Tecnologias de transmissão de dados em sistemas web

Prof. Alexandre de Oliveira Paixão

Apresentação

Você verá os conceitos de transmissão e consumo de dados em sistemas web, utilizando os formatos XML, JSON e YAML. Esses conceitos são fundamentais para a integração e transmissão de dados entre sistemas. Nesse sentido, trataremos a utilização de tais formatos de dados nos sistemas web.

Propósito

Preparação

Faça o download dos códigos e use editores de texto compatíveis com linguagens de marcação, como Notepad++ para Windows e Nano para Linux. Para visualizar os exemplos de código com AJAX, é necessário hospedar as páginas HTML em um servidor web. No Windows, o XAMP com o servidor web Apache é recomendado, ou você pode instalar o Apache ou o IIS separadamente. No Linux, os servidores Apache ou Nginx são recomendados.

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

Formatos de transmissão em requisições AJAX

Identificar a forma de utilização de dados nos formatos XML, JSON e YAML em requisições AJAX.

Introdução

Prepare-se para mergulhar nas tecnologias de transmissão de dados em sistemas web! Neste conteúdo, será abordada a linguagem de marcação XML e sua aplicabilidade, além dos formatos de transmissão de dados JSON e YAML. Você compreenderá como utilizar dados nos formatos XML, JSON e YAML em requisições AJAX, explorando estrutura, semelhanças e diferenças. Assista ao vídeo e aprenda a aproveitar essas tecnologias para criar aplicações web eficientes e interativas.

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

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.

O que é 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.

Neste vídeo, mergulhe na linguagem de marcação XML e sua aplicabilidade em sistemas web. Descubra como XML difere da HTML, sua estrutura de tags definidas pelo usuário e como criar um arquivo XML bem formatado.

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

Linguagem 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.

Atenção!

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

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 desse arquivo simples, começaremos a entender o seu formato.

XML

Em termos de sintaxe, observe os seguintes itens:

Primeira linha

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

Segunda linha

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

Terceira linha

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

Quarta linha

Nesta linha, temos o primeiro elemento filho do nó raiz, o elemento <pessoa>.

Quinta, sexta e sétima linhas

Nestas linhas, temos os elementos filhos do elemento <pessoa>.

Linhas seguintes

Nestas linhas, temos outro elemento <pessoa> e seus respectivos elementos filhos.

Charset

É o conjunto de caracteres utilizados para escrever o documento.

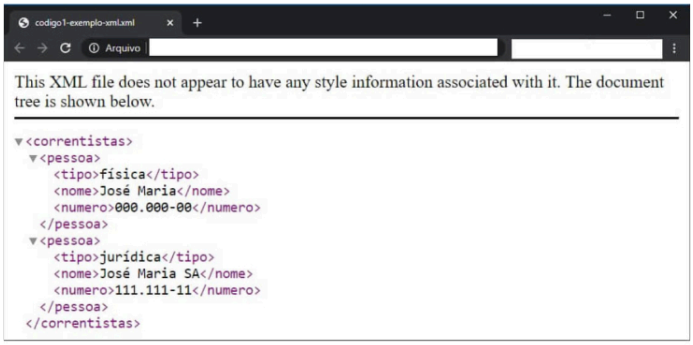
Em termos de sintaxe, observe os seguintes itens:

- 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

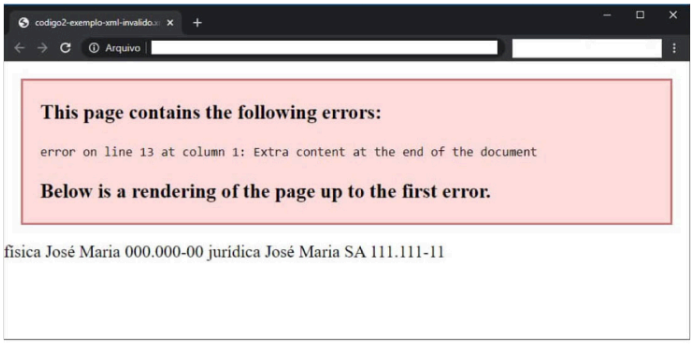
É o processo pelo qual se obtém o produto final de um processamento digital qualquer.



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.



