11

Mãos na Massa: Implementando nosso timer

Vamos agora começar implementando nosso módulo de `timer.js`, que será o responsável por controlar o tempo em nossa aplicação.

1- Crie o arquivo `timer.js` dentro da pasta `js/`, junto ao `renderer.js`. Lá dentro, adicione:

```
//timer.js

module.exports = {

}
```

2- No `timer.js`, vamos começar implementando nossas funções. A primeira que vamos recriar é a funçao de iniciar o timer, ou seja a função `iniciar()`. Mas antes disto, como vimos no vídeo, utilizaremos uma biblioteca externa para nos ajudar a trabalhar com tempo, a `moment.js`. Vamos começar instalando-a pelo nosso terminal:

```
npm install moment@2.17.1 --save
```

E importe ela no topo de nosso arquivo:

```
//timer.js
const moment = require('moment');
...
```

3- Com a moment instalada, podemos começar a implementar a função `iniciar`, que a cada segundo deve acrescer e alterar o tempo do timer. Crie a variável segundos, e comece a implementar a função:

```
//timer.js
const moment = require('moment');

let segundos;
let timer;
module.exports = {
    iniciar(el){
        let tempo = moment.duration(el.textContent);
        segundos = tempo.asSeconds();
        clearInterval(timer);
        timer = setInterval(()=>{
            segundos++;
            //Alterar view aqui
        }, 1000);
    }
}
```

4- Se queremos que a nosso tempo seja exibido conforme uma string "HH:mm:ss", de horas, minutos e segundos, precisamos criar uma função formatadora e não alterar a view diretamente com segundos. Vamos criar a função

`segundosParaTempo()`:

```
//timer.js
const moment = require('moment');

let segundos;
let timer;
module.exports = {
    iniciar(el){
        ...
    },
    segundosParaTempo(segundos){
        return moment().startOf('day').seconds(segundos).format("HH:mm:ss")
    }
}
```

5- E vamos utilizar nossa função formatadora para exibir o tempo corretamente a cada segundo que se passe na função `iniciar()`:

```
//timer.js
const moment = require('moment');

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

6- Agora vamos implementar nossa função de parar, de modo bem simples:

```
//timer.js
const moment = require('moment');

let segundos;
let timer;
module.exports = {
    iniciar(el){
        ...
    },parar(){
        clearInterval(timer);
    },segundosParaTempo(segundos){
```

```
        ...
    }
}
```

7- Por último, vamos voltar ao nosso `renderer.js` e fazer uso de nosso novo módulo de timer. Primeiro vamos importá-lo:

```
//renderer.js
const { ipcRenderer } = require('electron');
const timer = require('./timer');
```

8- Agora, dentro de nosso evento de click, vamos ficar alternando entre chamar as funções iniciar e parar, utilizando uma simples lógica com `ifs` :

```
//renderer.js
...
    let imgs = ['img/play-buttton.svg', 'img/stop-button.svg'];

    let tempo = document.querySelector('.tempo');
    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];
    });
...
```

Nosso timer deve estar com o tempo passando corretamente!