background-position: 70000%;

}

to {

background-position: 0;

}

}

Isso faz com que o background deslize da esquerda para a direita de forma animada. Só falta chamar o keyframes na estilização do background.

Vamos estilizar a div com a classe background inserindo a imagem do deserto e estilizando outras coisas.

Vamos fazer ela se repetir de maneira infinita com **background-repeat: repeat-x;** assim ele vai estender por toda a tela no eixo x, independente do tamanho da tela.

Chamo o keyframe com a propriedade **animation** e coloco que ele vai deslizar para a direita com a velocidade de 600s (segundos) de maneira infinita, e linear que define o estilo do tempo da minha variação da animação, sendo linear ela nunca muda, mas tem outras propriedades que faz ficar mais rápido no meio e depois devagar, etc.

.background {

position: absolute;

bottom: 0;

background-image: url(background.png);

background-repeat: repeat-x;

animation: slideright 600s infinite linear;

width: 100%;

height: 200px;

}