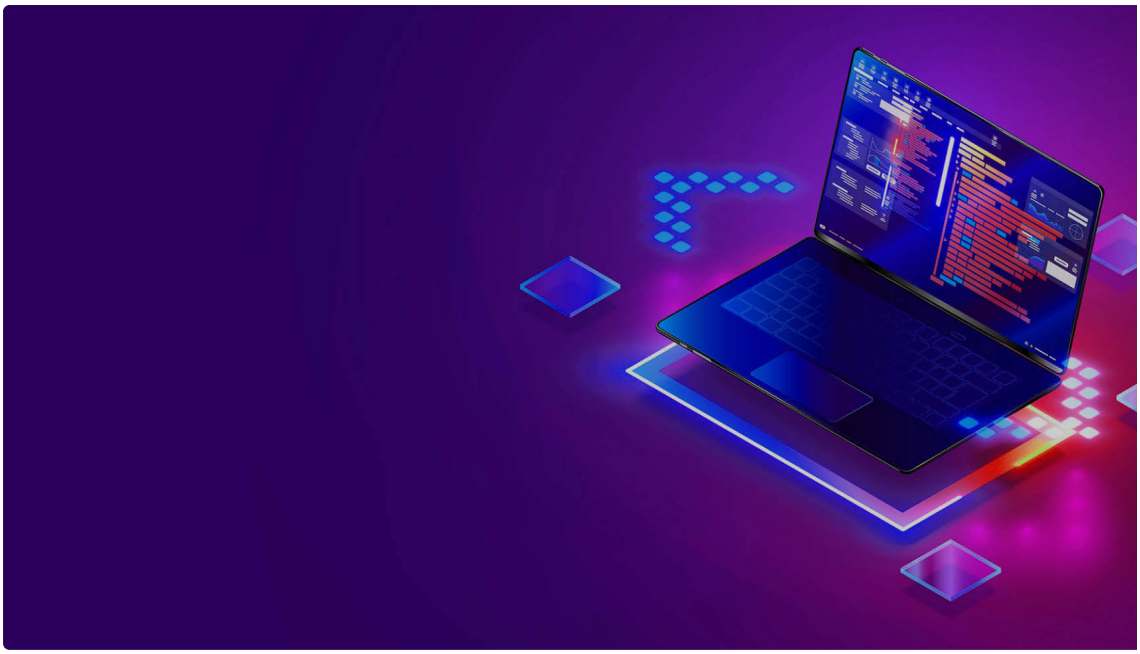


Tecnologias de transmissão de dados em sistemas web

Prof. Alexandre de Oliveira Paixão

Prof. Kleber de Aguiar

Descrição

Linguagem de marcação XML, serialização de dados em sistemas web, formato de transmissão de dados JSON, formato de transmissão de dados YAML, consumo de dados serializados nas requisições AJAX.

XML: eXtensible Markup Language.

JSON: Javascript Object Notation.

YAML: YAML Ain't Markup Language.

Propósito

Apresentar os conceitos de transmissão e consumo de dados em sistemas web, utilizando os formatos XML, JSON e YAML e obter o conhecimento básico necessário para o consumo de dados nesses formatos em requisições AJAX.

Preparação

Para aplicação dos exemplos, será necessário um editor de texto com suporte a linguagens de marcação. No Sistema Operacional Windows, é

indicado o Notepad++. No Linux, o Nano Editor. Além disso, para visualização dos exemplos dos códigos que utilizam AJAX, será necessário hospedar as páginas HTML em um servidor web. No SO Windows, é indicada a instalação/utilização de softwares, como o XAMP, que consistem em um ambiente contendo, entre outros softwares, o servidor web Apache. Uma alternativa é a instalação apenas do próprio Apache ou de outro servidor, como o IIS. Em ambiente Linux, recomenda-se a utilização dos servidores Apache ou Nginx.

Baixe o arquivo "**Códigos-fontes dos exercícios_Tecnologias de Transmissão de Dados em Sistemas Web**", descompactando-o em seu dispositivo. Assim, você poderá utilizar os códigos como material de apoio no seu momento de estudo!

Objetivos

Módulo 1

Linguagem de marcação XML

Descrever a linguagem de marcação XML e sua aplicabilidade em sistemas web.

Módulo 2

JSON e YAML

Descrever os formatos de transmissão de dados JSON e YAML.

Módulo 3

Os formatos de transmissão em requisições AJAX

Compreender como utilizar dados nos formatos XML, JSON e YAML em requisições AJAX.

Introdução

No desenvolvimento de software, mais precisamente no que concerne às informações a serem manipuladas, lidamos, normalmente, com algumas fontes de dados: os provenientes da própria aplicação, os provenientes de outras fontes, como outros sistemas, bases de dados externas, APIs (*Application Programming Interface*) etc. Em paralelo, temos ainda os processos de recuperação e armazenamento de tais informações.

Além disso, há outro fator que deve ser levado em consideração: o formato dos dados que trataremos em nosso software, onde podemos ter dados nos mais diversos formatos e até mesmo dados não formatados.

Ao longo deste estudo, veremos três dentre os vários formatos de serialização e transmissão de dados disponíveis. Tais formatos — a linguagem de marcação XML, o JSON e o YAML — serão descritos conceitualmente. Também demonstraremos, de forma prática, como utilizar esses formatos em requisições AJAX.

1 - Linguagem de marcação XML

Ao final deste módulo, você será capaz de descrever a linguagem de marcação XML e sua aplicabilidade em sistemas web.

Visão geral

A linguagem XML

A linguagem **XML** — acrônimo para *eXtensible Markup Language* – é, a exemplo da HTML, uma linguagem de marcação. Criada pelo W3C (World Wide Web Consortium), em 1996, e transformada em uma recomendação pelo mesmo órgão em 1998, essa linguagem possui algumas importantes diferenças em relação à HTML, visto que foi criada justamente para ser diferente desta — em outras palavras, para estender as funcionalidades da HTML.

A XML é um padrão para a formatação e transmissão de dados de fácil entendimento tanto para humanos quanto para máquinas. Sua principal

característica — e diferença para HTML – é ser composta por tags definidas pelo usuário (ou programador). Diferentemente da HTML, na qual todas as tags são predefinidas, em XML nós mesmos criamos nossas tags.

Saiba mais

A principal função e utilização da **XML** é transmitir dados por meio da web.

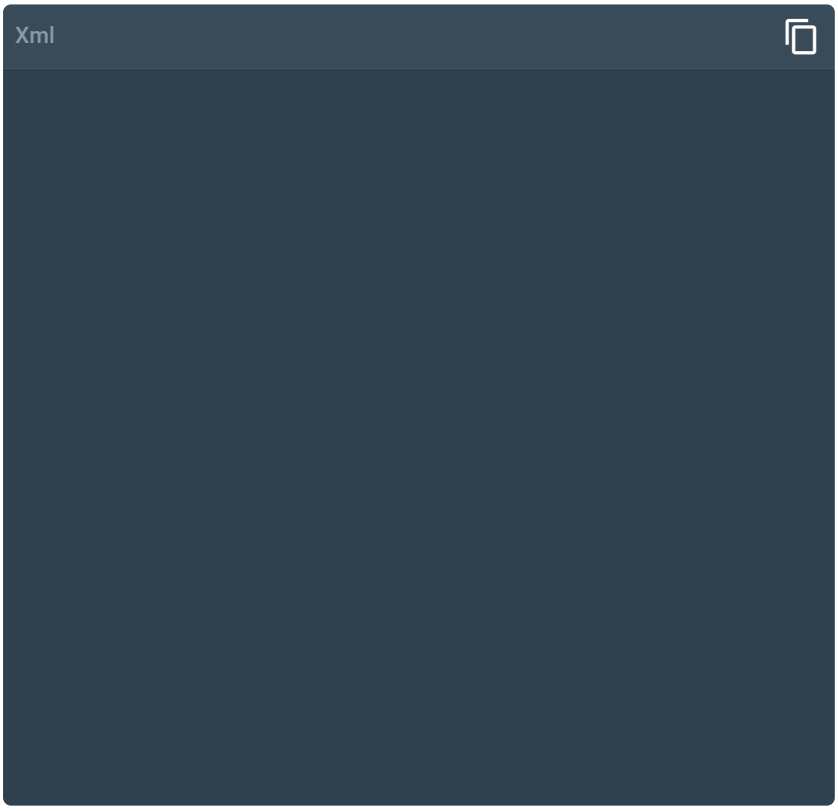
Documento XML

Anatomia de um arquivo XML

Os documentos XML são constituídos por unidades de armazenamento chamadas **entidades**, que contêm dados. Esses dados são compostos por caracteres, alguns dos quais formam dados de caracteres enquanto outros formam a marcação (W3C, 2020).

Vejamos, a seguir, nosso primeiro arquivo XML. A partir deste arquivo simples, começaremos a entender o seu formato.

Xml



Agora, veja a análise de linha a linha da XML apresentada.

Na **primeira linha**, temos a declaração XML, responsável por especificar a versão e por informar ao navegador que se trata de um arquivo XML. Nessa primeira linha, também é definido o **charset** do documento.

