Acessando Middleware via Ajax

EC021 - Tópicos Avançados II

Sistemas Distribuídos

Prof. Adauto Mendes | E-mail: adauto.mendes@inatel.br

O modo convencional da Web

- O comportamento típico de navegação consiste em carregar uma página da web, selecionar alguma ação que desejamos fazer, preencher um formulário, enviar as informações etc.
- Trabalhamos dessa maneira sequencial, solicitando uma página por vez, e temos que esperar que o servidor responda, carregando uma página web totalmente nova antes de continuarmos.
- Essa também é uma das limitações das páginas da Web, em que a transmissão de informações entre um cliente e um servidor geralmente requer que uma nova página seja carregada.

O modo convencional da Web

- O JavaScript é uma maneira de reduzir o tempo de resposta do cliente-servidor, usando-o para verificar as informações de formulário (ou outras) antes de enviá-las para um servidor.
- Uma das limitações do JavaScript costumava ser que não havia como se comunicar diretamente com um servidor da web.
- Outra desvantagem desse método de acesso sequencial usual é que há muitas situações em que você carrega uma nova página que compartilha muitas partes iguais às antigas (considere o caso em que você tem uma “barra de menu” na parte superior ou lateral da página. isso não muda de página para página).

Modo atual

- Até certo tempo não havia nenhuma alternativa para fugir desta maneira convencional. Eis que surgiu o Ajax...
- Ajax é um meio de usar JavaScript para se comunicar com um servidor web submeter um formulário ou carregar uma nova página.
- O Ajax faz uso de um objeto interno, **XMLHttpRequest**, para executar essa função.
- Este objeto ainda não faz parte do padrão DOM (Document Object Model), mas é suportado (em diferentes formas) pela maioria dos navegadores.
- O termo "Ajax" foi criado em 2005, mas o objeto XMLHttpRequest foi suportado pela primeira vez pelo Internet Explorer vários anos antes disso.

