UNINOVE

Ajax

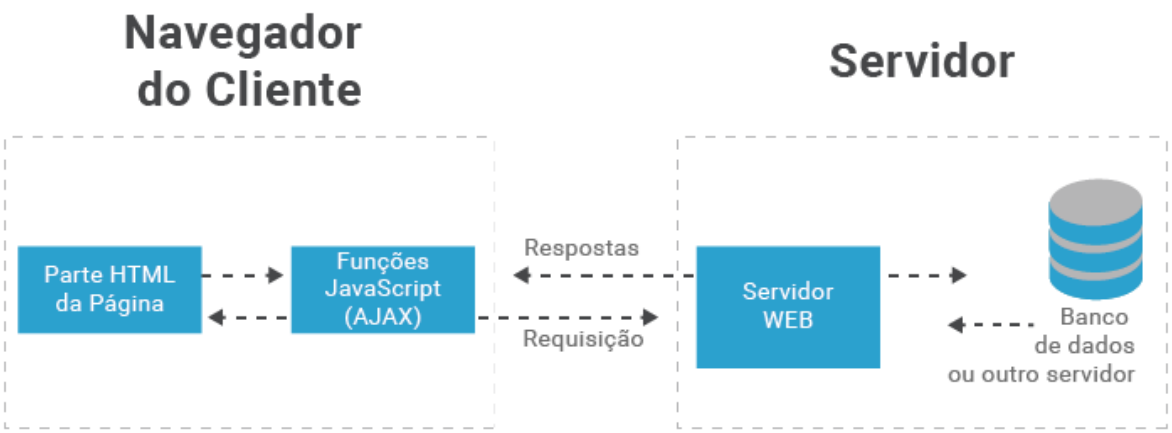
ESTUDAR O USO E OS RECURSOS DE AJAX E JSON

AUTOR(A): PROF. DANIEL FERREIRA DE BARROS JUNIOR

AJAX

O significado de AJAX é Asynchronous JavaScript e XML, utilizado para realizar comunicação entre o script e um servidor, recendo dados nos mais diversos formatos, por exemplo, HTML, arquivo texto, XML, JSON, CSV, entre outros. AJAX trabalha de maneira assíncrona, ou seja, permite realizar novas requisições ao servidor sem realizar um novo carregamento da página web, além de possibilitar a utilização (envio e recebimento) de dados com servidores.

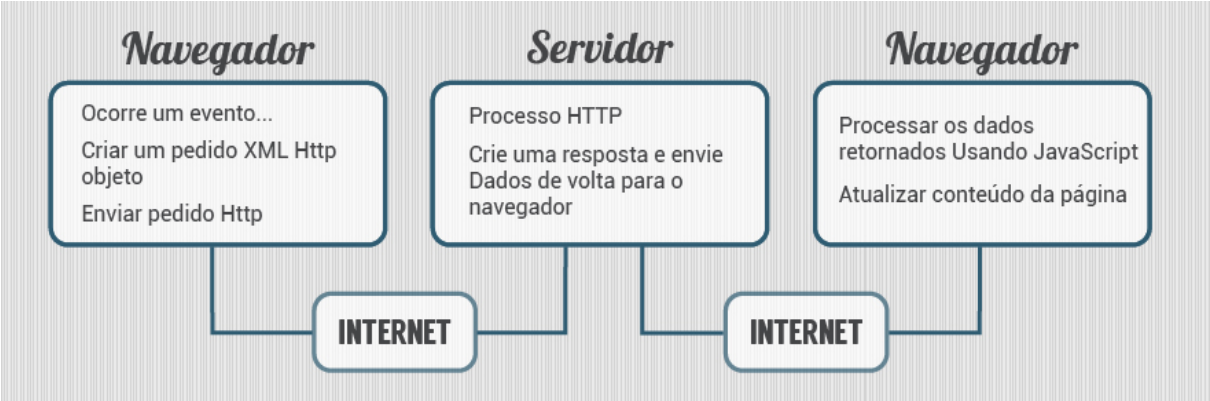
A imagem a seguir exemplifica as requisições entre o navegador e o servidor, inclusive as requisições internas, entre os scripts e servidores de dados (quando necessário).



Legenda: FLUXO DE DADOS CLIENTE - SERVIDOR

AJAX não uma linguagem de programação, mais sim a utilização de várias tecnologias, com a combinação de objetos, com o XMLHttpRequest para requisição de dados e a utilização de HTML DOM na apresentação destes valores, além de XML e o JavaScript conectando todos. O AJAX foi criado pela equipe da Microsoft e em 2006 foi inicialmente especificado pela W3C e recomendado posteriormente.