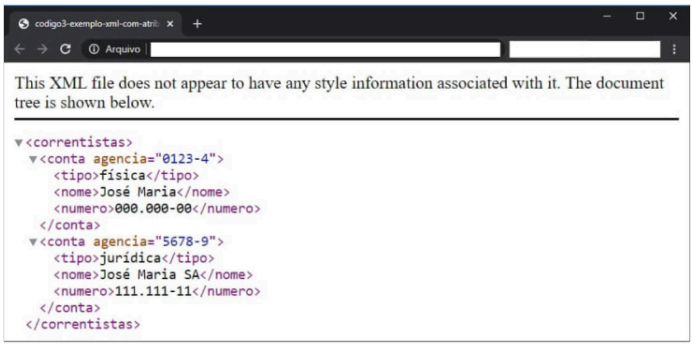
Renderização de XML mal formatado.

Utilizando atributos

Veja agora um 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 pelos seguintes elementos:

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 esse 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 por meio de DTD e Schema.

Convém também citar algumas desvantagens desse formato para a transmissão de dados por meio da internet em relação a outros disponíveis. Vamos conferir!

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 predefinida com regras de validação de um documento.

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 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, por meio 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.

XML

XML

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.

Propriedades e métodos do XML DOM

Vejamos as propriedades do XML DOM:

- nodeName
- nodeValue
- parenteNode
- childNodes

- attributes

E agora alguns dos métodos para o XML DOM 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, 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 caso precise deles.

Saiba mais

Para maior aprofundamento no assunto, é recomendado que você pesquise sobre a especificação W3C.

Atividade

Em relação ao XML DOM, assinale a afirmativa correta.

A

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

B

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.

C

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

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 C está correta.

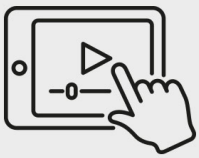
O Document Object Model (DOM) é uma interface de plataforma e linguagem neutra. Logo, documentos escritos com linguagens de marcação como HTML e XML podem ser manuseados por meio dessa interface (sendo possível usar a maioria das linguagens de programação) que oferece métodos e propriedades distintos,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

Criando o primeiro arquivo XML

Após conhecer o que é um arquivo XML e como ele pode ser utilizado, você agora criará um arquivo XML para transmissão de dados, conforme roteiro definido a seguir.

Neste vídeo prático, você aprenderá a criar o seu primeiro arquivo XML do zero! Passo a passo, iremos guiar você na criação de um documento XML válido, explicando a estrutura básica, a utilização de tags e atributos, e como organizar os dados. Você terá a oportunidade de colocar em prática todo o conhecimento adquirido sobre a linguagem de marcação XML, dando os primeiros passos na construção de arquivos XML para transmissão de dados em sistemas web. Não perca essa chance de se tornar expert na criação de XMLs!

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



Roteiro de prática

A prática, cujo roteiro será demonstrado logo a seguir, consistirá na criação e validação (em relação à estrutura) de um arquivo XML para transmissão de dados. Tanto os dados a serem inseridos como demais detalhes serão descritos na sequência.

Para realização do exercício, você precisará de:

- Um editor de textos – pode ser o próprio Bloco de Notas do Windows, o Nano Editor do Linux ou algo um pouquinho mais avançado, como o software (livre/gratuito) Notepad++,
- Um navegador para visualizar o XML – Google Chrome, Firefox, MS Edge etc.
- Uma ferramenta de validação de arquivos XMLs – existem diversas opões disponíveis on-line.

Vejamos agora o roteiro da nossa prática:

- No editor, crie um arquivo XML para transmissão de dados referentes a uma nota fiscal eletrônica.
- Em relação aos dados, a nota deverá ser composta por cinco seções:
    - Cabeçalho
    - Emitente
    - Destinatário
    - Item
    - Rodapé
- Cada uma das seções acima englobará dados, a saber:
    - Cabeçalho: UF, descrição da operação, número de série, número da nota fiscal e data/hora de emissão da nota.
    - Emitente: CNPJ, nome, nome fantasia e inscrição estadual.

    - Destinatário: CNPJ, nome, e-mail e endereço. Esta última será uma subseção, composta pelos seguintes dados: logradouro, número, bairro, município, UF, CEP, país e telefone.
    - Item: número do item (considerando que a nota poderá conter mais de um produto, cada item deverá ter um atributo chamado número, cujo valor é a ordenação da inclusão do produto na nota. Ex.: 1, 2, 3...), produto, valor e quantidade.
    - Rodapé: valor total e valor do ICMS.
- Após salvar o arquivo, com a extensão ".xml", visualize-o no navegador.
- Por fim, utilizando uma ferramenta on-line, valide o formato do arquivo, para garantir que tudo está certo.