Na **segunda linha**, representado pela tag <correntistas>, temos o nó principal — ou nó raiz — do documento.

Na **terceira linha**, temos um comentário. Repare que os comentários em um documento XML são iguais aos usados em HTML.

Na **quarta linha**, temos o primeiro elemento filho do nó raiz, o elemento <pessoa>.

Na **quinta, sexta e sétima linhas**, temos os elementos filhos do elemento <pessoa>.

Nas **linhas seguintes**, temos outro elemento <pessoa> e seus respectivos elementos filhos.

Charset

É o conjunto de caracteres utilizados para escrever o documento.

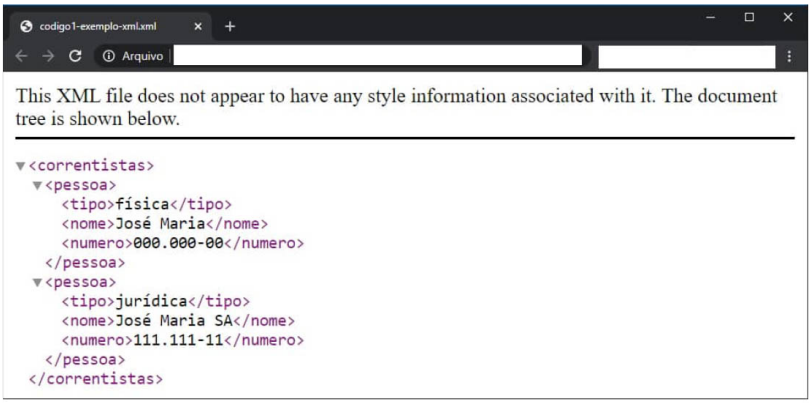
Em termos de sintaxe, repare nos seguintes itens:

- Toda tag deve ser iniciada e fechada.
- Há um aninhamento representando hierarquia entre os dados. Por exemplo: as tags <tipo>, <nome> e <numero> estão indentadas, aninhadas e englobadas pela tag <pessoa>. Da mesma maneira, todas as tags filhas estão aninhadas dentro do nó principal, <correntistas>;

Na imagem a seguir, veja como o documento XML é **renderizado** no navegador:

Renderizado

Renderização é o processo pelo qual se obtém o produto final de um processamento digital qualquer.



Documento XML exibido no navegador web.

Considerando sua sintaxe, dizemos que **XML** é um documento bem formatado. Logo, deixar de fechar uma tag ou construir um documento sem tag raiz torna o arquivo mal formatado ou inválido.

Na imagem a seguir, pode ser visto o resultado da renderização no navegador de um documento XML mal formatado – nesse caso, foi removida a tag de fechamento do elemento <correntistas>, do documento XML visto anteriormente.

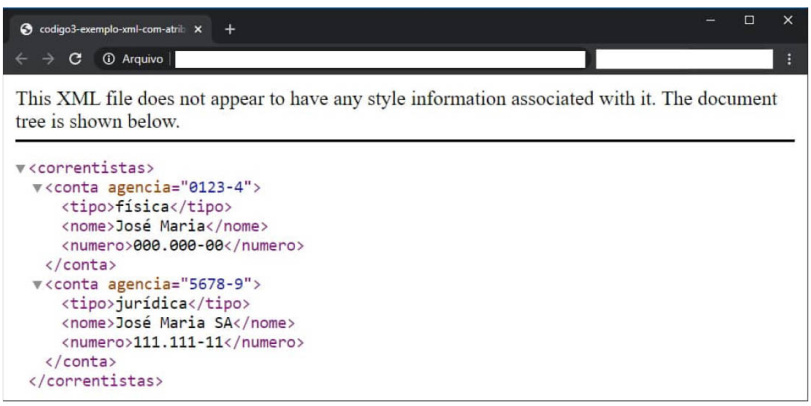
Renderização de XML mal formatado.

Utilizando atributos

Veja este novo exemplo de documento XML:

```xml
```

Agora, veja sua renderização no browser:



Exemplo de documento XML contendo atributo.

Em relação ao primeiro documento XML, neste último, no elemento <conta>, foi adicionado o atributo “agencia” e um valor para esse atributo. Logo, podemos ter em um documento XML, além dos elementos, atributos. Inclusive, poderíamos modificar o documento anterior, trocando os elementos filhos de <conta> por atributos, gerando este novo documento:

Xml

Perceba que, no lugar de elementos filhos, os dados de cada conta foram armazenados diretamente no elemento <conta>, na forma de atributos. Sobre os atributos, como visto no exemplo, é necessário envolvê-los com aspas. Por último, veja que o elemento <conta> foi simplificado, sendo aberto e fechado em uma única linha, com a utilização da barra antes do sinal de maior (/>).

Em termos conceituais, um arquivo XML é composto por:

1. Dados de caracteres (sequência de texto).
2. Instruções de processamento em anotações, normalmente inseridas no cabeçalho do documento.
3. Comentários, quando necessário.
4. Declaração de entidade.
5. Nós — elemento rotulado com um nome ou conjunto de atributos, cada um contendo nome e valor.

Por que utilizar XML? Como vimos, ao tratarmos de sistemas web, a aplicabilidade da XML é servir como formato de transmissão de dados. Nesse sentido, considerando que tal formato foi especificado para ser simples, de fácil leitura por humanos e computadores, temos a principal justificativa quanto à sua adoção em sistemas web.

Além disso, podemos ainda mencionar as seguintes vantagens de utilização da XML:

- XML é autodocumentado, ou seja, seu formato descreve sua estrutura, seus elementos e valores.
- XML é independente de plataforma e linguagem de programação.
- XML é facilmente interpretado pela maioria das linguagens de programação (nas quais normalmente utilizamos recursos chamados de "parsers", responsáveis por interpretar a estrutura e os dados do documento).
- XML é extensível. Logo, podemos tanto usar entidades criadas por outros quanto criar nossas próprias tags e atributos.
- XML possui suporte a **Unicode**.
- XML possui suporte à validação através de **DTD** e **Schema**.

Unicode

É um padrão internacional que permite que todos os caracteres, de qualquer sistema de escrita existente, possam ser entendidos e manipulados por computadores.

DTD

Document Type Definition.

Schema

Estrutura pré-definida com regras de validação de um documento.

Convém também citar algumas desvantagens desse formato para a transmissão de dados por meio da Internet em relação a outros disponíveis:

Desvantagem 1

Um documento XML possui a sintaxe detalhada e redundante quando comparada a outros formatos baseados em texto, como JSON.

Desvantagem 2

Um documento XML não suporta arrays.

Desvantagem 3

Um documento XML é menos legível que outros formatos como JSON e YAML.

Desvantagem 4

Um documento XML pode gerar arquivos grandes dada a sua sintaxe detalhada.

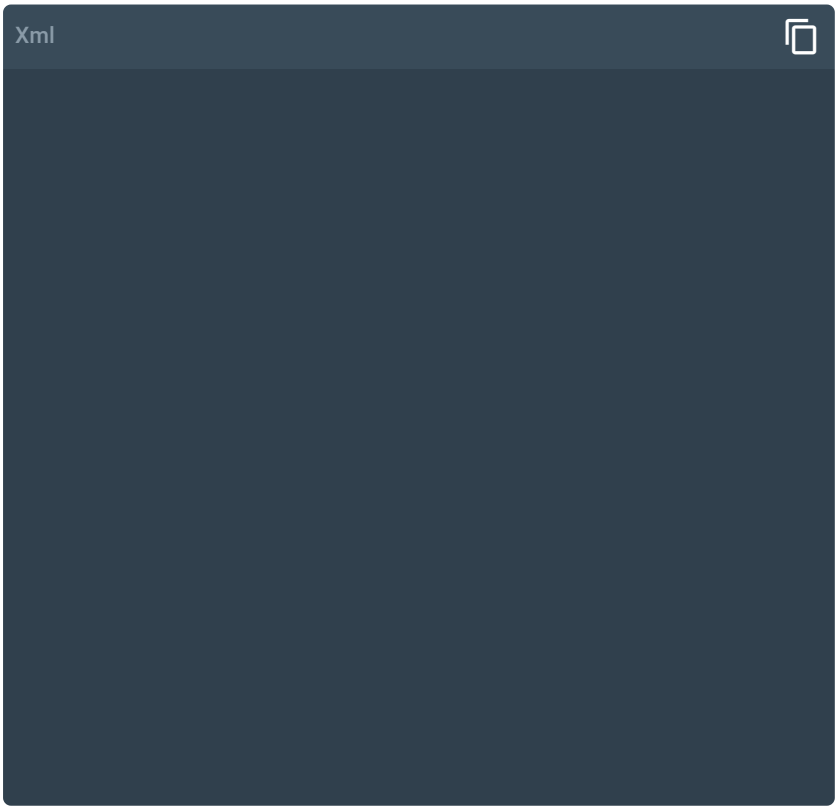
XML e a interface DOM

> **O DOM (*Document Object Model*) é uma interface de plataforma e linguagem neutra que permite que programas e scripts acessem e atualizem dinamicamente o conteúdo, a estrutura e o estilo de um documento (W3C, 2020).**

