PUC Minas

AJAX

Asynchronous JavaScript and XML

Prof. Rommel Carneiro

Javascript – Tópicos

- Introdução
- Modelo Clássico vs Modelo AJAX
- Vantagens e Desvantagens
- Exemplos práticos
  - API XMLHttpRequest
  - API Fetch
- CORS

Introdução – AJAX

Asynchronous Javascript and XML (AJAX) não é uma tecnologia, mas várias delas combinadas.

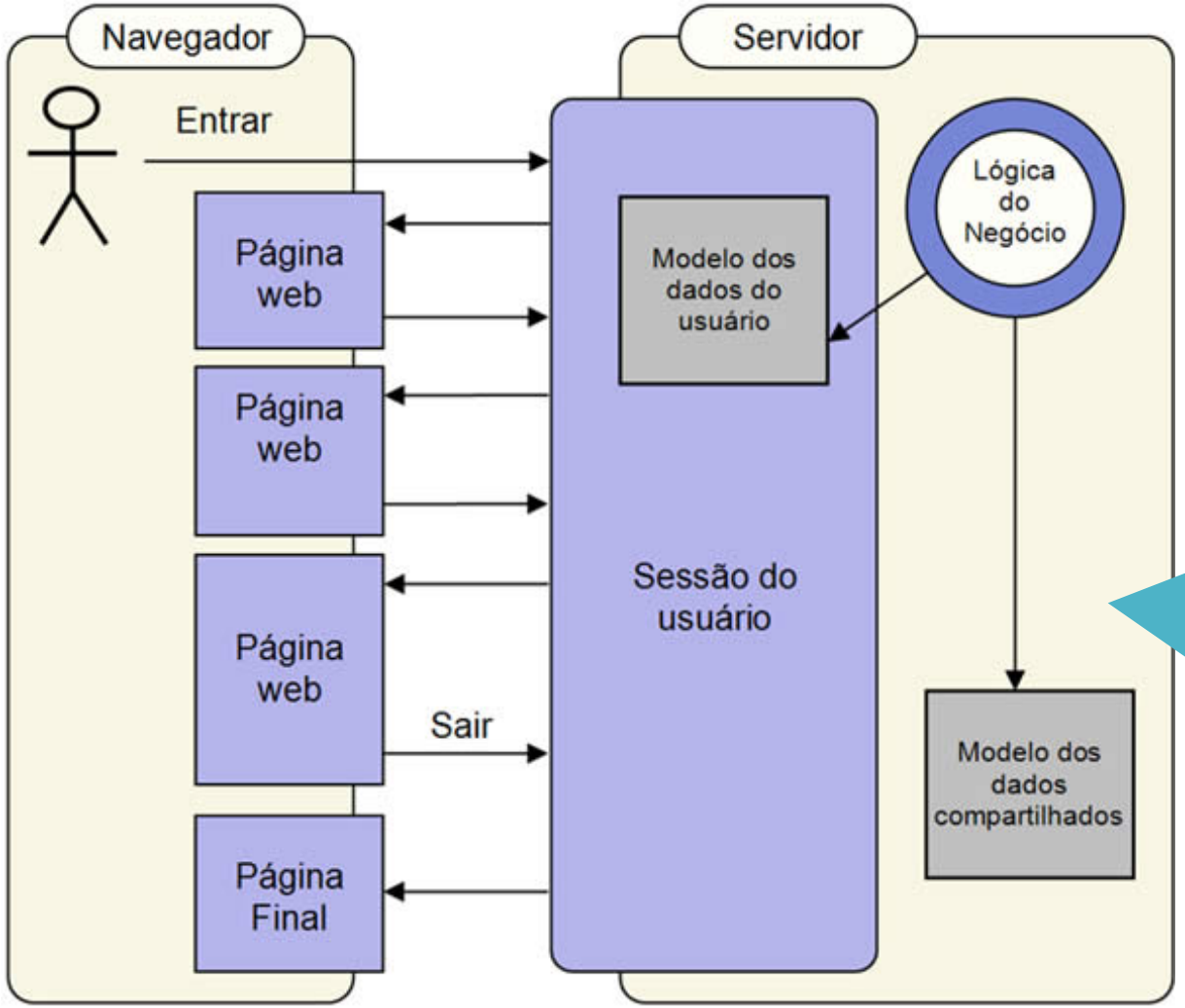
Recursos

- apresentação baseada em padrões como XHTML e CSS
- exibição dinâmica através do DOM
- troca e manipulação de dados em formato JSON, XML entre outros
- recuperação assincrônica de dados com **XMLHttpRequest**
- Javascript para juntar tudo isso