O Ajax

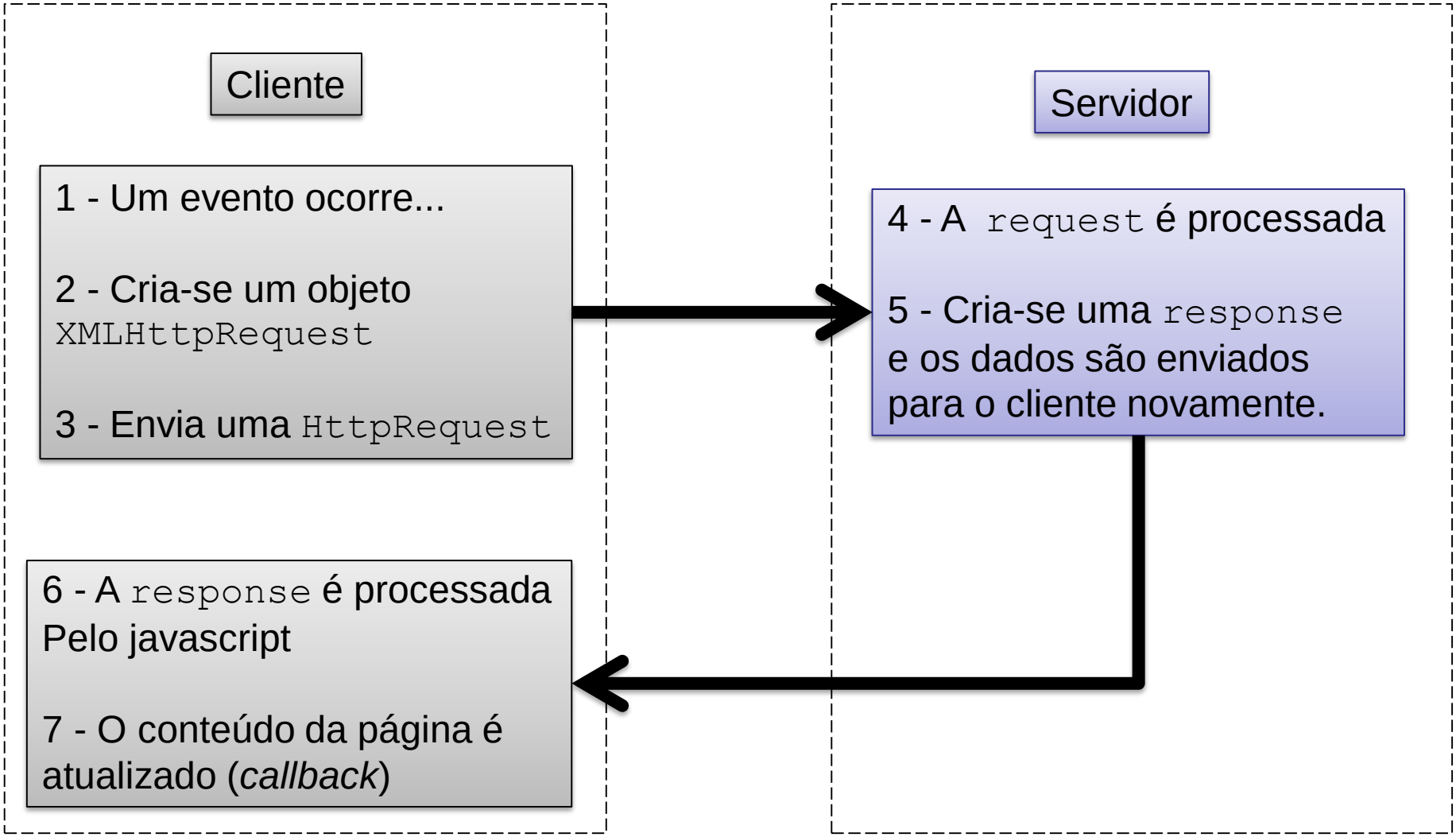
- De acordo com o w3c, o AJAX é o sonho de um desenvolvedor, porque você pode:
  - Ler dados de um servidor da web - após a página ter sido carregada.
  - Atualizar uma página da web sem recarregar a página.
  - Enviar dados para um servidor da Web - em segundo plano.

Definição

AJAX = **A**synchronous **J**avaScript **A**nd **X**ML.

- AJAX não é uma linguagem de programação.
- AJAX usa apenas uma combinação de:
  - Um objeto **`XMLHttpRequest`** integrado ao navegador (para solicitar dados de um servidor da Web).
  - DOM de JavaScript e HTML (para exibir ou usar os dados).
- AJAX é um nome enganoso. Aplicativos AJAX podem usar XML para transportar dados, mas o modo mais utilizado hoje para transportar dados é como texto simples ou texto **JSON**.

Como funciona?



Tratando respostas

- Um *callback* é necessário porque um AJAX, como o nome “Javascript e XML Assíncrono” sugere, é **assíncrono**.
- Quando você inicia uma chamada AJAX usando o método XMLHttpRequest nativo ou por jQuery, a requisição HTTP é enviada, mas o motor do JavaScript não espera por uma resposta.
- Ao invés disso, o fluxo de execução continua. Para conseguirmos monitorar o progresso da requisição, nos é permitido passar uma função de *callback* que será executada quando o estado da requisição HTTP muda. O callback pode ser enviado para o objeto **`XMLHttpRequest`** nativo através do atributo **`onreadystatechange`**.

Como funciona?