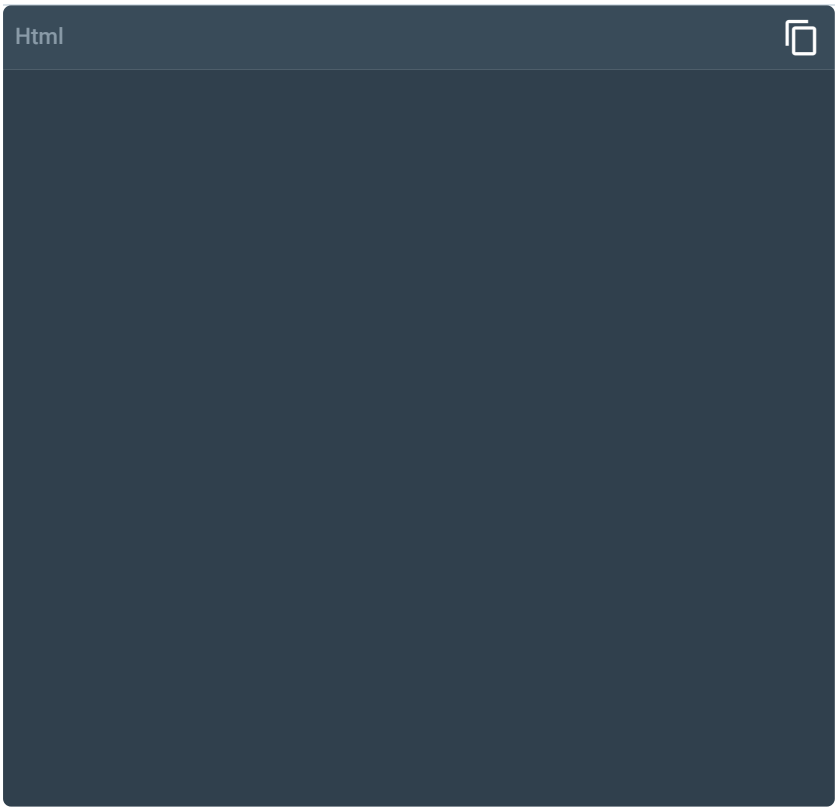
Utilizando a interface DOM, é possível manipular tanto documentos HTML quanto documentos XML. Em ambos, o documento, através desse objeto, é representado na forma de árvore.

Manipulando um arquivo XML com a interface DOM

O **XML DOM** é um padrão para obter, modificar, adicionar ou remover elementos XML. Por meio dele, é possível acessar todos os elementos de um documento XML. Utilizando o documento XML já apresentado e, novamente, disponibilizado a seguir, veja os dois passos do código no qual, utilizando Javascript, o arquivo em questão é manipulado.



Passo 01.



Passo 02.

Como visto no código, foi criada uma string JS que, em seguida, foi transformada em um documento XML válido. Estando nesse formato,

foi possível manipular o seu conteúdo por meio do DOM. Com o método **"getElementsByTagName"** e as propriedades **"childNodes"** e **"nodeValue"**, foi acessado o conteúdo do primeiro elemento <nome> e atribuído o seu valor à tag HTML <p>, para exibição do resultado na tela.

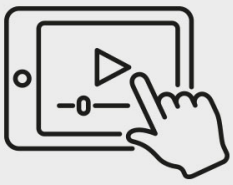
Atenção!

Nos exemplos aqui demonstrados, foi utilizada a linguagem de programação Javascript. Entretanto, conforme já mencionado, é possível usar qualquer linguagem de sua preferência para a manipulação do XML DOM.

Exemplos de manipulação de XML DOM

A seguir, assista ao vídeo sobre exemplos de manipulação de XML DOM.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Veja a seguir as propriedades do XML DOM:

- nodeName;
- nodeValue;
- parenteNode;
- childNodes;
- attributes.

Em relação aos métodos para o XML DOM, estes são alguns dos disponíveis:

- getElementsByTagName(name);
- getAttributeNode(node);
- appendChild(node);
- removeChild(node);

- createElement(name);
- createTextNode(value).

Outras formas de navegar por um documento XML

Existem outras formas de navegar pelos elementos e atributos de um documento XML, tais como:

XPATH

É uma recomendação do W3C e possui mais de 200 funções prontas para navegar em documentos XML utilizando a sintaxe “path like”.

XQUERY

É uma linguagem para realização de consultas no formato de queries – representando para o XML o mesmo que o SQL representa para os bancos de dados.

Recursos avançados

Além de tudo o que você aprendeu até aqui, há muito mais para se estudar e aprender a respeito de XML. Por exemplo: DTDs (*Document Type Definition*), Schemas, Entidades, Notações, tipos de dados (datas, números, strings etc.). Embora o que vimos seja o bastante para utilizar o formato XML para a transmissão de dados em sistemas web, é recomendado que você conheça esses recursos mais avançados para caso precise deles. Para um maior aprofundamento no assunto, você pode conhecer mais sobre a especificação W3C.

Falta pouco para atingir seus objetivos.

Vamos praticar alguns conceitos?

Questão 1

Considerando o documento XML a seguir, assinale a alternativa correta e que descreva o porquê desse documento não ser considerado válido (bem formatado).

XML

A — Não é permitido iniciar um documento XML com uma tag escrita em letras minúsculas.

B — Em XML podemos criar nossas próprias tags. Entretanto, não é possível utilizar nomes de tags já existentes em HTML, como a tag <body>, acima.

C — As tags XML precisam ser abertas e fechadas corretamente. Logo, uma tag não fechada da maneira correta torna o documento inválido.

D — Não é permitido utilizar verbos como nome de tags em XML.

E — Não é permitido iniciar nomes de tags com letra maiúscula em XML.

Parabéns! A alternativa C está correta.

Na linguagem de marcação XML, devemos ter alguns cuidados, embora tenhamos a flexibilidade de criarmos nossas próprias tags. Nesse sentido, o documento precisa, entre outras premissas, ser composto por tags válidas — sendo uma tag considerada válida quando aberta e fechada adequadamente, por exemplo. No exemplo anterior, a tag foi fechada incorretamente, tornando assim o documento inválido.

Questão 2

Em relação ao XML DOM, assinale a afirmativa correta:

A

A interface DOM é um padrão para manusear documentos nos formatos HTML e XML. Com essa interface, podemos acessar os elementos e atributos de um documento por meio das propriedades e métodos por ela disponibilizados.

B

A interface DOM, relacionada a documentos XML, nada tem a ver com a interface DOM, relacionada a documentos HTML, já que ambas são linguagens não relacionadas.

C

Em XML, temos a liberdade de criar nossos próprios métodos dentro da interface DOM para manipulação de um documento, da mesma forma que temos liberdade para criar nossas próprias tags.

D

A manipulação de documentos XML por meio do DOM só é possível com a linguagem de programação Javascript.

E

Por meio do padrão XML DOM é possível obter, adicionar ou remover elementos XML, entretanto não há como realizar modificações nos elementos acessados por esse padrão.

Parabéns! A alternativa A está correta.

O Document Object Model (DOM) é uma interface de plataforma e linguagem neutra. Logo, documentos escritos com linguagens de marcação como HTML e XML podem ser manuseados usando essa interface (sendo possível usar a maioria das linguagens de programação) que oferece métodos e propriedades distintos, para cada tipo de documento, que permitem acesso tanto aos seus elementos quanto ao seu conteúdo.

2 - JSON e YAML

Ao final deste módulo, você será capaz de descrever os formatos de transmissão de dados JSON e YAML.

Visão geral

Por muitos anos, XML foi o principal formato para armazenamento e transmissão de dados na Internet. Entretanto, após a sua criação, novos formatos foram e continuam sendo criados. Neste estudo, serão descritos dois entre os mais utilizados — **JSON** e **YAML**. Com isso, após conhecer cada um desses três formatos, é esperado que você tenha os subsídios necessários para escolher, frente a cada cenário, qual formato utilizar no desenvolvimento de sistemas web.

JSON

O formato JSON

JSON, acrônimo para **Javascript Object Notation**, é uma sintaxe para armazenamento e troca de dados. Trata-se de um formato de texto escrito com notação de objeto Javascript (W3C, 2020). Com JSON, é possível representar dados de forma estruturada e transmiti-los na web, enviando e recebendo dados de servidores remotos e exibindo-os em páginas HTML ou em aplicativos, por exemplo.

A data de criação desse formato remete ao início dos anos 2000. Em 2001, ele foi apresentado pela primeira vez no site json.org. A especificação mais atual do JSON é a ECMA-404. Tal especificação define um pequeno conjunto de regras para a representação estruturada de dados e seu principal objetivo é definir a sintaxe de um documento JSON válido.

A sintaxe JSON

A sintaxe JSON é composta por duas estruturas:

- Coleção de pares nome: valor, estrutura que, nas linguagens de programação, pode ser representada por objetos, dicionário, hash table, array associativo, entre outros.
- Lista ordenada de valores: nas linguagens de programação, pode ser representada como um array, vetor, lista, sequência, entre outros.