Introdução – AJAX – Exemplo – GMAIL

- Navegação entre pastas/tags
- Carga automática de mensagens
- Salvamento de mensagens durante a digitação
- Sugestão de contatos

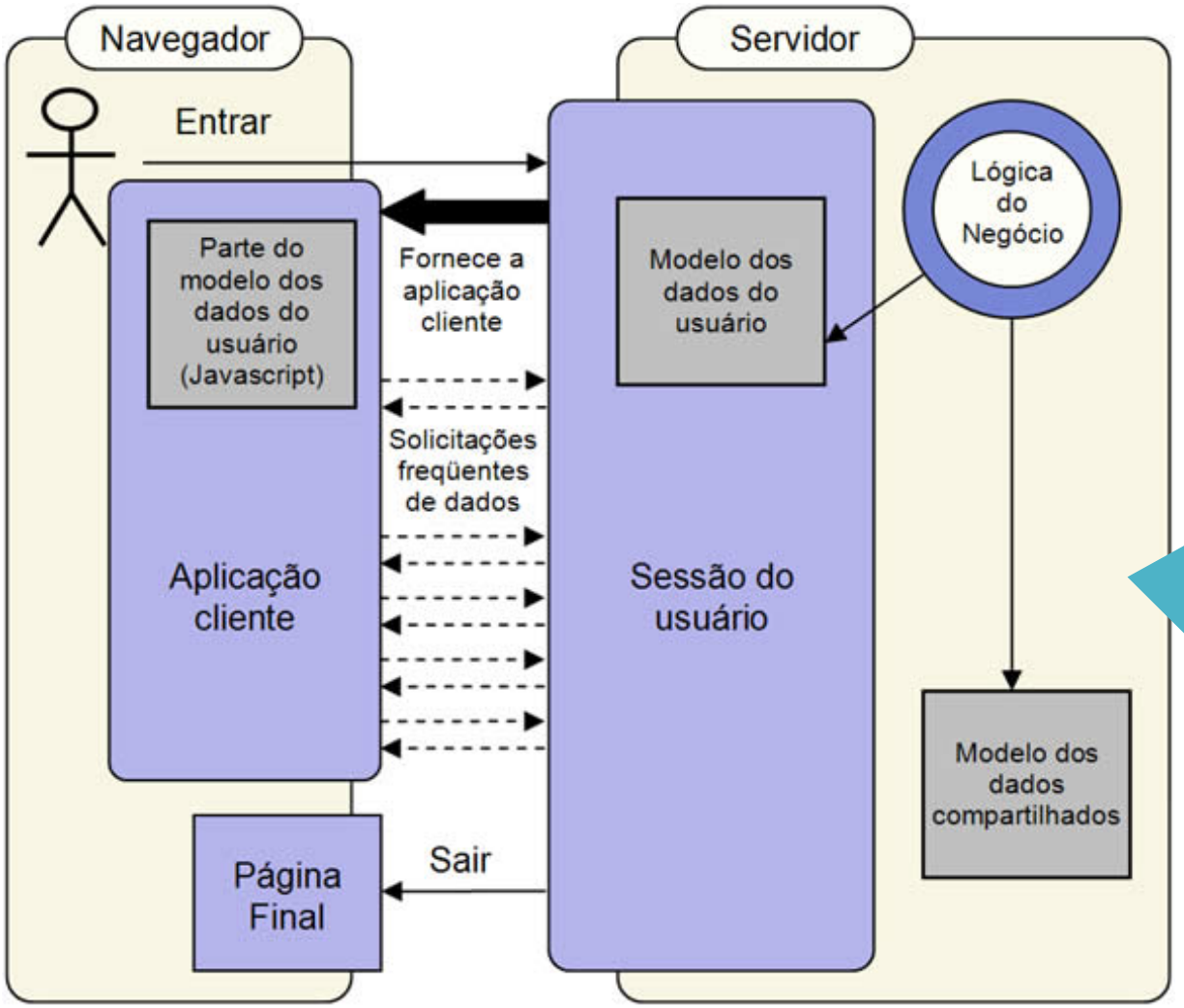
Modelo Clássico *vs* Modelo AJAX



Modelo Clássico

- Uma requisição → Uma página
- Lógica toda no servidor

Modelo Clássico *vs* Modelo AJAX



Modelo AJAX

- Diversas requisições para uma página
- Lógica dividida entre servidor e cliente

Vantagens e Desvantagens

Vantagens

- Melhoria significativa na experiência do usuário
- Redução do tráfego na rede
- Redução na carga do servidor web
- Flexibilidade de desenvolvimento do lado servidor

Desvantagens

- Maior complexidade no desenvolvimento de aplicações
- Exigência um equipamento melhor no lado cliente
- Requer compatibilidade do browser com padrões Javascript, HTML e CSS
- Possui tratamento diferenciado de um browser web para outro
- Tratamento complexo para uso das opções de avançar e voltar do browser

API XMLHttpRequest

- Criado pela Microsoft e depois adotado pela Mozilla, o XMLHttpRequest é um objeto ou API, fornecida pelo Browser, que permite que programas JavaScript troquem dados com servidores Web
- O objeto XMLHttpRequest é a base para a programação baseada em AJAX
- Apesar do nome, permite a troca de dados em outros formatos como JSON, HTML, TXT, entre outros, além de possibilitar a comunicação por meio de outros protocolos que não o HTTP.

API XMLHttpRequest – Requisição AJAX Exemplo

```
<!DOCTYPE html>
<html lang="en">
<head>
    <script>
        function success () { console.log(JSON.parse(this.responseText)); }
        function error (err) { console.log('Erro:', err); }

