MÓDULO V

Lógica de Programação utilizando a Linguagem JavaScript

MÓDULO V

JavaScript: convertendo String para número

Em **JavaScript**, assim como na maioria das linguagens de programação, podemos representar valores numéricos como **números** mesmo, ou como **strings**.

Geralmente usamos a representação como string quando é necessário exibir resultados ou informações para o usuário, e não precisamos realizar cálculos com os valores numéricos.

Porém, para ser capaz de realizar cálculos numéricos com os valores (por exemplo, advindos de um formulário), é necessário que esses valores estejam representados no formato adequado, ou seja, que sejam do tipo numérico. Caso contrário, as operações aritméticas podem falhar: uma soma pode ser interpretada como uma concatenação de strings, por exemplo.

MÓDULO V

Converter string para inteiro: Método parseInt()

Podemos converter uma string em um valor numérico inteiro empregando o método **parseInt()**.

Sintaxe:

```
parseInt(numero, base)
```

MÓDULO V

- numero é o valor de string que deve ser convertido em tipo numérico (deve ser um número que está sendo representado como string; caracteres de texto não podem ser convertidos).
- base é a base numérica a utilizar. O padrão é 10, caso nenhuma base seja informada.

O método retorna **NaN** (Not a Number) se o valor não puder ser convertido em inteiro.

Exemplo:

```
var entrada = '30';
var numInteiro = parseInt(entrada, 10);
//ou simplesmente
var numInteiro = parseInt(entrada, 10);
```

MÓDULO V

Converter string para decimal: Método parseFloat()

Podemos também converter uma string em um valor numérico de ponto flutuante empregando o método **parseFloat()**.

Sintaxe:

```
parseFloat(numero)
```

MÓDULO V

- numero é o valor de string que deve ser convertido em tipo numérico

(caracteres de texto não podem ser convertidos).

Exemplo:

```
var entrada = '3.1415';
var pi = parseFlot(entrada);
```

MÓDULO V

Converter strings para números: o objeto Number()

O objeto **Number()** também pode ser empregado para realizar a conversão de uma string em valor numérico.
O número retornado pode ser um inteiro ou um número de ponto flutuante.

Se uma string de texto for passada para este método, será retornado o valor "**NaN**", que significa "**Not a Number**" (*Não é um Número*).

Sintaxe:

```
Number(valor)
```

MÓDULO V

- valor é a string que deve ser convertida em tipo numérico (caracteres de texto não podem ser convertidos).

O método Number também pode converter valores de data em números, retornando o número de milissegundos decorridos desde 01/01/1970 00:00:00 UTC.

Exemplo:

```
var entrada = '600';
var result = Number(entrada);
```

No geral, se você souber qual o tipo de número esperado, recomenda-se usar o método apropriado (**parseInt ou parseFloat**), em vez do objeto **Number**.

MÓDULO V

JavaScript: Template literals

É comum em aplicações que tenhamos que lidar com grandes volumes de texto, a exemplo das informações que podem estar contidas em uma página web. Geralmente, ao lidarmos esses dados em editores necessitamos quebrar linhas para tornar o código mais legível, o que não é possível quando declaramos strings com aspas simples ou duplas.

```
const texto = "Ciclo novo, novas caras. O técnico Tite fez jus à expectativa
    de novidades na seleção brasileira e divulgou a primeira convocação depois
    ..."
```

Note que no código acima as aspas só são fechadas na linha 3, o que faz com que o script seja encerrado com um erro:

```
(function (exports, require, module, __filename, __dirname) {
const texto = "Ciclo novo, novas caras. O técnico Tite fez jus à expectativa

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

SyntaxError: Invalid or unexpected token
```

MÓDULO V

Com a sintaxe de **Template Literals** podemos substituir as aspas pelo caractere ` permitindo quebrar o texto em várias linhas:

```
const texto = 'Ciclo novo, novas caras. O técnico Tite fez jus à expectativa
    de novidades na seleção brasileira e divulgou a primeira convocação depois
    ...'
```

Template Literals também permite interpolar valores em um texto, sendo um recurso amplamente utilizado por programadores:

```
const produto = { id : 1, nome : 'Grampo', preco : 34.6 }

const descricao = `${produto.nome} por apenas R$ ${produto.preco}`

console.log(descricao) // Grampo por apenas R$ 34.6
```

MÓDULO V

Sintaxe

```
`Texto inicial ${[ valor ]} e texto final`
```

Exemplo 1

Até o ES5 quando era necessário quebrar uma string em múltiplas linhas devíamos utilizar o caractere **\n**:

```
const texto = "Amor é fogo que arde sem se ver\nÉ ferida que dói e não se sente"
```