A ordem de funcionamento do AJAX se dá da seguinte forma:



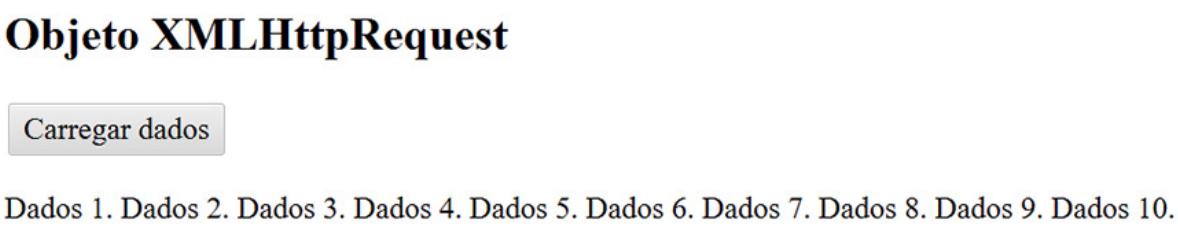
De acordo com a imagem acima, temos o seguinte fluxo de eventos:

- Um evento é iniciado em uma página web.
- Um script em JavaScript cria um objeto l XMLHttpRequest.
- O objeto envia uma requisição a um servidor web.
- O servidor processa a requisição.
- O servidor transmite a resposta ao cliente, diretamente ao navegador.
- A resposta é interpretada pelo script em JavaScript.
- Uma ação é realizada pelo JavaScript de acordo com a resposta obtida.

Exemplo AJAX

Vamos criar um exemplo simples de utilização de AJAX. Para tanto vamos utilizar um arquivo texto obtendo os dados a serem adicionados na página quando solicitado.
Vamos ao exemplo a seguir:

```
1. <!DOCTYPE html>
2. <html>
3.     <body>
4.         <div>
5.             <h2>Objeto XMLHttpRequest</h2>
6.             <button type="button" onclick="carregarArquivo()">Carregar dad
7.             <p id="retorno"></p>
8.         </div>
9.
10.         <script>
11.             function carregarArquivo() {
12.                 var ajax = new XMLHttpRequest();
13.                 ajax.onreadystatechange = function () {
14.                     if (this.readyState == 4 && this.status == 200) {
15.                         document.getElementById("retorno").innerHTML =
16.                                 this.responseText;
17.                     }
18.                 };
19.                 ajax.open("GET", "ajax_dados.txt", true);
20.                 ajax.send();
21.             }
22.         </script>
23.     </body>
24. </html>
```



A imagem acima corresponde ao exemplo acima.

Vamos analisar alguns aspectos dele.

A linha 13 captura o evento onreadystatechange do XMLHttpRequest e avalia se houve uma resposta do servidor a requisição transmitida.

Para tanto devemos tratar o status da comunicação com o objeto XMLHttpRequest, utilizando o readyState.

Existe uma lista de status a ser avaliado, conforme a seguir:

Status 0 – O objeto foi criado porem ainda não iniciado.

Status 1 - A comunicação foi iniciada.

Status 2 - A requisição foi iniciada com o servidor, método send foi iniciado.

Status 3 - A resposta a requisição está sendo recebida, em processamento.

Status 4 - A requisição está completa.

Na linha 14 avaliamos o readyState obtido, e de acordo com a resposta adequada, realizamos a operação necessária. Ainda na linha 14, avaliamos o status da requisição.

Status de requisição mais importantes:

- 200: "OK"
- 403: "Proibido"
- 404: "Não encontrado"

DICA:

Para saber mais, a lista completa pode ser vista em:
https://www.w3schools.com/tags/ref_httpmessages.asp
(https://www.w3schools.com/tags/ref_httpmessages.asp)

Para saber mais sobre AJAX, veja em:
https://www.w3schools.com/js/js_ajax_intro.asp (https://www.w3schools.com/js/js_ajax_intro.asp)

Determinamos que os dados serão adicionados apenas se o status da requisição foi igual a 200, ou seja, que está correto ou “OK”.

A linha 19 determina a origem dos dados, seja um endereço web, ou endereço do arquivo. O método pode variar de GET ou POST, por fim, o valor true determina que a requisição é assíncrona.

A linha 20 executa a função send(), que envia de fato a requisição ao servidor.

JSON

O JSON ou JavaScript Object Notation é um modelo e padrão destinado a troca de dados, a exemplo do XML. O uso do objeto JSON dispõe de métodos para análise ou conversões de dados de maneira simples. Ainda, representa os seus dados de maneira tipada, ou seja, determina o nome (campo) e o valor de cada dado.

Por trafegar os dados em formato texto, a sua leitura é mais simplificada e universal em relação a outros padrões o que também otimiza o transporte dos dados.

O seu tamanho de arquivo é reduzido em relação a outros formatos, como o XML. Fator altamente desejado para transferência de dados, seja entre cliente e servidor ou entre servidores.
Vamos verificar alguns exemplos de sintaxe json:

```
1. Array:
2. [ "SP", "RJ", "MG", "ES"]
```

```
1. Objeto
2. {
3.         "ano" :  1990,
4.         "altura": 1.80,
5.         "nome": "João"
6. }
```

```
1. Array de objetos
2. [
3.    {
4.      "titulo": "JSON x XML",
5.      "resumo": "Modelos de dados",
6.      "ano": 2017
7.     },
8.    {
9.      "titulo": "JSON e AJAX",
10.     "resumo": "A dupla dinâmica em ação",
11.     "ano": 2017
12.    }
13. ]
```