Note a resolução do exercício no código seguinte:

Código resolução atividade

XML

Atividade

Um documento XML é formado por nós e atributos, sendo possível utilizar ambos para armazenar valores. Nesse contexto, analise o XML abaixo:

XML

A partir desse XML inicial, altere sua formatação para que (apenas) a informação “heading” seja um atributo e não um nó. A seguir, assinale a alternativa abaixo cujo conteúdo corresponde, em termos de sintaxe, à alteração solicitada.

XML

A

XML

XML

B

XML

XML

C

XML

XML

D

XML

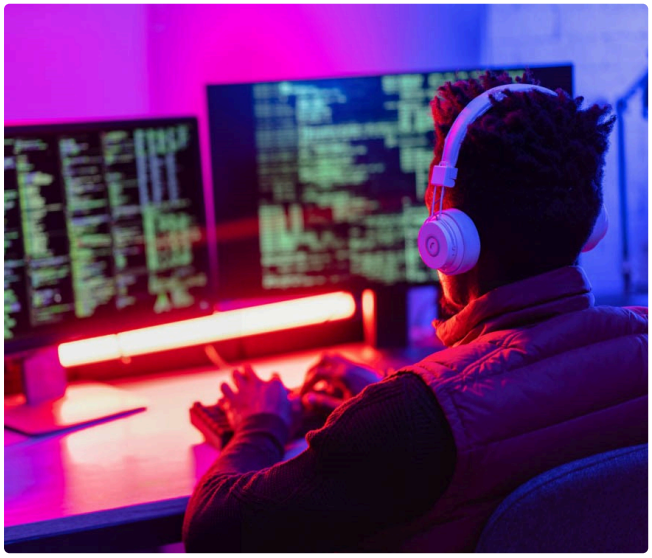
XML

E

XML

Parabéns! A alternativa C está correta.

O nó “heading” agora é um atributo do nó “body” e que mantém o seu valor original. As demais alternativas não atendem ao que foi solicitado, além de apresentarem problemas de sintaxe.



2 - JSON e YAML

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

JSON e YAML como formatos de transmissão de dados

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 formatos 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.

Neste vídeo, exploraremos os formatos de transmissão de dados JSON e YAML. Abordaremos a sintaxe e estrutura de cada formato, bem como

suas aplicações na transmissão de dados na web. Ao final, você terá o conhecimento necessário para escolher o formato adequado ao desenvolver sistemas web.

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

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).

Exemplo

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.

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. Essa 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.

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.

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

XML

O código visto também pode ser chamado de texto ou string 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. Esse conjunto de tokens possui seis tokens estruturais: [(colchetes aberto),] (colchetes fechado), { (chaves aberta), } (chaves fechada), : (dois pontos) e , (vírgula).

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.

XML

Recebendo dados

Neste exemplo, será criado um texto JSON, que será convertido em um objeto Javascript e então atribuído a um elemento HTML. Esse exemplo simula a situação em que, por meio 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.

XML

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, em seguida, armazenado. Além disso, também será demonstrado como recuperar os dados armazenados.

XML

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.

XML

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 em inglês, 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.

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:

- Recuo/tabulação, usado como estrutura.
- Sinal de dois pontos (":"), usado como separador do par "chave: valor".
- Travessão, usado para a criação de listas de marcadores.

Formato YAML

Podemos ver como YAML a mesma estrutura de dados já representada como XML e, posteriormente, como JSON:

XML

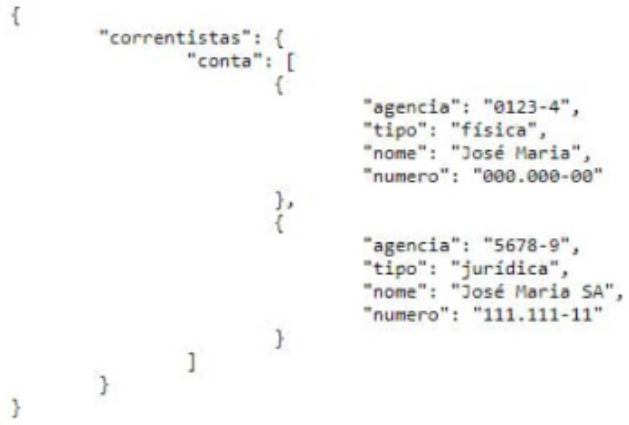
A fim de destacar as diferenças de estrutura, a seguir podem ser vistas as notações XML, JSON e YAML que representam a estrutura de dados

que armazena dados de correntistas e que vem sendo utilizada como exemplo ao longo deste estudo.