Logo a seguir, é apresentado o JSON correspondente ao XML usado previamente. Após o código, cada elemento JSON será descrito.

Json

O código acima também pode ser chamado de texto JSON. Os dados após as chaves “agencia”, “tipo”, “nome” e “numero” são os valores JSON. O par chave: valor — como “agencia”: “0123-4” — é chamado de objeto JSON. A chave “conta” é um array. Vejamos esses elementos em detalhes:

Texto JSON

Um texto JSON é uma sequência de tokens formados a partir de código Unicode, em conformidade com a gramática JSON. Tal conjunto de tokens possui seis tokens estruturais: **[** (colchetes aberto), **]** (colchetes fechado), **{** (chaves aberta), **}** (chaves fechada), **:** (dois pontos) e **,** (vígula).

Valores JSON

Um valor JSON pode ser: um objeto, um array, um número, uma string, true, false, null.

A estrutura de um objeto JSON é representada como um par de chaves "**{**" e "**}**" que engloba nenhum ou alguns pares de nome/valor, no qual o nome é uma **string**, seguido de dois pontos (**:**) que separam o nome do valor. Esse par "nome:valor" pode ou não ser procedido de um ou mais pares nome/valor, devendo esses serem separados por vírgula.

Dica

A sintaxe JSON não determina nenhuma restrição relacionada às strings utilizadas como nome, assim como não exige que os nomes sejam exclusivos. Além disso, a especificação JSON não define nenhum significado para a ordenação dos pares "nome:valor".

Arrays em JSON

Um array, em JSON, é uma estrutura representada por colchetes que englobam nenhum ou alguns conjuntos de pares "nome:valor", que deverão ser separados por vírgulas. Além disso, a sintaxe JSON não

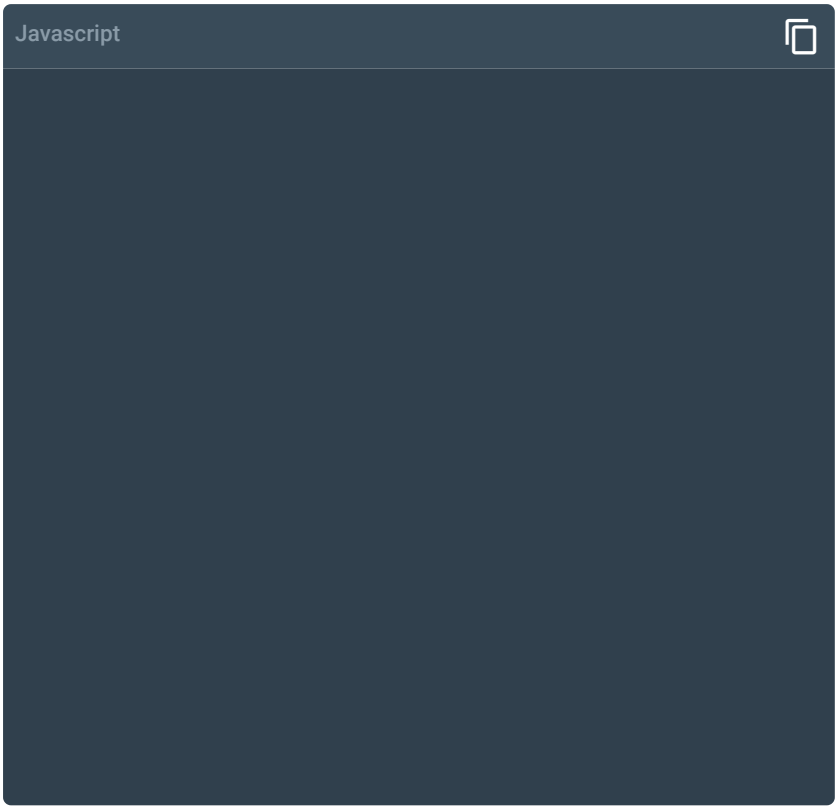
prevê nenhuma forma para ordenar os valores do array. Como mencionado no exemplo anterior, a chave “conta” é um array.

JSON na prática

Vejamos, analisando alguns códigos, como enviar, receber e armazenar dados em JSON, utilizando a linguagem Javascript.

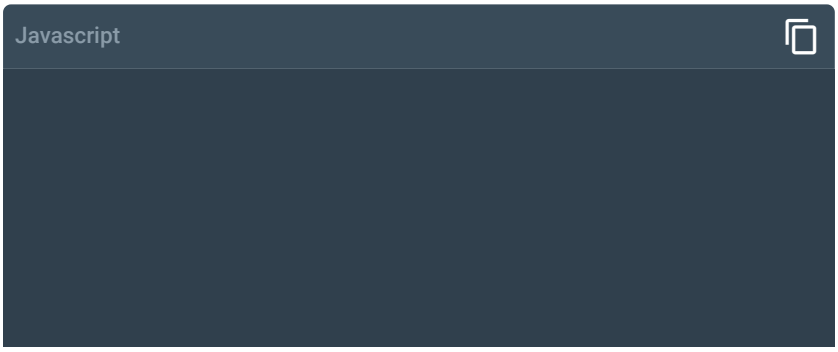
Enviando dados

Neste exemplo, é criado um objeto Javascript, que depois é convertido em texto JSON para ser enviado para um servidor remoto, onde poderá ser processado por uma linguagem server side, como Java, PHP etc.



Recebendo dados

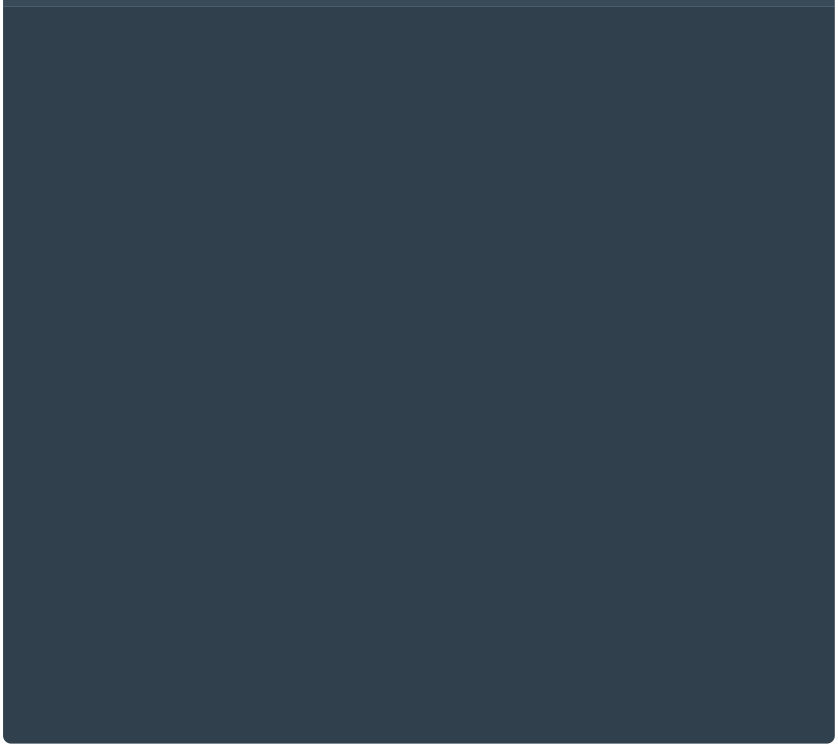
Neste exemplo, será criado um texto JSON, que será convertido em um objeto Javascript e então atribuído a um elemento HTML. Este exemplo simula a situação em que, através de uma requisição AJAX, ou outro tipo de requisição, recebemos como retorno um texto JSON e precisamos manipulá-lo para acessar e utilizar seus dados.



Armazenando dados

O exemplo, a seguir, demonstra como armazenar dados no formato JSON. Para isso, será utilizado um objeto Javascript que será convertido em texto JSON e, a seguir, armazenado. Além disso, também será demonstrado como recuperar os dados armazenados.

Javascript



Para visualizar os dados armazenados no navegador com o método “localStorage”, após inserir o código acima em uma página HTML e executá-lo, com a página aberta no navegador, abra o inspecionador de elementos, navegue até a aba “Application” e, no menu à esquerda, opção “Storage”, vá até “Local Storage”. Nesse item, será possível ver as chaves e os respectivos valores armazenados (em nosso caso, a chave será “stringJSON”). Para recuperar os dados apresentados, armazenados com “localStorage”, utilize o código a seguir:

Javascript

YAML

YAML, acrônimo original para "*Yet Another Markup Language*", e atualmente um acrônimo recursivo para "YAML *Ain´t Mark-up Language*", como o próprio nome diz, não é uma linguagem de marcação. Trata-se de uma linguagem para serialização e transmissão de dados cujo formato é amigável para humanos e de fácil entendimento para máquinas, podendo ser usada com a maioria das linguagens de programação. Assim como XML e JSON, essa linguagem é usada para a transmissão de dados na Internet. Em comparação com os outros dois formatos, sua sintaxe é parecida, em termos de estrutura, com JSON. É comum vermos esse formato sendo utilizado como arquivo de configuração ou arquivo de logs, embora não fique limitada a essas funções.

