Tratamento de Dados

O comando window.prompt que vai coletar o dado para armazenar dentro da variável vai sempre retornar um valor do tipo string, mesmo que coloque um número, ele vai tratar como string, então, se tentarmos somar dois números coletados pelo prompt sem antes convertê-los, teremos apenas os dois valores concatenados, para o + funcionar como adição será necessário antes fazer a conversão.

Para converter uma string para number em JavaScript:

Number.parseInt(n) para converter n para inteiro.

Number.parseFloat(n) para converter n para real.

Number(n) vai converter n para o tipo Number, e o próprio programa identifica se é int ou float.

Para converter um number para string em JavaScript:

String(n)

n.toString()

Formatando Strings:

var s = 'JavaScript'

Se for utilizado o comando 'Eu estou aprendendo s', o s nesse caso não aparecerá como JavaScript, mas somente como s, uma forma de fazer a variável aparecer é com concatenação: 'Eu estou aprendendo ' + s, que resultará em 'Eu estou aprendendo JavaScript'.

Porém, no caso da ocorrência de muitas variáveis, a concatenação se torna algo muito trabalhoso, para contornar isso usamos template string. `Eu estou aprendendo ${s}`. O cifrão seguido de chaves é chamado de placeholder, dessa forma é mais fácil formatar a String. (Tem que ser com a crase invertida pra funcionar)

Outros comandos com Strings:

s.length - diz quantos caracteres a string tem

s.toUpperCase() - deixa a string em maiúsculo

s.toLowerCase() - deixa a string em minúsculo

Comandos novos na aula

document.write('Isso vai escrever na tela ao invés de mostrar um pop up como o alert')

n.toFixed(2) Esse comando vai pegar o número real n e vai mostrar ele com duas casas após a vírgula, a quantidade de casas é especificada pelo número dentro dos parênteses.

n.toFixed(2).replace('.', ',') Aqui, além de mostrar com duas casas decimais, ele vai substituir o ponto no número decimal por vírgula.

n.toLocaleString('pt-BR', {style: 'currency', currency: 'BRL'}) Formata o número para dinheiro, BRL mostra em reais R$, USD mostra em dólares. Isso é localização

```
<script>
      var nome = window.prompt('Qual é o seu nome?')
      document.write(`Olá,  <strong>${nome}</strong>!  Seu  nome  tem
${nome.length} letras<br>`)
      document.write(`Seu       nome       em       maiúsculo       é
${nome.toUpperCase()}<br>`)
      document.write(`Seu nome em minúsculo é ${nome.toLowerCase()}`)
</script>
```

*document.write() vai escrever a mensagem no corpo do HTML

Operadores

O JavaScript possui várias famílias de operadores, vamos falar dos seguintes operadores: aritméticos, atribuição, relacionais, lógicos e ternário.

Aritméticos

Operadores usados para fazer cálculo: +, -, *, /, %, **. Esses são operadores binários, pois precisam de dois valores.

5 + 2 = 7 5 * 2 = 10 5 % 2 = 1 (resto da divisão inteira)
5 - 2 = 7 5 / 2 = 2.5 5 ** 2 = 25

É necessário ficar atento à precedência de operadores, como na matemática normal, multiplicação e divisão se resolvem antes de soma e subtração. O uso dos parênteses muda a ordem de precedência, como na matemática normal.

Ordem de precedência: ( ) → ** → *, /, % → +, -

```
> var a = 5 + 3
undefined
> var b = a % 5
undefined
> var c = 5 * b ** 2
undefined
> var d = 10 - a / 2
undefined
> var e = 6 * 2 / d
undefined
> var f = b % e + 4 / e
undefined
> a
8
> b
3
> c
```

45
> d
6
> e
2
> f
3

Incremento
var x = 5
x = x + 1 ou x+=1 é equivalente a x++, ++x também funciona
x = x - 1 ou x-=1 é equivalente a x--, --x também funciona

Operadores Relacionais

> - Maior que	<= - Menor ou igual	=== - Sinal de identidade
< - Menor que	== - Igual	!== - Desigual restrito
>= - Maior ou igual	!= - Diferente	

5 == 5 True
5 == '5' True
5 === '5' False

Operadores Lógicos
! - Negação
&& - Conjunção (and)
|| - Disjunção (or)

Ordem de execução: ! → && → ||

Precedência
Operadores aritméticos () ** / … → Operadores relacionais < > >=... → Operadores Lógicos

Operador Ternário
? : dentro de uma mesma expressão
teste ? true : false

média >= 7.0 ? "Aprovado" : "Reprovado"
Se a média for maior ou igual a 7, o programa retorna Aprovado, se não, ele retorna Reprovado.

var x = 8
var res = x % 2 == 0 ? 5 : 9
Aqui, se a variável x dividida por 2 tiver resto 0, a variável res recebe 5, se não, a variável res recebe 9.