AJAX e JSON

Podemos utilizar ambas as técnicas, AJAX e JSON, e obter um interessante resultado. Consumir dados de arquivos JSON é uma prática bastante atual, pois a disponibilidade de dados no formato JSON é cada vez mais comum devido a troca de dados nos mais diversos tipos de aplicativos.
Veja o exemplo a seguir:

```
<!DOCTYPE html>
<html>

    <body>

        <h2>Uso do XMLHttpRequest com JSON.</h2>
        <h3>FILMES</h3>
        <div id="dado"></div>
        <script>
            var xmlhttp = new XMLHttpRequest();
            xmlhttp.onreadystatechange = function () {
                if (this.readyState == 4 && this.status == 200) {
                    var valor = "";
                    myObj = JSON.parse(this.responseText);
                    for (i in myObj.filme) {
                        valor += "Titulo: " + myObj.filme[i].titulo + " |
                        valor += "Ano: " + myObj.filme[i].ano;
                        valor += "   ( " + myObj.filme[i].genero + " )<br>
                    }
                    document.getElementById("dado").innerHTML = valor;
                }
            };
            xmlhttp.open("GET", "json_dados.json", true);
            xmlhttp.send();
        </script>
        <p>Abrir o arquivo <a href="json_dados.json" target="_blank">json_
    </body>
</html>
```

Salve o arquivo JSON como json_dados.json no mesmo diretório do exemplo anterior, está com o seguinte valor:

```
{
  "filme":[
        {
            "titulo":"Titulo do seu filme 1",
            "ano":"2015",
            "genero":"Terror"
        },
        {
            "titulo":"Titulo do seu filme 2",
            "ano":"2016",
            "genero":"Comedia"
        },
        {
            "titulo":"Titulo do seu filme 3",
            "ano":"2017",
            "genero":"Aventura"
        }
    ]
}
```

No código acima é possível unir o AJAX com os dados provenientes do JSON. Esta combinação é muito proveitosa, uma vez que diversos provedores e serviços web disponibilizam dados em formato JSON. Outra possibilidade vista no programa é a utilização de arrays de múltiplas linhas, permitindo criar conteúdo dinâmicos provenientes até mesmo de bando de dados. Nesta simulação também foi possível determinar quais campos foram apresentados e na ordem adequada.

A seguir a imagem do resultado do script acima, resultante do AJAX e dados externos provenientes do JSON.

Uso do XMLHttpRequest com JSON.

FILMES

Titulo: Titulo do seu filme 1 | Ano: 2015 (Terror)
Titulo: Titulo do seu filme 2 | Ano: 2016 (Comedia)
Titulo: Titulo do seu filme 3 | Ano: 2017 (Aventura)

Abrir o arquivo json_dados.json

SUGESTãO:

Treine o uso deste recurso, crie seus próprios dados e apresente em seu site HTML nos mais diversos formatos.

DICA:

Para saber mais, veja em:
AJAX Introduction
https://www.w3schools.com/js/js_ajax_intro.asp (https://www.w3schools.com/js/js_ajax_intro.asp)
JAVASCRIPT WEB APIS
https://www.w3.org/standards/webdesign/script (https://www.w3.org/standards/webdesign/script)

Conclusão

O uso de AJAX potencializa a experiência do usuário, pois com ele é possível entregar páginas em menor tempo, devido à natureza de suas requisições assíncronas. Observe com atenção as respostas do servidor referente as requisições solicitadas, e sem bem tratadas, a resposta via AJAX será muito satisfatória.
Complementar o uso de AJAX com dados de origem em JSON é outro diferencial. JSON tornou-se muito atrativo nos dias atuais, devido a sua grande popularidade e disponibilidade.
Aproveite o máximo desta combinação em suas páginas HTML, por meio do JavaScript.

ATIVIDADE

Entre os status de requisição em AJAX, reportado pelo servidor, qual é a alternativa que represena o status "Não encontrado"?
Escolha a alternativa correta:

A. 4: "Não encontrado"
B. 404: "Não encontrado"
C. 501: "Não encontrado"
D. 403: "Não encontrado"

ATIVIDADE

Escola a alternativa que indica a uma forma correta de se criar um array em JSON.

A. ["SP", "RJ", "MG", "ES"]
B. ["SP, RJ, MG, ES"]
C. ["SP" "RJ" "MG" "ES"]
D. "SP", "RJ", "MG", "ES"

ATIVIDADE

Sobre AJAX, escolha a alternativa que apresenta a maneira correta de se solicitar dados de origem externa e assíncrona.

A. ajax.open("GET", "ajax_dados.txt", true);
B. ajax.open("GET", "ajax_dados.txt", false);
C. ajax.open("GET", "http://.txt", true);
D. ajax.open("_GET", "ajax_dados.txt", POST);

REFERÊNCIA

MORRISON, M. Use a cabeça JavaScript. 5º Ed. Rio de Janeiro: Alta Books, 2012. 606 p.

OLIVIERO. C. A. J. Faça um site JavaScript orientado por projeto. 6º ed. São Paulo: Érica, 2010. 266 p.

ZAKAS, Nicholas C. JavaScript de alto desempenho. 8º Ed. São Paulo: Novatec, 2010. 245 p.