A sintaxe YAML

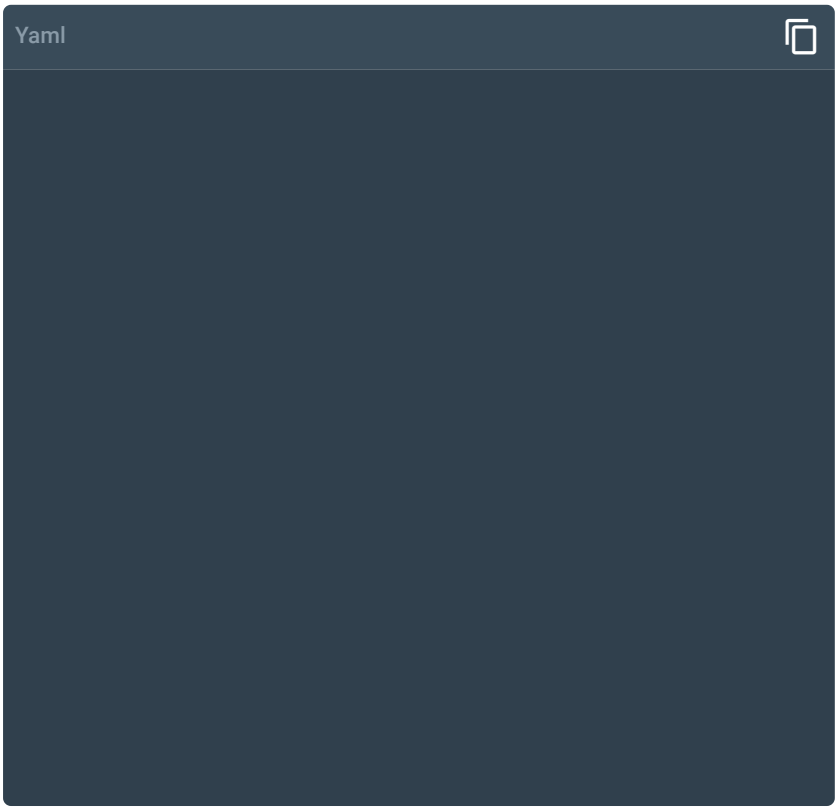
Em termos de estrutura e sintaxe, a YAML tem por característica usar poucos caracteres estruturais a fim de permitir que os dados sejam exibidos de forma natural. Nesse sentido, a sintaxe de um arquivo YAML é composta por:

1. Recuo/tabulação, usado como estrutura.
2. Sinal de dois pontos (":"), usado como separador do par "chave: valor".

3. Travessão, usado para a criação de listas de marcadores.

O formato YAML

A seguir, podemos ver, como YAML, a mesma estrutura vista como XML e, posteriormente, como JSON:



A fim de destacar as diferenças de estrutura, a seguir podem ser vistas as notações XML, JSON e YAML que representam a estrutura de dados que armazena dados de correntistas e que vem sendo utilizada como exemplo ao longo deste estudo.

XML

```
▼<correntistas>
  ▼<conta agencia="0123-4">
     <tipo>física</tipo>
     <nome>José Maria</nome>
     <numero>000.000-00</numero>
   </conta>
  ▼<conta agencia="5678-9">
     <tipo>jurídica</tipo>
     <nome>José Maria SA</nome>
     <numero>111.111-11</numero>
   </conta>
 </correntistas>
```

Primeira parte do quadro comparativo entre XML, JSON E YAML.

JSON

```
{
        "correntistas": {
                "conta": [
                        {
                                "agencia": "0123-4",
                                "tipo": "física",
                                "nome": "José Maria",
                                "numero": "000.000-00"
                        },
                        {
                                "agencia": "5678-9",
                                "tipo": "jurídica",
                                "nome": "José Maria SA",
                                "numero": "111.111-11"
                        }
                ]
        }
}
```

Segunda parte do quadro comparativo entre XML, JSON E YAML.

YAML

```
correntistas:
  conta:
  - agencia: 0123-4
    tipo: física
    nome: José Maria
    numero: 000.000-00
  - agencia: 5678-9
    tipo: jurídica
    nome: José Maria SA
    numero: 111.111-11
```

Terceira parte do quadro comparativo entre XML, JSON E YAML.

Como podemos ver, YAML utiliza menos tokens, ou sinais, em sua estrutura, se comparada a XML e JSON — da mesma forma que JSON o faz em relação a XML. Tal minimalismo faz com que YAML seja bastante leve. Além disso, ainda no âmbito de comparação com os outros formatos vistos, o formato YAML privilegia a leitura e compreensão por humanos, em detrimento da interpretação por linguagens de programação. Nesse ponto, tem comportamento inverso ao do JSON, que, ao custo de ser menos legível por humanos, é mais fácil de ser gerado e interpretado por linguagens de programação.

YAML na prática

Vejamos, de forma prática, como trabalhar com dados armazenados com o formato YAML. No primeiro código, será demonstrado como realizar a leitura de um arquivo YAML utilizando a linguagem Javascript e, no segundo, como gerar um arquivo YAML a partir de Javascript.

Atenção!

Repare que, em ambos os exemplos, faz-se necessária a utilização de uma biblioteca de parser (a Standalone JavaScript YAML 1.1 Parser & Encoder, disponível em https://github.com/jeremyfa/yaml.js). Essa necessidade se deve ao fato de que o processo de interpretação do formato em questão, por linguagem de programação, é mais complexo que o de outros formatos (como XML e JSON.).

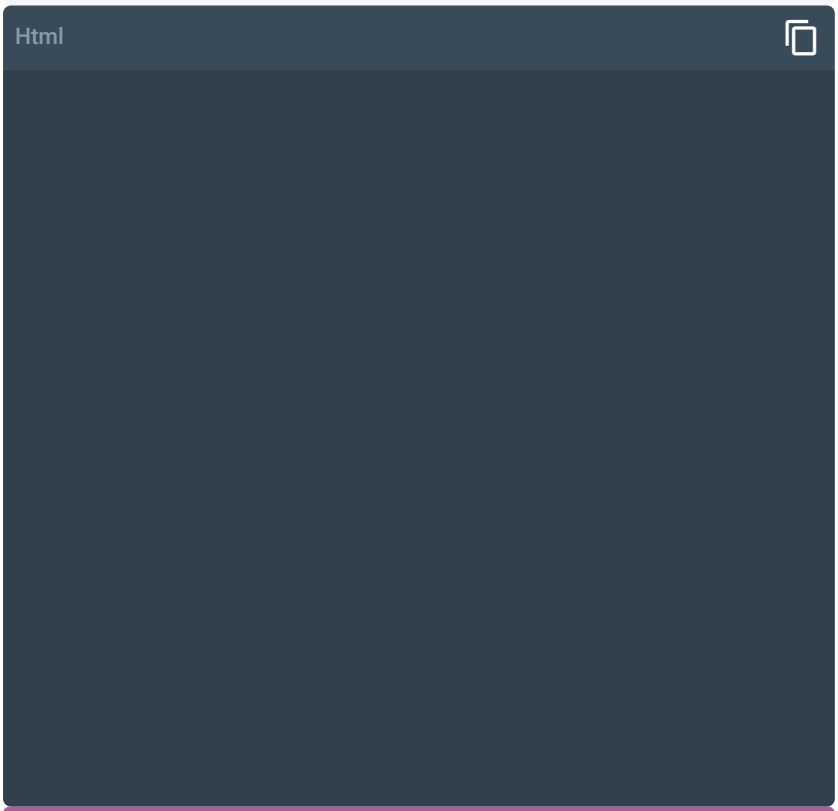
Html

Para executar o código, salve-o em uma pasta, em um servidor web, com o arquivo da biblioteca/parser e com o arquivo ‘clientes.yml’, cujo conteúdo pode ser visto na figura que compara XML, JSON E YAML, vista anteriormente.

Dica

Leia com atenção os comentários inseridos no código, pois explicam o que acontece linha a linha.

Html



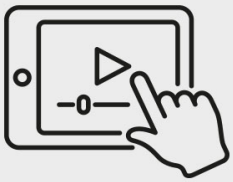
Atenção!

Todos os arquivos utilizados nesses exemplos podem ser encontrados na seção “Preparação”, dentro do arquivo compactado: “Códigos-fonte dos exercícios_Tecnologias de Transmissão de Dados em Sistemas Web”.

JSON e YAML na prática

Assista agora ao vídeo sobre os arquivos JSON e YAML.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Como vimos, se compararmos o processo de leitura e geração de conteúdo entre os três formatos descritos, veremos que o mais complexo deles diz respeito ao YAML. Além disso, a linguagem utilizada nos exemplos foi a Javascript. Entretanto, para cada linguagem de programação, o grau de dificuldade para manuseio desses formatos pode variar. Logo, não existe um formato melhor do que o outro. Na verdade, o melhor formato é o que melhor se adequa, em primeiro lugar, às necessidades de cada projeto e, em segundo lugar, às linguagens de programação utilizadas.

Falta pouco para atingir seus objetivos.

Vamos praticar alguns conceitos?