Com Template Literals podemos reescrever o mesmo código da seguinte forma:

```
const texto = "Amor é fogo que arde sem se ver\nÉ ferida que dói e não se sente"
```

MÓDULO V

Exemplo 2

É possível interpolar strings e expressões em JavaScript:

```
const mensagem = "O valor do desconto é: " + (preco - (preco * desconto))
```

Com Template Literals podemos escrever essa sentença desta forma:

```
const mensagem = `O valor do desconto é: ${(preco - (preco * desconto))}`
```

MÓDULO V

Exemplo 4

Algumas vezes em aplicações web precisamos manipular características de elementos em uma página atribuindo ou removendo deles classes que alterem sua apresentação, após alguma interação do usuário, como o clique de um botão.

Apesar de trivial essa funcionalidade pode gerar código confusos e propensos a erros se necessitamos chegar alguma condição para a atribuição ou remoção da classe:

```
var resposivo = true

var classes = 'nav' + (resposivo ? ' collapsed' : '')
```

MÓDULO V

Com Template Literals esse mesmo código pode ser simplificado.

```
var responsivo = true

const classes = `nav ${responsivo ? 'collapsed' : ''}`
```

MÓDULO V

Exemplo 5

```
const contato = {
    nome : "Estevão Dias",
    login : "edrd"
}

const urlAtivacaoCadastro = "https://devmedia.com.br/cadastro/ativacao/"

const mensagem = `Parabéns, ${contato.nome}!

Agora você faz parte da maior comunidade de programadores do país \\o

Falta apenas ativar a sua conta, mas isso é bem rapidinho

Dá um pulo aqui $?usuario=${contato.login}&ativaocao=true`

console.log(mensagem)
```

Frequentemente necessitamos criar textos padrão em aplicações para envio de e-mails ou preenchimento de certas partes de uma página web. Com Template Literals podemos fazer isso com facilidade e utilizando constantes para garantir que apenas certas áreas do texto possam ser alteradas:

MÓDULO V

Tagged Template Literals

Um outro recurso útil é possibilidade de criar Tagged Template Literals a partir de Template Literals com o auxílio de funções:

```
function taggedLiteral(string){
    console.log(string)
}

taggedLiteral`Texto ou expressão` // [ "Texto ou expressão" ]
```

MÓDULO V

Podemos com isso separar o texto de uma expressão em Template Literals. Esse recurso pode ser útil quando desejamos processar as partes de um texto de independentemente, porém ele não foi recebido desta forma.

```
const template = (texto, ...expressao) => ({
  texto,
  expressao
});

const preco = 5000;
const desconto = 0.10;

const { texto, expressao } = template`O valor do desconto é: ${(preco * desconto)}`;

console.log(texto);
console.log(expressao);
```

MÓDULO V

Javascript **alert**, **confirm** e **prompt**: caixas de diálogo **Popup**

O que é JavaScript alert?

Antes de explicar o que é o **método alert()**, vamos falar rapidamente sobre o que é o ***Browser Object Model*** (BOM). Ao abrir uma página **HTML**, o navegador cria um conjunto de objetos que são organizados hierarquicamente. No topo dessa hierarquia está o objeto **window**, que controla toda a página e permite acessar os objetos filhos, além de propriedades e métodos associados a eles.

Os objetos relacionados ao **BOM** estão associados a recursos para a manipulação do navegador, como os objetos **screen**, **location**, **history** e outros, que são controlados pelo objeto **window**. O método **alert()**, portanto, faz parte desse conjunto e **é usado para permitir a interação com a pessoa usuária da página**.

MÓDULO V

O método **alert()** exibe uma mensagem no navegador por meio de uma caixa de diálogo, que nada mais é que uma pequena janela **popup**. Além do texto, também é exibido um botão de confirmação para indicar que a pessoa realmente leu a mensagem. Portanto, a janela só será fechada após o clique no botão de confirmação.

Por ter a característica de bloquear a navegação da pessoa usuária enquanto o botão confirmar não for clicado, esse recurso deve ser usado com moderação. Além disso, existem navegadores que oferecem a função de bloquear o uso dessa funcionalidade.

MÓDULO V

Qual é a sintaxe?