XML

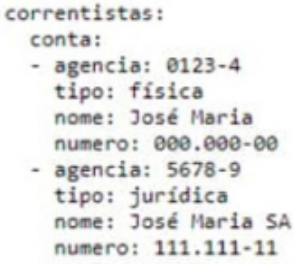
Primeira parte da comparação entre XML, JSON E YAML.

JSON



Segunda parte da comparação entre XML, JSON E YAML.

YAML



Terceira parte da comparação entre XML, JSON E YAML.

Como podemos ver, YAML utiliza menos tokens, ou sinais, em sua estrutura, se comparada a XML e JSON — assim como JSON o faz em relação a XML. Esse 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.

Repare que, em ambos os exemplos, faz-se necessária a utilização de uma biblioteca de parser (a Standalone JavaScript YAML 1.2 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.).

XML

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 imagem que compara XML, JSON E YAML, vista anteriormente.

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

XML

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”.

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 aquele que melhor se adequa, em primeiro lugar, às necessidades de cada projeto e, em segundo lugar, às linguagens de programação utilizadas.

Atividade

Como vimos, o formato JSON possui uma sintaxe própria, formada por algumas características, sendo a principal a representação de dados com a utilização de um par “chave:valor”. A seguir, são apresentados alguns exemplos de strings JSON. Assinale a alternativa cuja sintaxe representa uma string válida.

A [3.14, true,"Joe"]

B ["Value", 3.14, true, "name": "Joe"]

C {"value": $3.14, "name": "Joe"}

D {"name": "Joe", "age": 45, }

E {"name": "Joe", "lastname": "Dude"s"" }

Parabéns! A alternativa A está correta.

Na alternativa A, temos um array em formato Json, que representa uma string válida. Todas as demais possuem formato inválido, seja pela utilização de chaves em um array, como na opção B, seja pela utilização de sinal de moeda junto a valor, como na opção C, pela vírgula sobressalente ao final da string, visto na opção D, ou pela utilização de aspas, sem que estas sejam “escapadas”, dentro do valor de uma chave, como visto na opção E.

Criando o primeiro arquivo JSON

Assim como o XML, o JSON é um formato para transmissão de dados. Consequentemente, caso você precise realizar tal operação, integrando, por exemplo, diferentes sistemas, poderá escolher entre esses dois formatos – ou até mesmo utilizar o YAML. Com base na prática realizada anteriormente, a seguir, você deverá criar e validar um arquivo JSON para a transmissão de dados seguindo o roteiro definido.

Descubra, neste vídeo, como criar e validar um arquivo JSON eficiente para a transmissão de dados entre sistemas. Aprenda a estruturar corretamente os dados no arquivo JSON, seguindo um roteiro definido, e garanta uma integração perfeita e otimizada entre diferentes sistemas.

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

Roteiro de prática

Neste exercício, você deverá criar e validar um arquivo no formato JSON, conforme o roteiro definido adiante. Você precisará de um editor de textos – pode ser o próprio Bloco de Notas do Windows ou o Nano Editor do Linux, ou então algo um pouco mais avançado, como o software (livre/gratuito) Notepad++. Caso opte por utilizar o Notepad++, esse softwer contém plugins para a visualização e navegação em arquivos JSON. Logo, é recomendado que você utilize um dos plugins disponíveis para auxiliá-lo nessa tarefa. Por último, você precisará de uma ferramenta on-line de validação de arquivos JSON. Agora, vamos ao roteiro da prática?

- No editor, crie um arquivo JSON para transmissão de dados referentes a uma nota fiscal eletrônica.

- Em relação aos dados, a nota deverá ser composta por 5 “chaves-pai”:
    - Cabeçalho
    - Emitente
    - Destinatário
    - Item
    - Rodapé
- Cada uma das “chaves-pai” acima englobará dados, a saber:
    - Cabeçalho: UF, descrição da operação, número de série, número da nota fiscal e data/hora de emissão da nota.
    - Emitente: CNPJ, nome, nome fantasia e inscrição estadual.
    - Destinatário: CNPJ, nome, email e endereço. Esta última será uma subseção, composta pelos seguintes dados: logradouro, número, bairro, município, UF, CEP, país e telefone.
    - Item: número do item (considerando que a nota poderá conter mais de um produto, cada item deverá der um atributo chamado número, cujo valor é a ordenação da inclusão do produto na nota. Ex.: 1, 2, 3...), produto, valor e quantidade.
    - Rodapé: valor total e valor do ICMS.
- Após salvar o arquivo, com a extensão “.json”, visualize/navegue pelo mesmo utilizando o plugin do editor Notepad++.
- Por fim, utilizando uma ferramenta on-line, valide o formato do arquivo para garantir que tudo está certo.

