- TADS – Análise e desenvolvimento de sistemas

- PW - Programação Web

- ADO 01 – DOM

- Aluno: CHARLES APARECIDO DA SILVA MOREIRA

- Professor: Carlos Veríssimo

Conceitos

DOM significa Modelo de Objetos de Documentos e tem finalidade de descrever e padronizar os objetos e seus inter-relacionamentos em uma aplicação.

O DOM subdivide-se em três subconjuntos: DOM core (core significa coração e é o termo consagrado no jargão técnico para designar as funções gerais de uma Interface de Programação para Aplicativos), DOM HTML e DOM XML.

DOM é uma API criada com a finalidade de auxiliar o desenvolvimento de aplicações de natureza geral.

Uma API é um conjunto padronizado de funções, rotinas, métodos, classes, protocolos e procedimentos gerais destinados a fornecer funcionabilidades a serem usadas por softwares ou programas com a finalidade de simplificar a tarefa de desenvolvimento.

Podemos dizer que o DOM é uma API que padroniza a estrutura de documentos HTML e XML, simplificando a tarefa de se acessar e manipular tais documentos. O DOM fornece aos programadores maneiras simples de acessar a estrutura, criar, modificar, adicionar, retirar e manipular elementos e conteúdos de documentos HTML e XML.

O dom é uma especificação do W3C (World Wide Web Consortium), isto é, trata-se de uma interface padronizada que pode ser usada em qualquer tipo de ambiente e aplicação. O DOM foi desenvolvido para ser usado por qualquer linguagem de programação.

A definição do DOM segundo o W3C é: “O DOM – Document Object Model do W3C é uma interface independente de plataforma e linguagem que permite aos programas e scripts acessar e atualizar dinamicamente a estrutura, o conteúdo e a estilização de documentos”.

O Document Object Model ou simplesmente DOM é utilizado pelo navegador Web para representar a sua página Web. Quando altera-se esse modelo com o uso do Javascript altera-se também a página Web.

O DOM HTML é uma representação da estrutura do documento HTML. O diagrama representativo do DOM é do tipo árvore, tal como o diagrama representativo de uma família no qual são representados os graus de parentesco, ascendência e descendência entre seus membros. O termo Modelo de Objetos, escolhido para designar o DOM, tem o mesmo sentido que na sua definição tradicional em Programação Orientada a Objetos.

O Objeto document representa um documento aberto no navegador. Assim, toda vez que se abre um documento HTML ou XML em um navegador, é criado um objeto

document. O objeto document é um objeto do objeto window e pode ser acessado com o uso da sintaxe window. document. O objeto document permite que se acesse, via Javascript, todos os elementos HTML de uma página, (X)HTML ou XML.

A função init() não é um método próprio de nenhum objeto Javascript nem mesmo um nome de função reservado da linguagem. É uma função cujo uso em scripts está consagrado, mas pode-se nomeá-la com um nome de livre escolha. Nada mais é do que uma função que contém várias funções a serem executadas somente quando o documento tiver sido carregado.

Manipular o DOM com Javascript exige que o documento esteja totalmente carregado antes do carregamento script, ou seja, o script precisa conhecer o DOM para poder manipulá-lo.

Se você insere o script na seção head do documento, ele será carregado antes da seção body, que contém elementos HTML da página. Assim não reconhecerá o DOM e falhará.

Uma solução seria colocar o script antes do carregamento do DOM, porém usar o evento onload do objeto window para instruir a execução da função que contém o script somente após o carregamento do DOM. Como o script seria inserido na seção head do documento, poderá ser colocado em um arquivo externo e linkado nessa seção. Porém, apresenta a grande desvantagem de admitir apenas uma função para execução após o carregamento da página.

Segue abaixo algumas propriedades que podem ser utilizadas com o objeto document:

Propriedade documentElement: Captura o elemento raiz <html> de um documento HTML.

Propriedade getElementById: Busca um elemento da página Web com o uso do atributo id do elemento.

Propriedade getElementsByTagName: Retorna um array dos elementos com o mesmo nome.

O DOM core é uma Application Programming Interface (API), que define um conjunto de objetos e interfaces para acessar e manipular os objetos de um documento.

As funcionabilidades dessa API permitem aos desenvolvedores de software e autores de scripts acessar e manipular conteúdos HTML e XML.