O método **alert()** é um dos tipos de caixas de diálogo disponíveis no navegador. Sua função é exibir o parâmetro mensagem, que pode ser fornecido como uma variável ou ter um valor constante definido no código. Existem outras duas maneiras possíveis de se comunicar com o usuário e falaremos sobre elas mais adiante. Confira a sintaxe a seguir:

```
window.alert("mensagem");
```

MÓDULO V

É importante dizer que a declaração window antes do **alert()** é opcional, pois o objeto **window** tem escopo global. Portanto, é possível escrever o comando simplesmente dessa forma:

```
alert("mensagem");
```

MÓDULO V

O que são caixas de diálogo e quais são os 3 tipos?

Basicamente, as caixas de diálogo são janelas modais que são exibidas no navegador com o objetivo de interagir com a pessoa usuária da página. Por meio delas é possível exibir alertas, fazer perguntas e obter respostas. Como cada uma tem a função diferente da outra, **a sintaxe e a forma de utilização do recurso também mudam**. Confira, a seguir, como os tipos de alert JavaScript funcionam.

MÓDULO V

Alert()

Como já mencionamos, o método **alert()** é indicado para quando é **preciso informar algo à pessoa usuária**, como avisar sobre uma falha no preenchimento de um formulário, dar boas-vindas em um novo cadastro efetuado em uma página e muitas outras situações. Para demonstrar o seu funcionamento, confira o código a seguir.

MÓDULO V

```
<!DOCTYPE html>
<html>
<head>
<title>
    Teste do método window.alert()
</title>
<meta charset="utf-8">
</head>
<body>
<form name="formCadastro" onsubmit="return validarFormulario()" method="post">
    <label for="nome">Informe o seu Nome:</label>
    <input type="text" name="nome">
    <input type="submit" value="Enviar">
</form>
<script>
    function validarFormulario(){
        var nomePessoa = document.forms["formCadastro"]["nome"].value;
        if (nomePessoa == "") {
            alert("Favor informar o seu nome!");
            return false;
        }
        else{
            alert("Olá, " + nomePessoa + " !");
            return true;
        }
    }
</script>
</body>
</html>
```

MÓDULO V

No exemplo acima, criamos um formulário em que a pessoa deve informar o seu nome. A seguir, fazemos a validação para verificar se o campo não está vazio. Assim, ao clicar no botão enviar, o evento onsubmit definido na tag **HTML** form chama a função **validarFormulário()**.

Além de verificar se o campo está vazio, a função exibe a mensagem de alerta para que a pessoa informe seu nome. Caso o conteúdo tenha sido preenchido, é exibida uma mensagem de cumprimento. Perceba que para recuperar o nome, usamos o comando: ***document.forms["formCadastro"]["nome"].value***.

Assim como o **BOM** está relacionado com as funções do navegador, existe o **DOM** (***Document Object Model***), que se relaciona com os elementos **HTML** da página e permite a sua manipulação por meio do **JavaScript**.

MÓDULO V

Confirm()

Outro tipo de caixa de diálogo é o **método confirm()**. Seu objetivo é permitir que a pessoa usuária da página decida se deseja ou não executar uma ação determinada. Para isso, exibe uma janela modal com uma mensagem e dois botões: um de confirmação e outro que cancela a ação. Confira a sintaxe:

```
confirm("mensagem");
```

É importante dizer que não é possível alterar o texto usado nos botões da janela modal do método **confirm()**, que correspondem a OK e Cancelar. Veja no algoritmo abaixo uma forma de aplicação do **JavaScript alert confirm** usado para confirmar a exclusão dos elementos de uma lista.

MÓDULO V

```
<!DOCTYPE html>
<html>
<head>
<title>
    Teste do método window.confirm()
</title>
<meta charset="utf-8">
</head>
<body>
    <p>Escolha um item da lista para removê-lo:</p>
    <select name="itens" id="itens" size="6" style="width: 150px">
        <option>Camiseta</option>
        <option>Calça</option>
        <option>Sapato</option>
        <option>Bolsa</option>
    </select>
    <br/>
    <input type="button" value="Excluir item" onclick=excluirItemSelecionado()>

<script>
    function excluirItemSelecionado(){
        var itens = document.getElementById("itens");
        if (itens.selectedIndex == -1){
            alert("Selecione um item na lista!");
            return;
        }
        var indice = itens.options[itens.selectedIndex].index;
        var itemSelecionado = itens.options[itens.selectedIndex].text;
        var resultado = confirm("Deseja excluir o item: " + itemSelecionado + " ?");
        if (resultado == true) {
            itens.removeChild(itens[indice]);
            alert("O item " + itemSelecionado + " será excluído da lista!");
        }
        else{
            alert("Você desistiu de excluir o item " + itemSelecionado + " da lista!");
        }
    }
</script>
</body>
</html>
```

MÓDULO V

Em primeiro lugar, criamos um código **HTML** com uma lista com quatro elementos que poderão ser excluídos por meio de um botão, que chama a função **excluirItemSelecionado()** ao ser clicado.