Agora, veja o código dessa atividade!

Código resolução atividade

XML

Atividade

Como vimos, cada formato de transmissão de dados possui sua própria sintaxe. A seguir, apresentaremos um documento no formato YAML. Esse documento está inválido. Considerando as alternativas abaixo, selecione a opção que descreve o porquê de esse arquivo ser considerado inválido.

XML

A Um arquivo YAML não pode conter chaves escritas em letras maiúsculas.

B Um arquivo YAML não pode ter uma chave principal – no caso do exemplo, a chave “NFe”.

C Para ser vá lido, um arquivo YAML deverá conter uma chave principal aberta em seu início e fechada ao seu final. Logo, o documento acima ficou sem a tag de encerramento “NFe”.

D

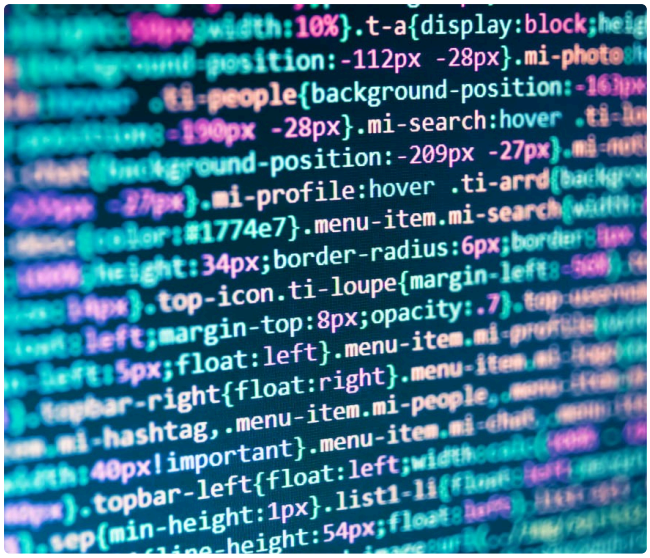
Uma das principais características, e também vantagem de um arquivo YAML, é o seu tamanho reduzido. Logo, para manter tal característica, arquivos nesse formato não podem ter mais de 10 chaves.

E

O arquivo está inválido pois as seções principais não estão terminadas com o símbolo ":" (dois pontos).

Parabéns! A alternativa E está correta.

Arquivos YAML possuem uma estrutura de chaves seguidas de valores ou de outras chaves (seguidas de valores). A separação entre chave e valor deve ser feita utilizando o símbolo ":" (dois pontos). Logo, o arquivo acima está incorreto, pois esse símbolo não está presente após as seções "pai".



3 – Formatos de transmissão em requisições AJAX

Ao final deste módulo, você será capaz de identificar a forma de utilização de dados nos formatos XML, JSON e YAML em requisições AJAX.

Utilizando os formatos de transmissão de dados com AJAX

Após conhecer alguns dos formatos de transmissão de dados, chegou a hora de utilizá-los na prática. Nesse sentido, veremos, ao longo do nosso estudo, como transmitir dados nos formatos XML, JSON e YAML em requisições AJAX.

Neste vídeo, você aprenderá sobre os formatos de transmissão de dados em requisições AJAX, com foco nos formatos XML, JSON e YAML. Serão apresentados exemplos de como utilizar e manipular

esses formatos, destacando suas diferenças e principais características.

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

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", esses 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 apenas o primeiro nó para entendimento da estrutura. Os demais dados estarão no arquivo disponível para download.

