01

Lidando com erros

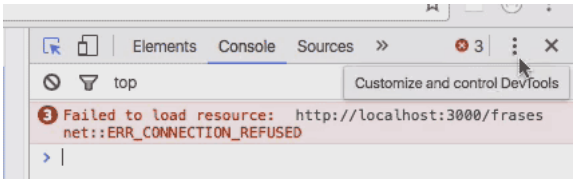
Transcrição

A nossa aplicação está funcionando e a próxima frase já está sendo carregada corretamente através de uma requisição AJAX.

Aqui pode vir uma dúvida: O que acontece se o servidor retorna um erro? Ou se a rede apresentasse algum problema na comunicação entre navegador e servidor? Em geral, como a nossa aplicação se comporta quando algo inesperado acontece?

Podemos simular um problema rapidamente para ver o comportamento, basta desligar o servidor e tentar carregar uma nova frase!

Ao testar podemos ver que a aplicação não mostra nenhum sinal ou mensagem ao usuário, ela fica passiva quando algum problema acontece. O erro aparece apenas no console do navegador:



No entanto o usuário (o jogador) não tem noção desse console pois ele é para o desenvolvedor. Temos que pensar como apresentar o erro de uma outra forma, mais amigável para o usuário!

Mensagem de erro

O primeiro passo é criar uma mensagem de erro dentro do HTML. Vamos adicionar um novo `div` logo abaixo do elemento `p` da frase:

```
<div class="center">
    <p id="erro">Ocorreu um erro, por favor tente novamente!</p>
</div>
```

E no nosso arquivo `estilos.css` vamos deixar o parágrafo vermelho e esconder a mensagem por padrão:

```
#erro{
    color: red;
    display: none;
}
```

Mensagem quando AJAX falha

A nossa mensagem deve aparecer quando a requisição AJAX realmente falha. Lembrando que executamos a requisição dentro do arquivo `frase.js` e é exatamente aí que vamos mexer agora. Mãos à obra!

Neste arquivo `frase.js`, na função `fraseAleatoria` vamos enfileirar a chamada da função `fail` à função `.get`. A `fail` recebe uma função anônima com o código que é executado quando um erro acontece:

```
function fraseAleatoria() {
    $.get("http://localhost:3000/frases2222", trocaFraseAleatoria) //URL errada para simular um
    .fail(function(){
        $("#erro").show(); //ao falhar mostra a mensagem de erro
    });
}
```

Ao testar a nossa aplicação com servidor desligado, a mensagem aparece pois a URL da função `.get` está errada. No entanto, a mensagem de erro fica visível para sempre - melhor seria se ela sumisse após de um determinado tempo.

Dentro da função `fail` vamos definir um *timeout* para chamar `hide` e esconder a mensagem de erro. Podemos ainda melhorar o código e, invés de chamar `show` e `hide`, usar a função `toggle`:

```
function fraseAleatoria() {
    $.get("http://localhost:3000/frases", trocaFraseAleatoria)
    .fail(function(){
        $("#erro").toggle();
        setTimeout(function(){
            $("#erro").toggle();
        },1500);
    });
}
```

Com a implementação da função `fail` garantimos que o usuário receba uma notificação em caso de erro, algo indispensável para qualquer aplicação mais robusta.

O que aprendemos?

- A não deixar o usuário no escuro e exibir os erros para ele.
- Como lidar quando o AJAX falha: a função .fail()
- Exibindo erros para o usuário.