

EFEITOS NO DOM

`setTimeout` e `setInterval`

setTimeout()

`setTimeout(callback, tempo, arg1, arg2, ...)` método assíncrono que ativa o callback após `tempo`. Não existe garantia de que o código será executado exatamente após o tempo, pois o callback entra na fila e espera pela Call Stack estar vazia.

```
function espera(texto) {  
  console.log(texto);  
}  
setTimeout(espera, 1000, 'Depois de 1s');
```

Imediato

Se não passarmos o argumento de tempo ele irá assumir o valor 0 e entrará na **fila** imediatamente para ser executado. Podemos passar uma função anônima diretamente com argumento.

```
setTimeout(() => {  
  console.log('Após 0s?');  
});
```

Exemplo de setTimeout

Loops e setTimeout

Um loop é executado rapidamente, em milissegundos. Se colocarmos um setTimeout dentro do loop, todos eles serão adicionados à Web Api praticamente no mesmo tempo. Um evento de setTimeout não espera o tempo do anterior acabar para iniciar.

```
for(let i = 0; i < 20; i++) {  
  setTimeout(() => {  
    console.log(i);  
  }, 300);  
}
```

Corrigindo o Loop

Agora ele está multiplicando o tempo por i. Assim o primeiro aparecerá em 0ms, o segundo em 300ms, o terceiro em 600ms e assim em diante.

```
for(let i = 0; i < 20; i++) {  
  setTimeout(() => {  
    console.log(i);  
  }, 300 * i);  
}
```

This e Window

setTimeout é um método do objeto Window. O valor de `this` dentro do mesmo callback é uma referência ao seu objeto no caso Window.

```
const btn = document.querySelector('button');
btn.addEventListener('click', handleClick);

function handleClick(event) {
  setTimeout(function() {
    this.classList.add('active');
  }, 1000)
}

// Erro pois window.classList não existe
```

Arrow Function

Quando utilizamos uma Arrow Function como callback, o contexto de `this` passa a ser do local onde o `setTimeout` foi iniciado.

```
const btn = document.querySelector('button');
btn.addEventListener('click', handleClick);

// this agora é btn.
function handleClick(event) {
  setTimeout(() => {
    this.classList.add('active');
  }, 1000)
}
```


setInterval

`setInterval(callback, tempo, arg1, arg2, ...)`, irá ativar o callback toda vez que a quantidade de tempo passar.

```
function loop(texto) {  
  console.log(texto);  
}  
setInterval(loop, 1000, 'Passou 1s');  
  
// loop a cada segundo  
let i = 0;  
setInterval(() => {  
  console.log(i++);  
}, 1000);
```

clearInterval

`clearInterval(var)`, podemos parar um intervalo com o `clearInterval`. Para isso precisamos atribuir o `setInterval` a uma variável.

```
const contarAte10 = setInterval(callback, 1000);

let i = 0;
function callback() {
  console.log(i++);
  if (i > 10) {
    clearInterval(contarAte10);
  }
}
```

Exercícios

```
// Mude a cor da tela para azul e depois para vermelho a cada 2s.
```

```
// Crie um cronometro utilizando o setInterval. Deve ser possível
```

```
// iniciar, pausar e resetar (duplo clique no pausar).
```