XML e AJAX

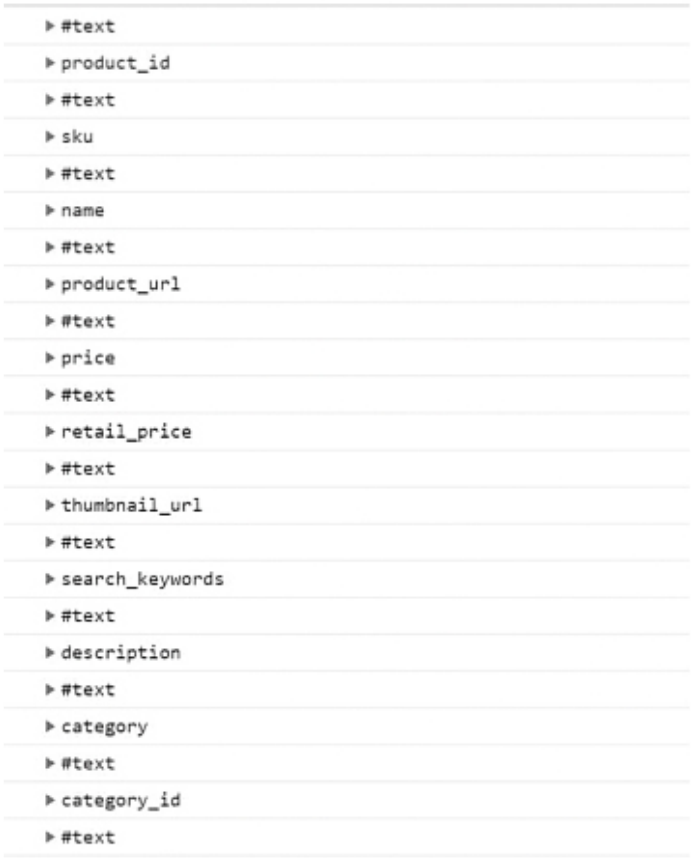
Vamos ao código, começando com o arquivo XML, lido a seguir pelo arquivo HTML e Javascript, que consumirá os dados do XML e os exibirá na página (baixe o arquivo na seção "Preparação").

XML

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. Confira o fragmento:

XML

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 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.

XML

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.

XML

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!

responseXML x responseTEXT

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 responseText, tratamos os dados recebidos que não estejam no formato XML. Logo, 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 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 ou responseText). Usando esse método, podemos ver, no console do inspecionador de elementos, o formato dos dados recebidos. 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.

Nas próximas imagens, é possível reparar onde apenas parte dos resultados da aplicação do método “console.dir” sobre cada retorno é apresentada e a diferença entre cada dado.

Nesta primeira imagem, veja a saída no formato XML, em que, 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.

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

Agora, na próxima imagem, temos apenas um texto plano na string JSON:

```
{
   "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”.

Por fim, observe na última imagem que 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.

```
ay(20)
t_ID: "7631", SKU: "HEH-9133", Name: "On Cloud Nine Pillow", Product_URL: "https://www.domain.com/product/heh-9133", Price: "24
t_ID: "7615", SKU: "HEH-2245", Name: "Simply Sweet Blouse", Product_URL: "https://www.domain.com/product/heh-2245", Price: "42
t_ID: "8100", SKU: "WKS-6016", Name: "Uptown Girl Blouse", Product_URL: "https://www.domain.com/product/wks-6016", Price: "58",
t_ID: "6489", SKU: "DKO-PROF", Name: "Knock Your Socks Off Lace-Up Heels", Product_URL: "https://www.domain.com/product/dko-pr
t_ID: "7732", SKU: "HEH-2172", Name: "My Cup of Tea Sweater", Product_URL: "https://www.domain.com/product/heh-2172", Price: "
t_ID: "7609", SKU: "HEH-2211", Name: "Walk On Out Slip On Sneakers", Product_URL: "https://www.domain.com/product/heh-2211", Pr
t_ID: "7675", SKU: "DKO-CAMEL", Name: "Warm Hearts Sweater", Product_URL: "https://www.domain.com/product/dko-camel", Price: "5
t_ID: "7463", SKU: "WKS-5026", Name: "Silver Lining Dress", Product_URL: "https://www.domain.com/product/wks-5026", Price: "62
t_ID: "7677", SKU: "PCH-8738", Name: "Follow The Beat Sneakers", Product_URL: "https://www.domain.com/product/pch-8738", Price:
t_ID: "8099", SKU: "PCH-8475", Name: "Cup of Joe Pillow", Product_URL: "https://www.domain.com/product/pch-8475", Price: "36",
ct_ID: "7425", SKU: "BCO-SK101", Name: "Burst Your Bubble Denim Jacket", Product_URL: "https://www.domain.com/product/bco-sk10
ct_ID: "8102", SKU: "HEH-2254", Name: "Walk It Out Heels", Product_URL: "https://www.domain.com/product/heh-2254", Price: "32",
ct_ID: "9964", SKU: "BCO-2208", Name: "Word To The Wise Journal", Product_URL: "https://www.domain.com/product/bco-2208", Pric
ct_ID: "10448", SKU: "KOI-721", Name: "Basic Beauty Off-The-Shoulder Dress", Product_URL: "https://www.domain.com/product/koi-7
ct_ID: "6060", SKU: "VBH-V5102", Name: "Sunset Boulevard Pants", Product_URL: "https://www.domain.com/product/vbh-v5102", Pric
ct_ID: "10448", SKU: "SKE-15460", Name: "Across The Pond Boots", Product_URL: "https://www.domain.com/product/ske-15460", Pric
ct_ID: "8137", SKU: "PCH-8705", Name: "Once Upon A Time Lace Dress", Product_URL: "https://www.domain.com/product/pch-8705", Pr
ct_ID: "10018", SKU: "PGF-ESS", Name: "Lovey Dovey Maxi Dress", Product_URL: "https://www.domain.com/product/pgf-ess", Price: "
ct_ID: "5670", SKU: "HEH-2223", Name: "Shot in the Dark Pillow", Product_URL: "https://www.domain.com/product/heh-2223", Price:
ct_ID: "9020", SKU: "PGF-RIK", Name: "Diamonds Are Forever Pillow", Product_URL: "https://www.domain.com/product/pgf-rik", Pric