Entendendo o DOM

DOM (Document Object Model) é um modelo de objetos para documentos, um conjunto de objetos dentro do navegador que dá acesso aos documentos internos no website.

Árvore DOM

A árvore DOM começa da raiz, e essa raiz dentro do navegador é chamada de window, e tudo dentro do JavaScript está dentro desse objeto chamado window (a janela do navegador é esse objeto DOM chamado window). Dentro do window tem vários outros objetos, por ex: location (diz a localização do seu site- URL, página atual, página anterior), document (documento atual), history (guarda de onde você veio, pra onde vai…). Dentro do document existe o objeto html, dentro de html tem dois objetos (child), o head e o body. Head e body são child (filhos) de html, e html é um parent (parente, pai, mãe) de head e body, quem está abaixo é child, quem está imediatamente acima é parent. Dentro de head tem as tags que colocamos no nosso documento, e dentro de body temos as os objetos que estão lá (h1, p, div, etc).

Selecionando elementos para navegar dentro da árvore DOM
Você pode selecionar:

por Marca - getElementByTagName()

```
<h1>Iniciando estudos com DOM</h1>
    <p>Aqui vai o resultado</p>
    <p>Aprendendo a usar <strong>DOM</strong> em JavaScript</p>
    <div>Clique em mim</div>

    <script>
        var p1 = window.document.getElementsByTagName('p')[0]
        window.document.write(p1.innerText)
    </script>
```

Pega mais de um elemento caso haja mais de um elemento com a mesma TagName, a seleção de qual elemento será é feita dentro dos colchetes, como um fatiamento, começando do zero, na ordem de cima para baixo.

Também é possível manipular o texto através desse script, como no exemplo abaixo, que o script pega o que está dentro da primeira tag p e manda mudar a cor do texto para azul:

```
<script>
      var p1 = window.document.getElementsByTagName('p')[0]
      p1.style.color = 'Blue'
</script>
```

Algo parecido é a possibilidade de fazer uma mudança no body, usando a mesma lógica dentro do script:

```
<script>
      var corpo = window.document.body
      corpo.style.background = 'Black'
</script>
```

Voltando para o getElementByTagName, no exemplo a seguir o script vai pegar o que está dentro do segundo parágrafo <p>, porém se tentarmos exibir o que foi resgatado apenas com o innerText, o texto vem sem a formatação do HTML, ou seja, a tag na palavra 'DOM' não será considerado, para resgatar o elemento com a formatação usamos innerHTML.

```
<script>
      var corpo = window.document.body
      var p1 = window.document.getElementsByTagName('p')[1]
      document.write(p1.innerHTML)
</script>
```

por ID - getElementById()

```
<div id="msg">Clique em mim</div>

<script>
      var d = window.document.getElementById('msg')
      d.style.background = 'Green' // Aqui deixa o background da
div verde
      d.innerText = 'Estou aguardando' // Aqui muda o texto dentro
da div
</script>
```

É possível também usar esses comandos resumidos em um só:

```
<script>
      window.document.getElementById('msg').innerText =  'Olá'
      // Isso vai apenas mudar o texto dentro da div msg para Olá
</script>
```

por Nome - getElementsByName()

```
<div name="msg">Clique em mim</div>

<script>
      var d = window.document.getElementsByName('msg')[0]
      d.innerText = 'Olá'
</script>
```

por Classe - getElementsByClassName()

```
<div class="msg">Clique em mim</div>

<script>
      window.document.getElementsByClassName('msg')[0].innerText =
'Olá'
</script>
```

por Seletor

É uma forma nova de selecionar

querySelector()

querySelectorAll()

Quando vamos selecionar uma ID

```
<div id="msg">Clique em mim</div>

<script>
      var d = window.document.querySelector('div#msg')
      d.style.color = 'Blue'
</script>
```

Quando vamos selecionar uma Classe

```
<div class="msg">Clique em mim</div>

<script>
      var d = window.document.querySelector('div.msg')
      d.style.color = 'Blue'
</script>
```

Eventos DOM

Imagine uma div dentro de uma página HTML, um evento é tudo que pode acontecer com essa div, como, por exemplo, o mais comum que é eventos de mouse.

mouseenter: evento que dispara quando o cursor do mouse está dentro da div

mousemove: enquanto o mouse se mover dentro da div ele dispara os eventos do mousemove

mousedown: enquanto clica e segura o botão do mouse, os eventos de mousedown são disparados

mouseup: dispara outro evento no momento que o botão do mouse para de ser pressionado

click: dispara o evento quando o mouse faz um clique comum

mouseout: quando move o mouse para fora da div o evento é disparado

A lista de eventos de JavaScript é muito extensa, e pode ser encontrada na internet, pesquise por "JavaScript DOM events list" e procure a página "Event reference" da documentação da Mozilla.