Questão 1

Dentre as afirmativas abaixo, uma não descreve a sintaxe e anatomia de uma string JSON. Assinale a alternativa em questão.

A — Tags: assim como em XML ou HTML, um arquivo JSON é composto por tags para nomear um atributo ou propriedade. Tal tag é seguida pelo sinal de dois pontos (“:”) e pelo seu valor.

B — Uma string JSON possui alguns tokens estruturais, como “[“, “{“, “:” e “,”.

C — Os tokens “[“ e “]” são usados para definir/englobar os dados de um array.

D — É possível ter, em um documento JSON, um par “chave:valor” cujo valor seja nulo (null).

E	Um objeto JSON tem a sua estrutura representada como um par de chaves “{“ e “}”.

Parabéns! A alternativa A está correta.

No formato JSON, diferentemente de linguagens de marcação, como XML, não temos tags para etiquetar os nossos dados. Em seu lugar, é usada uma estrutura composta por uma coleção de pares “nome:valor”, sendo o nome a etiqueta do dado, seguido pelo sinal de dois pontos “:” e pelo seu valor.

Questão 2

Assinale a afirmativa que corresponda à sintaxe de um documento no formato YAML.

A	No formato YAML, sendo esse derivado do formato JSON, estão disponíveis os mesmos tokens/sinais para a criação de um documento, como: “{“ , “[“ e “,”.
B	O formato YAML, como seu nome diz, é uma linguagem de marcação. Entretanto, um documento YAML pode ou não fazer uso de tags, como as vistas em documentos XML.
C	A sintaxe do formato YAML foi criada para privilegiar a sua leitura por máquinas/linguagens de programação. Logo, trata-se do formato mais fácil de ser interpretado por esses meios.
D	A estrutura de um documento escrito no formato YAML é caracterizada por utilizar poucos caracteres estruturais, fazendo com que os dados representados se pareçam com sua forma natural. Logo, um arquivo nesse formato é muito amigável a humanos.

E

O formato YAML é usado para a transmissão de dados na Internet, tal como as linguagens XML e JSON. A sintaxe YAML é similar, em termos de estrutura, com XML.

Parabéns! A alternativa D está correta.

O formato YAML se difere de outros formatos como XML e JSON por não fazer uso de muitos elementos estruturais para a definição e estruturação de seus dados. Tal característica, embora dificulte sua leitura através de linguagens de programação, dada a sua simplicidade, a torna, por outro lado, muito amigável à leitura de humanos.

3 - Os formatos de transmissão em requisições AJAX

Ao final deste módulo, você será capaz de compreender como utilizar dados nos formatos XML, JSON e YAML em requisições AJAX.

Visão geral

Neste módulo, todos os conceitos teóricos, além de parte dos exemplos práticos, serão consolidados por meio da demonstração de como utilizar, na prática, os formatos de transmissão de dados XML, JSON e YAML em requisições AJAX. Cada código será repetido, a fim de

evidenciar como trabalhar com essas tecnologias usando Javascript puro e com o framework Javascript jQuery.

XML e JSON em requisições AJAX

Inicialmente, veremos como utilizar dados armazenados no formado XML por meio de requisições AJAX. Embora aqui sejam representados por arquivos salvos com a extensão ".xml", tais arquivos poderiam ser substituídos por conteúdo gerado por linguagens de programação server side — desde que, claro, estejam no formato em questão. Para isso, troque o endereço do arquivo local para o do recurso em questão.

Atenção!

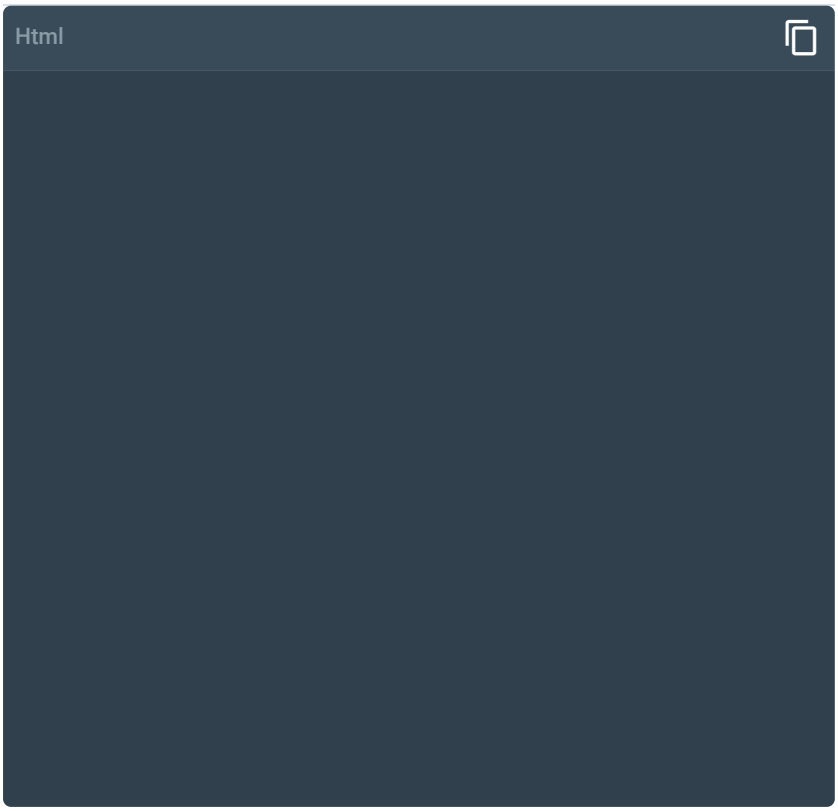
Demonstraremos, neste módulo, apenas o primeiro nó para entendimento da estrutura. Os demais dados estão no arquivo disponível para download.

XML e AJAX

Vamos ao código, começando com o arquivo XML, lido abaixo, pelo arquivo HTML e Javascript, que consumirá os dados do XML e os exibirá na página (baixe o arquivo na seção "Preparação").

Xml

Passo 01.



Passo 02.

Ao analisar o código apresentado previamente, perceba que é realizada uma verificação em relação ao tipo de nó, no ponto onde os nós filhos são coletados. Esse fragmento pode ser visto a seguir:

```
noFilho.nodeType === 1
```

Essa verificação é necessária, uma vez que os espaços em branco, no arquivo XML, também são considerados como nós. Veja uma parte dos nós filhos do elemento <Product> e repare que existem vários nós “#text” junto aos demais.

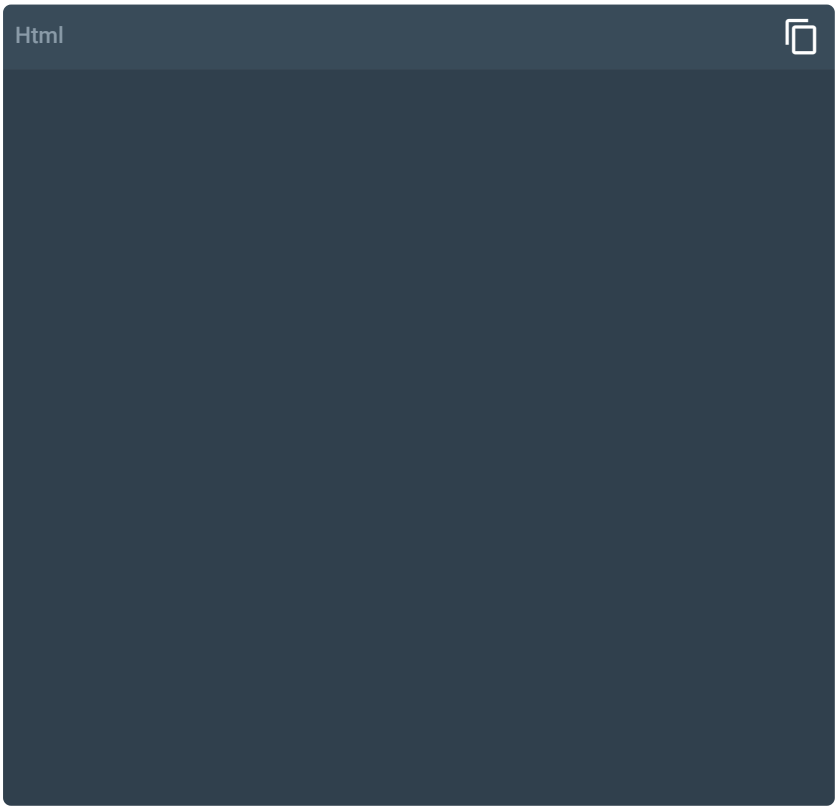
Fragmento dos nós do arquivo XML.

JSON e AJAX

Veremos agora como trabalhar com JSON e AJAX. Nesse contexto, o primeiro código contém um fragmento do arquivo JSON que será requisitado via AJAX. Vamos observar o código AJAX que recupera o conteúdo do arquivo JSON, o processa e o exibe, no formato de tabela, na página web.