1. Um evento ocorre em uma página da web (ex.: a página é carregada, um botão é clicado)
2. Um objeto `XMLHttpRequest` é criado pelo JavaScript
3. O objeto `XMLHttpRequest` envia uma solicitação para um servidor da web
4. O servidor processa a solicitação
5. O servidor envia uma resposta de volta para a página da web
6. A resposta é lida pelo JavaScript
7. A ação apropriada (como a atualização da página) é executada pelo JavaScript – *callback*.

jQuery Ajax

- jQuery é uma biblioteca JavaScript que funciona cross-browser desenvolvida para simplificar os scripts client side que interagem com o HTML.
- jQuery é uma biblioteca de código aberto e possui licença dual, fazendo uso da Licença MIT ou da GNU General Public License versão 2. A sintaxe do jQuery foi desenvolvida para tornar mais simples a navegação do documento HTML, a seleção de elementos DOM, criar animações, manipular eventos e desenvolver aplicações AJAX.

jQuery Ajax

- Na biblioteca jQuery, uma das funções mais utilizadas é a `$.ajax()`, que, com uma sintaxe bastante simples, permite enviar e tratar o resultado de requisições assíncronas.

jQuery Ajax

- Abaixo temos a sintaxe de uma requisição AJAX feita no jQuery:

```
$.ajax({
    url: /* URL do seu endpoint */,
    type: /* tipo de método HTTP (o padrão é GET) */,
    data: {
        /* Dados que irão no body da request */
    },
    beforeSend: function () { /* Request está para ser enviada */ },
    success: function (result,status,xhr) { /* Request completou com sucesso */},
    error: function () { /* Request completou com erro */},
    complete: function () { /* Request completou */}
})
```

jQuery Ajax

- Ao lado temos um exemplo de uma request GET via AJAX escrita usando jQuery:

```
$.ajax({
    url: 'http://localhost:3000/toddy/listar',
    type: 'GET',
    beforeSend: function () {
        console.log('Enviando request...');
    },
    success: function (result,status,xhr) {
        console.log(result);
    },
    error: function () {
        console.log('Erro na request...');
    },
    complete: function () {
        console.log('Request finalizada.');
    }
});
```

jQuery Ajax

- Ao lado temos um exemplo de uma request POST via AJAX escrita usando jQuery:

```
$.ajax({
    url: 'http://localhost:3000/toddy/inserir',
    type: ‘POST’,
    data: {
        lote: ‘ABC’,
        conteudo: 150,
        validade: 15/12/2018
    },
    beforeSend: function () {
        console.log('Enviando request...');
    },
    success: function (result,status,xhr) {
        console.log(result);
    },
    error: function () {
        console.log('Erro na request...');
    },
    complete: function () {
        console.log('Request finalizada.');
    }
});
```