Antes de disparar e tratar um evento é necessário entender o que é uma função.

Funções

Uma função em JavaScript é um conjunto de códigos que serão executados somente quando o evento ocorrer.

Por exemplo:

- Programei 10 linhas.
  - Essas 10 linhas a gente chama de bloco.
  - Essas 10 linhas não serão executadas automaticamente.
  - Os comandos desse bloco só ocorrerão, por exemplo, quando eu clicar dentro da div. As 10 linhas serão disparadas somente quando o evento ocorrer.
  - Para que essas 10 linhas sejam executadas somente quando o evento ocorrer, o primeiro passo é colocá-las dentro de um bloco. Um bloco em JavaScript é delimitado entre os sinais de chaves "{}".
  - Esse bloco precisa ser nomeado como uma function: function{ código }. No JavaScript uma função sem nome é chamada de anônima, mas para o método funcionar, é preciso que a função tenha um nome, e os nomes das funções são ações que a gente pode fazer: function ação(){ código }, opcionalmente, dentro dos parênteses também podem ser colocados alguns parâmetros.

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
                <meta    name="viewport"    content="width=device-width,
initial-scale=1.0">
    <title>Eventos DOM</title>
    <style>
        div#area{
            font: normal 20pt Arial;
```

```
            background: rgb(5, 202, 5);
            color: white;
            width: 200px;
            height: 200px;
            line-height: 200px;
            text-align: center;

        }
    </style>
</head>
<body>
          <div   id="area"   onclick="clicar()"   onmouseenter="entrar()"
onmouseout="sair()">
        interaja...
    </div>

    <script>
        var a = document.getElementById('area')
        function clicar() {
            a.innerText = 'Clicou!'
            a.style.background = 'blue'
        }
        function entrar() {
            a.innerText = 'Entrou'
            a.style.background = 'red'
        }
        function sair() {
            a.innerText = 'Saiu'
            a.style.background = 'rgb(5, 202, 5)';
        }
    </script>
</body>
</html>
```

Este exemplo, quando executado, mostra uma div que muda a cor e o texto quando o mouse entra, clica e sai.

Se quiser deixar o html mais limpo, é possível adicionar event listeners no JavaScript que irá substituir os eventos no html:

```
<div id="area">
        interaja...
    </div>
```

```
    <script>
        var a = document.getElementById('area')
        a.addEventListener('click', clicar)
        a.addEventListener('mouseenter', entrar)
        a.addEventListener('mouseout', sair)
        function clicar() {
            a.innerText = 'Clicou!'
            a.style.background = 'blue'
        }
        function entrar() {
            a.innerText = 'Entrou'
            a.style.background = 'red'
        }
        function sair() {
            a.innerText = 'Saiu'
            a.style.background = 'rgb(5, 202, 5)';
        }
    </script>
```

Calculadora com JavaScript

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
                <meta    name="viewport"    content="width=device-width,
initial-scale=1.0">
    <title>Somando Números</title>
    <style>
        body {
            font: normal 18pt Arial;
        }
        input {
            font: normal 18pt Arial;
            width: 100px;
        }
        div#res {
            margin-top: 20px;
        }
    </style>
```

```
</head>
<body>
    <h1>Somando Valores</h1>
    <input type="number" name="txtn1" id="txtn1"> +
    <input type="number" name="txtn2" id="txtn2">
    <input type="button" value="Somar" onclick="somar()">
    <div id="res">Resultado</div>
    <script>
        function somar(){
            var tn1 = window.document.getElementById('txtn1').value
            var tn2 = window.document.getElementById('txtn2').value
            var res = window.document.getElementById('res')
            var n1 = Number(tn1)
            var n2 = Number(tn2)
            var s = n1 + n2

                res.innerHTML = `A soma entre ${n1} e ${n2} é igual a
<strong>${s}</strong>`
        }
    </script>
</body>
</html>
```