Json

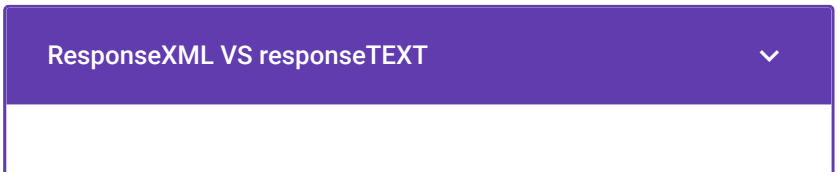
Como visto, estamos trabalhando com a mesma estrutura de dados usada no primeiro exemplo, em XML, ou seja, uma lista de produtos. A seguir, veja o código para coletar e processar essa estrutura.



A exemplo do que foi feito com o arquivo XML, o código acima trata os dados recebidos da requisição AJAX — aqui no formato texto e, posteriormente, convertido em objeto JSON — e atribui as informações a uma tabela HTML. Ao longo dos códigos, estão inseridos comentários explicando algumas de suas partes.

Principais diferenças no processamento de XML e JSON

Os códigos vistos tratam da requisição e manipulação dos dados nos formatos XML e JSON e contêm alguns comentários explicando cada passo do processo. Ainda assim, é importante destacar algumas diferenças referentes à manipulação desses dois formatos de arquivos. Vejamos quais são essas diferenças:



Por meio de requisições AJAX, é possível recuperar dados de recursos remotos em diferentes formatos, como XML, texto puro, HTML e JSON. Já no código Javascript — no objeto xmlHttpRequest —, há dois métodos disponíveis para tratar esses formatos: responseText e responseXML. Com o primeiro, tratamos os dados recebidos que não estejam no formato XML. Logo, com ele teremos os dados da resposta representados como texto. Daí a necessidade de, ao recuperar dados como JSON, realizar um parse desses dados a fim de poder manipulá-los como um objeto JS. Com isso, conseguimos acessar cada par “chave: valor” do arquivo JSON na notação de objeto.

Com o segundo método, responseXML, tratamos os dados no formato XML. Nesse caso, há métodos específicos para recuperar os dados por meio dos nós e valores do arquivo XML.

Dados como objetos JS

Quando estiver trabalhando com AJAX, e sempre que possível, converta o resultado obtido em objeto JS. Isso facilita o trabalho de interagir sobre os dados, pois teremos acesso diretamente aos métodos e objetos nativos da linguagem Javascript.

Uma dica para descobrir qual o formato dos dados recebidos é utilizar o método “console.dir” no resultado da requisição (seja responseXML seja responseText). Por meio desse método, podemos ver, no console do inspecionador de elementos, o formato desses dados. Veja, a seguir, o resultado desse método aplicado sobre o responseXML e responseText dos exemplos anteriores. Além disso, há também o resultado após ter sido realizado o “JSON.parser” sobre o responseText do segundo exemplo.

Primeira parte dos fragmentos de dados após aplicação do método “console.dir”.

```
{
   "Products": {
      "Product": [
         {
            "Product_ID": "7631",
            "SKU": "HEH-9133",
            "Name": "On Cloud Nine Pillow",
            "Product_URL": "https://www.domain.com/product/heh-9133",
            "Price": "24.99",
            "Retail_Price": "24.99",
            "Thumbnail_URL": "https://www.domain.com/images/heh-9133_600x600.png",
            "Search_Keywords": "lorem, ipsum, dolor, ...",
            "Description": "Sociosqu facilisis duis ...",
            "Category": "Home>Home Decor>Pillows|Back In Stock",
            "Category_ID": "298|511",
            "Brand": "FabDecor",
            "Child_SKU": "",
            "Child_Price": "",
            "Color": "White",
            "Color_Family": "White",
            "Color_Swatches": "",
            "Size": "",
            "Shoe_Size": "",
            "Pants_Size": "",
            "Occassion": "",
            "Season": "",
            "Badges": "",
            "Rating_Avg": "4.2",
            "Rating_Count": "8",
            "Inventory_Count": "21",
            "Date_Created": "2018-03-03 17:41:13"
         },
```


Segunda parte dos fragmentos de dados após aplicação do método “console.dir”.


```
▼Object
  ▼Products:
    ▼Product: Array(20)
      ▶0: {Product_ID: "7631", SKU: "HEH-9133", Name: "On Cloud Nine Pillow", Product_URL: "https://www.domain.com/product/heh-9133", Price: "24.99", …}
      ▶1: {Product_ID: "7615", SKU: "HEH-2245", Name: "Simply Sweet Blouse", Product_URL: "https://www.domain.com/product/heh-2245", Price: "42", …}
      ▶2: {Product_ID: "8100", SKU: "WKS-6016", Name: "Uptown Girl Blouse", Product_URL: "https://www.domain.com/product/wks-6016", Price: "58", …}
      ▶3: {Product_ID: "6489", SKU: "DKO-PROF", Name: "Knock Your Socks Off Lace-Up Heels", Product_URL: "https://www.domain.com/product/dko-prof", Price: "38", …}
      ▶4: {Product_ID: "7732", SKU: "HEH-2172", Name: "My Cup of Tea Sweater", Product_URL: "https://www.domain.com/product/heh-2172", Price: "68", …}
      ▶5: {Product_ID: "7609", SKU: "HEH-2211", Name: "Walk On Out Slip On Sneakers", Product_URL: "https://www.domain.com/product/heh-2211", Price: "34.99", …}
      ▶6: {Product_ID: "7675", SKU: "DKO-CAMEL", Name: "Harm Hearts Sweater", Product_URL: "https://www.domain.com/product/dko-camel", Price: "54.49", …}
      ▶7: {Product_ID: "7463", SKU: "WKS-5026", Name: "Silver Lining Dress", Product_URL: "https://www.domain.com/product/wks-5026", Price: "62", …}
      ▶8: {Product_ID: "7677", SKU: "PCH-8738", Name: "Follow The Beat Sneakers", Product_URL: "https://www.domain.com/product/pch-8738", Price: "32", …}
      ▶9: {Product_ID: "8099", SKU: "PCH-8475", Name: "Cup of Joe Pillow", Product_URL: "https://www.domain.com/product/pch-8475", Price: "36", …}
      ▶10: {Product_ID: "7425", SKU: "BCO-SK101", Name: "Burst Your Bubble Denim Jacket", Product_URL: "https://www.domain.com/product/bco-sk101", Price: "84", …}
      ▶11: {Product_ID: "8102", SKU: "HEH-2254", Name: "Walk It Out Heels", Product_URL: "https://www.domain.com/product/heh-2254", Price: "32", …}
      ▶12: {Product_ID: "9964", SKU: "BCO-2208", Name: "Word To The Wise Journal", Product_URL: "https://www.domain.com/product/bco-2208", Price: "14.95", …}
      ▶13: {Product_ID: "10440", SKU: "KOI-721", Name: "Basic Beauty Off-The-Shoulder Dress", Product_URL: "https://www.domain.com/product/koi-721", Price: "52", …}
      ▶14: {Product_ID: "6060", SKU: "VBH-V5102", Name: "Sunset Boulevard Pants", Product_URL: "https://www.domain.com/product/vbh-v5102", Price: "44", …}
      ▶15: {Product_ID: "10448", SKU: "SKE-15460", Name: "Across The Pond Boots", Product_URL: "https://www.domain.com/product/ske-15460", Price: "74.49", …}
      ▶16: {Product_ID: "8137", SKU: "PCH-8705", Name: "Once Upon A Time Lace Dress", Product_URL: "https://www.domain.com/product/pch-8705", Price: "50", …}
      ▶17: {Product_ID: "10018", SKU: "PGF-ESS", Name: "Lovey Dovey Maxi Dress", Product_URL: "https://www.domain.com/product/pgf-ess", Price: "68", …}
      ▶18: {Product_ID: "5670", SKU: "HEH-2223", Name: "Shot in the Dark Pillow", Product_URL: "https://www.domain.com/product/heh-2223", Price: "40", …}
      ▶19: {Product_ID: "9020", SKU: "PGF-RIK", Name: "Diamonds Are Forever Pillow", Product_URL: "https://www.domain.com/product/pgf-rik", Price: "36", …}
        length: 20
      ▶__proto__: Array(0)
    ▶__proto__: Object
  ▶__proto__: Object
```


Terceira parte dos fragmentos de dados após aplicação do método “console.dir”.

Repare na imagem que contém apenas parte dos resultados da aplicação do método “console.dir” sobre cada retorno e a diferença entre cada dado. Veja que, no formato XML, já na primeira linha, temos a informação “#document”. Em seguida, quando expandimos esse cabeçalho, temos disponíveis diversas propriedades, como “childNodes”, “firstChild”, entre outras.

Todas essas propriedades estão relacionadas especificamente ao formato em questão. Já na segunda, perceba que temos apenas um texto plano, na string JSON. Por fim, após aplicação do “JSON.parser” sobre a string JSON, na primeira linha temos o “Object”. Ao expandi-lo, temos a estrutura dos dados no formato de objetos e arrays.