        var xhr = new XMLHttpRequest();
        xhr.onload = success;
        xhr.onerror = error;
        xhr.open('GET', 'https://api.github.com/users/rommelcarneiro');
        xhr.send();
    </script>
</head>
<body></body>
</html>
```

API XMLHttpRequest – Propriedades

Propriedade	Descrição
status	O código de status HTTP da resposta da solicitação.
statusText	O texto de status HTTP que vai com o código.
readyState	O staus do pedido.
responseText	Texto bruto da resposta.
responseXML	Resposta analisada em um objeto de documento DOM. Funciona apenas se o tipo de conteúdo é text/xml.
onreadystatechange	Disparo de evento chamado quando o readyState muda.
onerror	Disparo de evento que é chamado quando um erro acontece durante uma solicitação.
onprogress	Disparo de evento que é chamado em um intervalo, como o conteúdo é carregado.
onload	Disparo de evento que é chamado quando o documento for concluído.

API XMLHttpRequest – Fluxo ReadyState

Código de Status	Descrição
(0) UNINITIALIZED	O objeto foi criado mas não foi inicializado (O método **open** não foi disparado.)
(1) LOADING	O objeto foi criado, porém o método **send** não foi executado.
(2) LOADED	O método **send** foi executado, porém o status e os cabeçalhos ainda não estão disponíveis.
(3) INTERACTIVE	Alguns dados foram recebidos. Utilizar as propriedades **responseBody** e **responseText** neste estado para obter resultados parciais vai gerar um erro, uma vez que o status e os cabeçalhos ainda não estão completamente disponíveis.
(4) COMPLETED	Todos os dados foram recebidos e os dados completes estão disponíveis através das propriedades **responseBody** e **responseText**.

