06

Parando o Timer

Transcrição

No vídeo anterior, vimos que ao tentar parar o *timer*, além dele não parar, o tempo fica acelerado após cada clique que damos no botão. Para revolvermos esse problema, primeiramente devemos entender o motivo disso estar acontecendo.

No **renderer.js**, toda vez que clicamos no botão de *play*, executamos a função `iniciar` do *timer*. Por sua vez, a função `iniciar` executa a função `setInterval`, que a cada 1 segundo, soma 1 segundo no *timer*. Logo, toda vez que clicamos no botão, uma função `setInterval` é executada, adicionando mais 1 segundo ao *timer* a cada segundo. É como se a cada vez que o botão for clicado, um *timer* é executado por debaixo dos panos, e para cada *timer* desses, 1 segundo é adicionado no nosso tempo. Por exemplo, se clicarmos 10 vezes no botão, a cada segundo serão adicionados 10 segundos no *timer* da nossa aplicação.

Não é isso que queremos, queremos que só um `setInterval` seja executado. Para isso, assim que clicarmos no botão de *play*, vamos limpar o *timer* anterior, começando um novo logo em seguida.

Limpando o setInterval antes de começar um novo

Toda vez que criarmos um novo `setInterval`, a função nos retorna um `id`. Então, o que podemos fazer é guardar esse `id`, e com ele em mãos, toda vez que clicarmos no botão, nós deletamos o `setInterval` anterior, que é identificado pelo `id`. Assim, nunca haverá mais de um `setInterval` executando.

Para isso, no **timer.js**, vamos criar a variável `timer`, fora do módulo. E vamos atribuir o `setInterval` a essa variável:

```
// timer.js

const moment = require('moment');
let segundos;
let timer;

module.exports = {
    iniciar(el) {
        let tempo = moment.duration(el.textContent);
        segundos = tempo.asSeconds();
        timer = setInterval(() => {
            segundos++;
            el.textContent = this.segundosParaTempo(segundos);
        }, 1000);
    }, parar() {

    }, segundosParaTempo(segundos) {
        return moment().startOf('day').seconds(segundos).format("HH:mm:ss");
    }
}
```

Até agora, nada mudou, o comportamento ainda é o mesmo. Precisamos remover o `setInterval` antes de executar um novo, para isso vamos utilizar a função `clearInterval` passando o `id` para ela:

```
// timer.js

const moment = require('moment');
let segundos;
let timer;

module.exports = {
    iniciar(el) {
        let tempo = moment.duration(el.textContent);
        segundos = tempo.asSeconds();
        // removendo o setInterval anterior
        clearInterval(timer);
        timer = setInterval(() => {
            segundos++;
            el.textContent = this.segundosParaTempo(segundos);
        }, 1000);
    }, parar() {

    }, segundosParaTempo(segundos) {
        return moment().startOf('day').seconds(segundos).format("HH:mm:ss");
    }
}
```

Ao testar a nossa aplicação, vemos que o bug do tempo ficar acelerado foi corrigido, já que agora somente um `setInterval` é executado.

Parando o timer

Agora que corrigimos o bug, falta implementar a função de parar, que é muito simples, basta limparmos o `setInterval` , que já vimos como fazer nesse vídeo, chamando a função `clearInterval` :

```
// timer.js

const moment = require('moment');
let segundos;
let timer;

module.exports = {
    iniciar(el) {
        let tempo = moment.duration(el.textContent);
        segundos = tempo.asSeconds();
        // removendo o setInterval anterior
        clearInterval(timer);
        timer = setInterval(() => {
            segundos++;
            el.textContent = this.segundosParaTempo(segundos);
        }, 1000);
    }, parar() {
        clearInterval(timer);
    }, segundosParaTempo(segundos) {
        return moment().startOf('day').seconds(segundos).format("HH:mm:ss");
    }
}
```

Resta agora chamar essa função no **renderer.js**. Atualmente, estamos sempre chamando a função `iniciar`, mas queremos que quando o botão for clicado pela primeira vez, devemos chamar a função `iniciar`, se o botão for clicado novamente, devemos chamar a função `parar`. Se o botão for clicado mais uma vez, a função `iniciar` deve ser chamada, e assim sucessivamente, tendo uma alternância nas chamadas das funções.

Para isso, vamos criar uma variável, chamada `play`. Como o *timer* começa parado, ela será inicializada com o valor `false`. Dentro do evento, vamos fazer um `if`, se a variável for verdadeira, nós vamos executar a função `parar` e atribuir o valor `false` para a variável. Mas se o valor da variável for falso, nós vamos executar a função `iniciar`, atribuindo o valor `true` para a variável:

```
// renderer.js

// restante do código comentado

let play = false;
botaoPlay.addEventListener('click', function () {
    if(play) {
        timer.parar();
        play = false;
    } else {
        timer.iniciar(tempo);
        play = true;
    }
    imgs = imgs.reverse();
    botaoPlay.src = imgs[0];
});
```

Ao testar a aplicação e iniciar o *timer*, ao clicar no botão de *stop*, vemos que o tempo é parado corretamente!