YAML em requisições AJAX

YAML e AJAX

O trabalho com dados transferidos no formato YAML é semelhante ao que vimos em relação ao JSON, uma vez que os dados também serão transferidos no formato de texto puro e que, através do método responseText, teremos acesso a eles. Então, de posse dos dados no

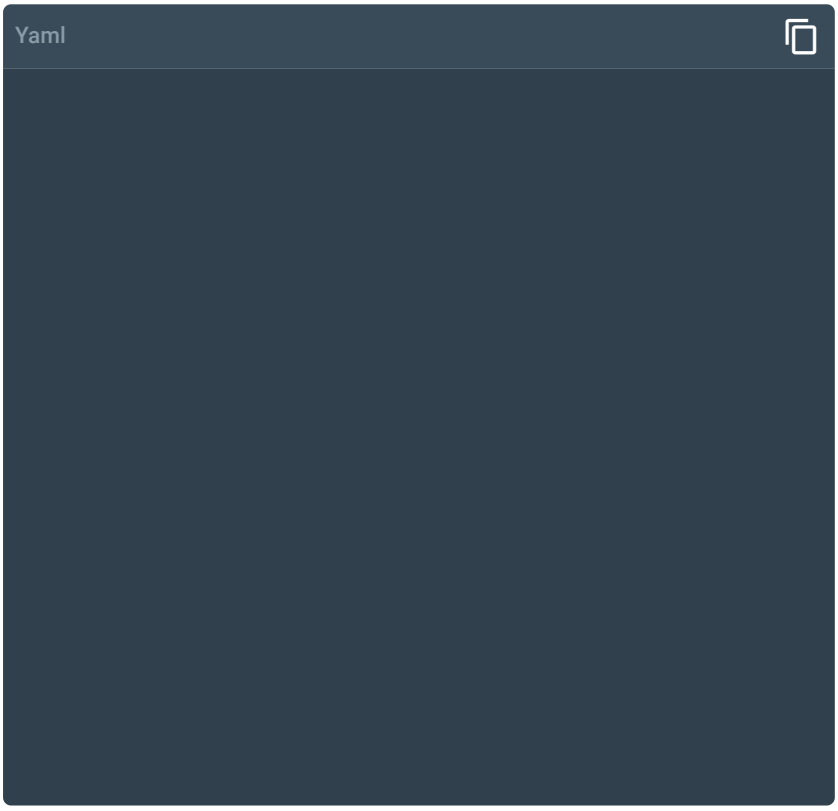
formato YAML, teremos, nesse caso, um passo adicional, que é o de transformá-los no formato de objeto JS/JSON.

Recomendação

Para essa tarefa, é recomendado utilizar um parser, uma biblioteca de terceiros — como vimos no módulo que tratou desse formato de dados. Após a conversão, todo o código restante será exatamente igual ao que utilizamos no último exemplo — por esse motivo, os passos relacionados à recuperação dos dados e exibição na página HTML não serão repetidos aqui.

Abaixo, podemos ver o fragmento do arquivo YAML contendo os dados dos produtos, trata-se do mesmo conteúdo dos exemplos anteriores.

Yaml



Formatos XML, JSON e YAML em requisições AJAX

Agora, assista ao vídeo sobre os formatos XML, JSON e YAML em AJAX.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.

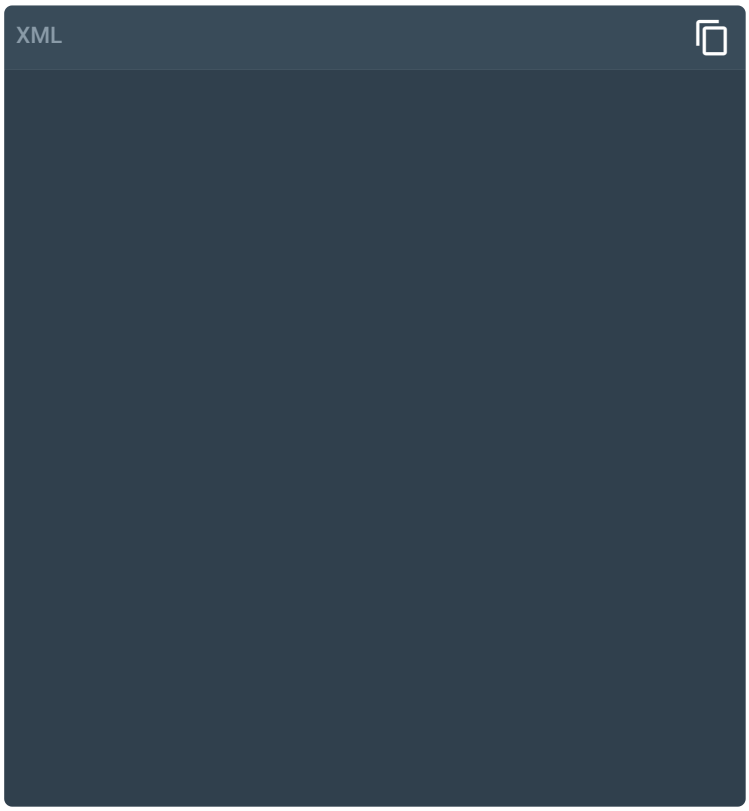
Falta pouco para atingir seus objetivos.

Vamos praticar alguns conceitos?

Questão 1

Assinale qual a afirmativa equivale à saída do programa abaixo, na instrução “alert”.

XML



A “multiples”

B “Tove”

C “undefined”

D “multiples” e “Tove”

E “Jani”

Parabéns! A alternativa B está correta.

Para manipular dados no formato XML, podemos fazer uso da XML DOM ou de outros recursos, como o XPATH, por exemplo. Em relação ao XML DOM, temos métodos e propriedades para tratar os

nós, atributos e seus respectivos dados. No fragmento acima, temos um nó e um atributo que possuem o mesmo nome, “to”. Entretanto, a forma de acessá-los é diferente: para o primeiro, temos o “getElementsByTagName”; já para o segundo, o “getAttribute”. Além disso, é necessário ter atenção para os nós do tipo “#text”, que representam os espaços em branco de um documento XML, sobretudo ao fazer uso de índices para acessar os nós do documento.

Questão 2

Uma requisição AJAX pode retornar dados em diferentes formatos. Nesse sentido, é correto afirmar que:

A

Dados de diferentes tipos podem ser tratados através dos métodos responseXML e responseText, sendo o primeiro responsável por tratar dados no formato XML e o segundo por tratar dados em formato de texto, devendo então serem realizados os devidos tratamentos sobre eles, visando sua manipulação.

B

Há vários métodos disponíveis para o tratamento de dados, conforme o seu formato, como o responseXML, o responseJSON, o responseText, o responseHTML, entre outros.

C

Em linguagens como o Javascript, todos os dados retornados a partir de uma requisição AJAX, independentemente do seu formato original, são convertidos automaticamente em objetos Javascript.

D

Não é possível, apenas utilizando recursos nativos da linguagem Javascript, manipular dados em diferentes formatos. Logo, para todo formato de dado, incluindo XML e JSON, é preciso utilizar bibliotecas de terceiros para realizar a sua transformação em um formato entendido por JS.

E

Os resultados obtidos por uma requisição AJAX podem ser transformados em objeto XML. Tal prática facilita a tarefa de manipulação dos dados obtidos.

Parabéns! A alternativa A está correta.

As requisições AJAX retornam dados em dois formatos: XML e Texto. Logo, qualquer formato que não seja o XML é visto como um texto puro. Nesse sentido, as devidas tratativas são necessárias a fim de converter esses dados em uma estrutura manipulável pela linguagem de programação utilizada, seja por meio de recursos nativos da própria linguagem, seja por meio de bibliotecas de terceiros.

Considerações finais

Neste estudo, descrevemos alguns dos principais formatos para transmissão de dados utilizados no desenvolvimento web — XML, JSON e YAML. Tais formatos foram apresentados de modo conceitual e prático, por meio de exemplos que demonstraram como recuperar e armazenar dados.

Assim, vimos como usar esses dados, os quais foram recuperados através de requisições AJAX para popular uma página HTML. Além disso, a fim de facilitar a compreensão entre as similaridades e as diferenças, apresentamos algumas distinções entre os formatos em questão.

Para encerrar, ouça sobre os principais pontos abordados neste estudo.

Para ouvir o *áudio*, acesse a versão online deste conteúdo.

Explore +

Para saber mais sobre os assuntos tratados neste estudo, pesquise na Internet:

Arrow Functions: A especificação Javascript ES6 apresentou alguns importantes novos recursos. Embora não possua suporte em todos os navegadores, como nas versões do Internet Explorer anteriores à Edge, tal especificação deve ser estudada a fundo. Entre as novidades, destacam-se as arrow functions. Veja como o guia de referência Javascript da Fundação Mozilla aborda as arrow functions, introduzindo uma série de conceitos e diversos exemplos.

Referências

Extensible Markup Language (XML). W3C. Consultado em meio eletrônico em: 20 ago. 2020.

XML DOM Tutorial. W3C schools. Consultado em meio eletrônico em: 20 ago. 2020.

JSON Introduction. W3C schools. Consultado em meio eletrônico em: 20 ago. 2020.

Material para download

Clique no botão abaixo para fazer o download do conteúdo completo em formato PDF.

Download material

O que você achou do conteúdo?

Relatar problema