API Fetch – Requisição AJAX Exemplo

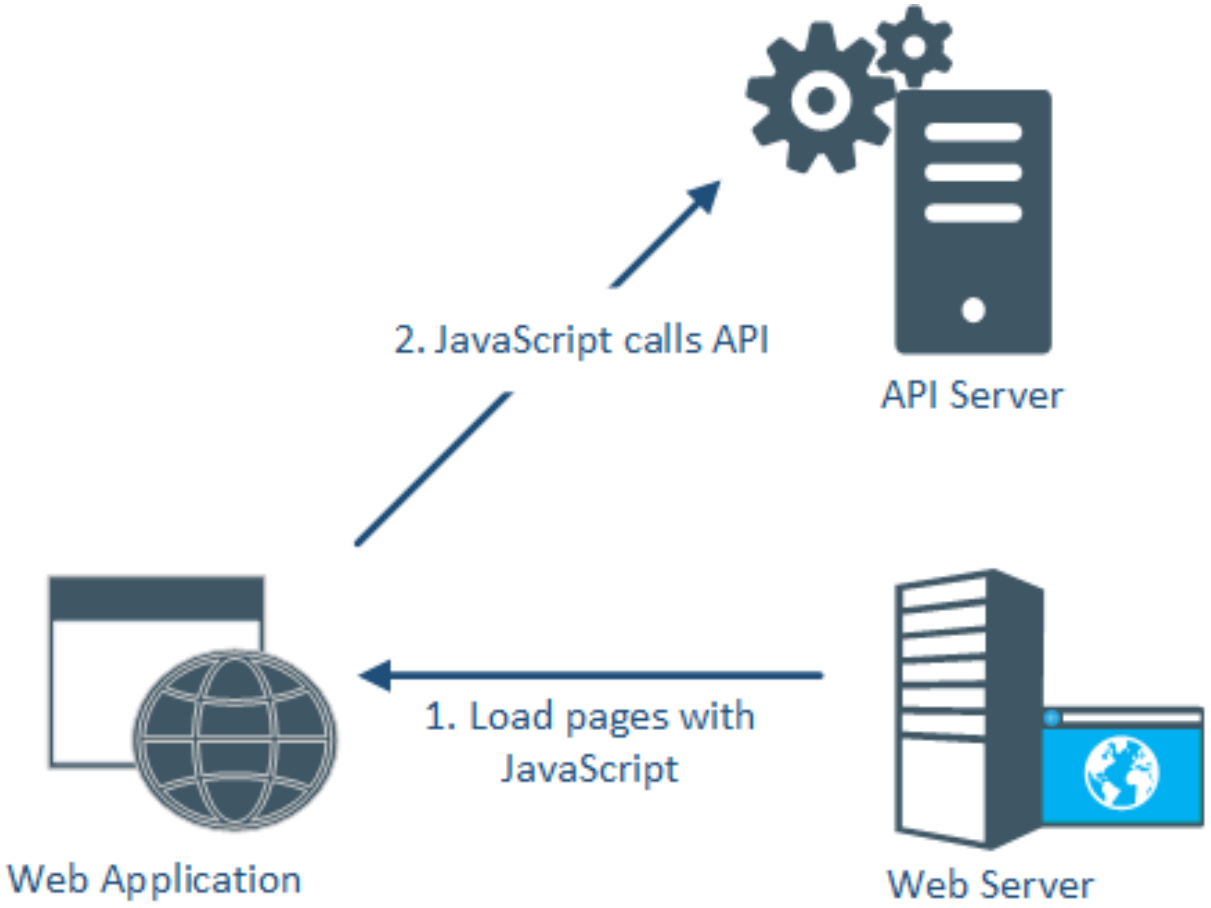
```
<!DOCTYPE html>
<html>

<head>
    <script>
        fetch('https://api.github.com/users/rommelcarneiro')
            .then(res => res.json())
            .then (data => console.log(data))
            .catch(err => console.error('Erro:', err))
    </script>
</head>

<body></body>

</html>
```

Cross-Origin Resource Sharing (CORS)



Fonte: https://www.w3.org/TR/cors/

Obrigado!