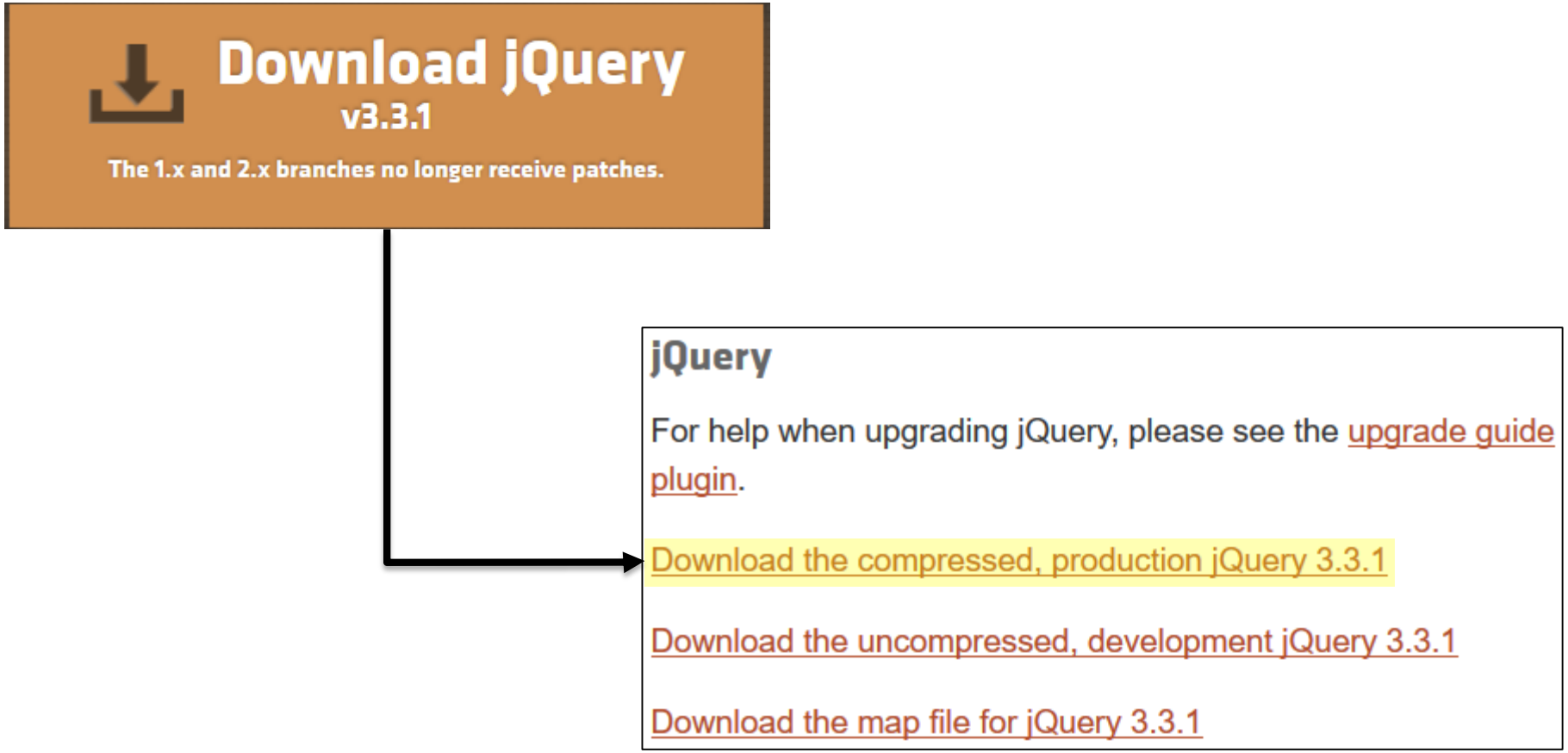
Prática

- Crie uma página HTML com a seguinte estrutura:

```
<!DOCTYPE html>
<html lang="pt-BR">

<head>
    <meta charset="utf-8">
    <title>EC021</title>
</head>

<body>

</body>

</html>
<script>

</script>
```

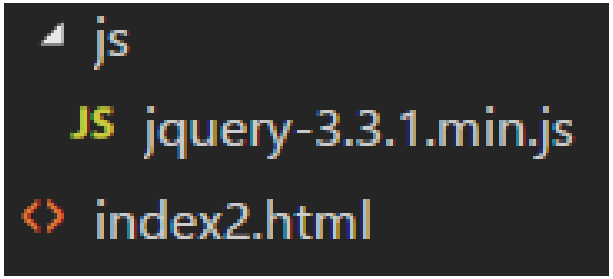
Prática

- Faça o download do jQuery do site www.jquery.com



Prática

- Crie uma pasta ‘js’ dentro de onde se encontra o seu projeto e salve o arquivo do jQuery dentro dela, ficando com esta estrutura:



- Importe o jQuery no final ‘body’ da sua página:

```
    <script type="text/javascript" src="js/jquery-3.3.1.min.js"></script>
</body>
```

Prática

- Implemente em uma página HTML uma chamada AJAX para cada endpoint do seu middleware. Lembre-se:
  - Nas operações de INSERT, UPDATE e DELETE, o type deverá ser setado como **POST** e a request deverá ter **body**, ou seja, no AJAX você deverá setar o parâmetro '**data**'.