 Array(0)
bject
ject
```


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

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, com o 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, utilize um parser, uma biblioteca de terceiros. 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.

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

XML

Atividade

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

A

Dados de diferentes tipos podem ser tratados por meio 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 ser realizados os devidos tratamentos, visando à 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. Essa prática facilita a tarefa de manipulação dos dados obtidos.

Parabéns! A alternativa A está correta.

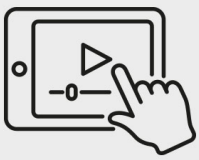
As requisições AJAX retornam dados em dois formatos: XML e Texto. Logo, qualquer formato que não o XML é visto como um texto puro. Nesse sentido, as devidas tratativas são necessárias a fim de converter esses dados em uma estrutura manipulável pela linguagem de programação utilizada, seja por meio de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

Consumindo dados no formato JSON com AJAX

Atualmente, o formato mais utilizado para a transmissão de dados no ambiente web é o JSON. Para consumir dados JSON em requisições AJAX, precisaremos recorrer aos conceitos já vistos sobre sua sintaxe e também a outros conhecimentos de programação – em especial HTML e Javascript. Nesse contexto, ao longo dessa prática, usaremos cada uma dessas tecnologias para requisitar, consumir, manipular e exibir os dados provenientes de uma API REST em uma página HTML.

Neste vídeo, exploraremos o formato JSON e como utilizá-lo em requisições AJAX. Aprenderemos a consumir dados JSON, manipulá-los e exibi-los em uma página HTML, utilizando conceitos de programação, como HTML, JavaScript e API REST.

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

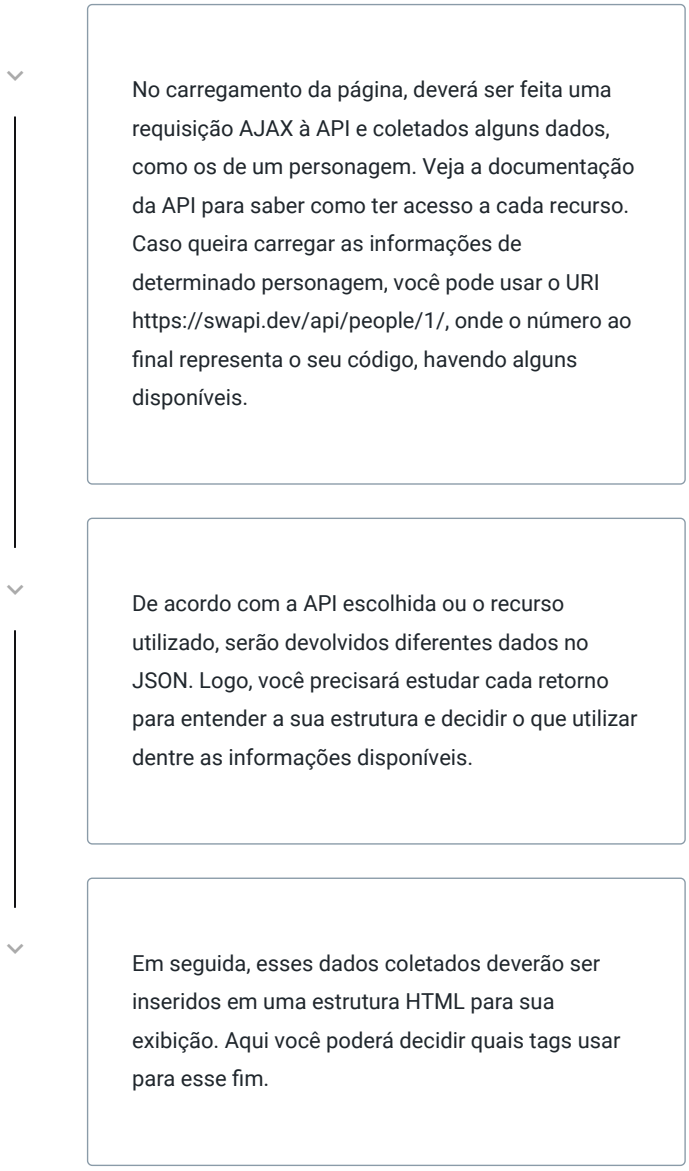


Roteiro de prática

Para realização dessa prática, você precisará de alguns recursos. Vejamos!

- Inicialmente, será necessário um editor para construir seu código-fonte. Você poderá usar o próprio Bloco de Notas do Windows ou o Nano Editor do Linux, ou então algo um pouco mais avançado, como o software (livre/gratuito) Notepad++.
- Também será necessário um navegador web, como o Google Chrome, Firefox, MS Edge etc., para exibir a página HTML que será codificada.
- Por fim, precisará do endereço de uma API REST que seja gratuita e retorne dados em formato JSON. Sugerimos a SWAPI (The Star Wars API), disponível no endereço https://swapi.dev/. Entretanto, em caso de indisponibilidade desse recurso ou por alguma preferência pessoal, você poderá utilizar qualquer outra API (lembrando apenas que o formato devolvido por ela deverá ser JSON).

O fluxo da página HTML será o seguinte:



Agora, vejamos o roteiro propriamente dito:

- No editor, crie o esqueleto de uma página HTML básica.

- Dentro da tag body, crie uma div e atribua um id para a mesma. Essa div será usada para exibir os dados obtidos, por meio do Javascript.
- Dentro do arquivo, na seção header, crie o código JS para realizar a requisição AJAX, via xmlHttpRequest, da API (informada acima).
- Ainda no código JS, crie uma função para receber o resultado da requisição AJAX.
- Na função criada, você deverá utilizar o objeto contendo a resposta (responseText) para ter acesso ao JSON.
- Faça o parse do JSON em um objeto Javascript para facilitar o acesso aos dados recebidos.
- Estando com acesso, via objeto JS, a cada dado recebido, utilize-os para montar a estrutura HTML necessária para exibi-los, de acordo com sua preferência.
- Após salvar o arquivo, com a extensão ".html", visualize a página utilizando o navegador de sua preferência.
- Tendo ocorrido tudo certo, você deverá ver na página os dados coletados, organizados na estrutura HTML que você montou.
- Caso nenhum dado seja exibido, cheque a aba "console", no inspecionador de elementos, no navegador, e veja se ocorreu algum erro. Caso não, revise todo o seu código.

A resolução da prática acima poderá ser feita de diferentes formas. Veja, a seguir, uma das formas possíveis.

Veja aqui o código

XML

Atividade

A estrutura de um arquivo XML é composta por nós “pais” e nós “filhos”. Logo, é importante entender essa sintaxe para manipular os dados transmitidos por meio desse formato. Nesse sentido, no fragmento de código abaixo, é feito o acesso a um dos dados constantes em um fragmento de XML. Assinale qual a afirmativa, entre as opções abaixo, equivale à saída do código, 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. Em relação ao XML DOM, temos métodos e propriedades para tratar os nós, atributos e seus

respectivos dados. No fragmento visto, 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.

O que você aprendeu neste conteúdo?

- Formatos de transmissão de dados em sistemas web
- Diferentes formatos de dados em requisições AJAX
- XML
- JSON
- YAML

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, essa 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.

API REST: As APIs estão intimamente ligadas aos conceitos vistos em nosso estudo. Para saber mais a seu respeito, recomendamos as seguintes leituras:

- O tutorial **JSON with Ajax**, publicado no site REST API Tutorial em 17 de agosto de 2021.
- O site do JSON API.
- O artigo **O que é API REST**?, publicado no site Red Hato em 8 de maio de 2020.

Referências

EXTENSIBLE Markup Language (XML). W3C schools. Consultado na internet em: 20 ago. 2020.

JSON Introduction. W3C schools. Consultado na internet em: 20 ago. 2020.

XML DOM Tutorial. W3C schools. Consultado na internet em: 20 ago. 2020.