O DOM core representa os documentos como uma hierarquia de objetos denominados de nós. Os nós por sua vez, podem implementar interfaces próprias. Alguns tipos de nós podem conter nós-filhos de diferentes tipos. Outros tipos de nós são últimos na cadeia hierárquica e não podem conter nós-filhos.

O DOM ainda especifica ainda duas interfaces denominadas nodeList e NameNodeMap destinadas a manipular uma lista de nós.

A interface nodeList refere-se a uma lista de ordenada de nós, como o conjunto de nós-filhos de um determinando nó ou os conjuntos dos elementos HTML retornados pelo método getElementsByTagName() da interface Element.

A interface NameNodeMap refere-se a uma lista não ordenada de nós, como o conjunto dos elementos HTML retornados pelo método getElementsByName() da interface Element ou a coleção de elementos cujo determinado atributo tenha um determinado valor.

Os objetos de ambas as interfaces NodeList e NameNodeMap são dinâmicos, isto é, qualquer modificação no DOM, seja acrescentando, seja suprimindo nós, é refletida instantaneamente na lista.

Importante citarmos o DOMException, que é uma interface destinada a tratar exceções ou erros. Quando uma operação qualquer no DOM se torna impossível de ser realizada, diz-se que ocorreu uma exceção ou erro.

As exceções são definidas por um número inteiro indicativo do erro ocorrido. O conjunto dos números indicativos de erro denomina-se ExceptionCode ou código de exceções. Cada número de erro tem a ele associada uma constante Javascript.

A manipulação de erros em Javascript se faz com a declaração try-catch-finally, embora a manipulação de erros do DOM apresente pouca utilidade prática no desenvolvimento diário de aplicações Javascript, devemos sempre ficar em alerta da sua existência, caso necessite dela em aplicações avançadas ou particulares.

Podemos determinar que a aplicação DOM é essencial para o uso no desenvolvimento web, e que sem ele teríamos uma enorme dificuldade de acessar os elementos na nossa página.

Caso prático

```
<!--
            SENAC - TADS - Programação Web
               Aula #02 - Introdução ao JavaScript
               Objetivos deste código: Demonstrar JS em Páginas html
                   "Interação com Usuário - Fazendo Cálculos"
        -->


        <!DOCTYPE html>
        <html lang="pt-br">
        <head>
            <meta charset="UTF-8">
            <meta name="viewport" content="width=device-width, initial-scale=1.0">
            <title>Números com JS</title>
            <style>
                body { font: 12pt Arial; }
                button { font-size: 12pt; padding: 30px; }
            </style>
        </head>
        <body>
             <h1>Senac - TADS - PW - 2º Semestre </h1>
             <h2>Aula #02 - Introdução ao JS</h2>
               <h3>Fazendo Cálculos:Soma de Números</h3>
            <button onclick="somarNumeros()">Iniciar a soma</button>
            <section id="res">
                <p>O resultado aparecerá aqui...</p>
            </section>

            <script>

                function somarNumeros() {
                    let n1 = Number(window.prompt('Por favor, Digite um número: '))
                    let n2 = Number(window.prompt('Por favor, Digite outro
        número:'))
                    soma = n1 + n2

                    let res = document.querySelector('section#res')
                    res.innerHTML = `<p>A soma entre <mark>${n1}</mark> e
        <mark>${n2}</mark> é igual a <strong>${soma}</strong>!</p>`
                }
            </script>
        </body>
        </html>
```

Nesse exemplo de caso prático de interação com usuário para somar, conseguimos identificar algumas interações entre o Javascript e HTML, dentro do script temos uma function somarNumeros(), onde temos duas variáveis n1 e n2, onde através do comando window.prompt vamos receber os valores para a soma. Como os valores recebidos são retirados de uma caixa de texto, devemos utilizar um Number antes do window para converter para números os valores que estão na caixa de texto.

É criado uma variável soma para receber os valores de n1 + n2, e em seguida é criado outra variável nomeada res que através do document.querySelector buscamos a section que está com id res.

Por fim apresentamos o resultado diretamente no HTML através do res.innerHTML com uma String interpolada e com soma em negrito através do .

Referencial Bibliográfico

SILVA, Maurício Samy. JavaScript guia do Programador. São Paulo: Novatec Editora, 2010.