Perceba que aqui também usamos o objeto document para recuperar o elemento select e encontrar qual foi o item selecionado para a exclusão. Também colocamos uma validação logo no início da função, caso a pessoa usuária pressione o botão sem selecionar nenhum elemento da lista. Assim, será exibido um alerta para que ela faça uma escolha.

O próximo passo é a recuperação sobre qual é o índice do elemento selecionado e o seu conteúdo. Vamos precisar dessas informações para informar qual item será excluído e exibir na tela o alerta de confirmação, no qual mostramos para a pessoa qual foi o item escolhido por ela.

MÓDULO V

O **método confirm()** retorna verdadeiro se a escolha for a alternativa confirmar, e falso se a opção escolhida for cancelar. Por isso, precisamos usar uma variável para armazenar o retorno do comando quando o utilizamos no código. No nosso exemplo, criamos a variável resultado para guardar esse valor.

Portanto, se a variável resultado for verdadeira, excluímos o elemento correspondente da lista e mostramos uma mensagem para informar que o item escolhido foi removido. Já se a escolha for cancelar a operação, exibimos uma mensagem para mostrar que o item não será removido.

MÓDULO V

Prompt()

O próximo tipo de caixa de diálogo é o **método prompt()**. Seu objetivo é obter alguma informação da pessoa usuária da página. Confira a sua sintaxe:

```
window.prompt("argumento1", ["argumento2"]);
```

No qual:

argumento1: a mensagem destinada à pessoa usuária, que corresponde à solicitação feita.

argumento2: texto opcional que serve como um exemplo do que deve ser preenchido no prompt.

MÓDULO V

Além dos argumentos, o método **prompt()** também exibe dois botões: **OK** e **Cancelar**. O **OK** indica que a pessoa preencheu o campo, enquanto o cancelar significa que ela deseja cancelar a operação.
Vamos continuar com o exemplo anterior, entretanto, agora vamos adicionar um novo elemento à lista.
Veja o código a seguir.

MÓDULO V

```
<!DOCTYPE html>
<html>
<head>
<title>
    Teste do método window.prompt()
</title>
<meta charset="utf-8">
</head>
<body>
    <p>Vamos adicionar novos itens:</p>
    <select name="itens" id="itens" size="6" style="width: 150px">
        <option>Camiseta</option>
        <option>Calça</option>
        <option>Sapato</option>
        <option>Bolsa</option>
    </select>
    <br/>
    <input type="button" value="Adicionar item" onclick=adicionarItem()>

<script>
    function adicionarItem(){
        var item = prompt("Qual objeto você deseja incluir na lista?", "Adicione um novo
objeto");
        if (item == null || item == "") {
            alert("O uso do prompt foi cancelado!");
        } else {
            var itens = document.getElementById("itens");
            var option = document.createElement("option");
            option.text = item;
            itens.add(option, itens[0]);
        }
    }
</script>
</body>
</html>
```

MÓDULO V

Perceba que na função **adicionarItem()** usamos o método **prompt()** para solicitar à pessoa o nome do objeto que ela deseja adicionar. O próximo passo é verificar se ela pressionou a opção cancelar e exibir uma mensagem de alerta. Já se for informado um valor, ele será adicionado à lista de objetos.

Observe que usamos o **método prompt()** com o segundo argumento preenchido. Por isso, aparece a frase "*Adicione um novo objeto*" na linha do prompt.

MÓDULO V

Como pular linha no alert?

Como mencionado, não é possível alterar o texto dos botões da caixa de diálogo e também não podemos estilizar a janela modal com **CSS**. Entretanto, ainda é possível usar o **alert JavaScript personalizado**.

Uma forma de fazer isso é pulando linhas. O objetivo é tornar a mensagem mais clara e visível à pessoa usuária da página. Para isso, basta adicionar "**\n**" no texto a ser exibido. Confira o exemplo a seguir.

MÓDULO V

```
<!DOCTYPE html>
<html>
<head>
<title>
    Teste de quebra de linha
</title>
<meta charset="utf-8">
</head>
<body>
    <h2>Exibir um alert com quebra de linha!</h2>
    <input type="button" value="Exibir alert" onclick=exibeAlert()>

<script>
    function exibeAlert(){
        alert("Agora você já sabe\ncomo pular linha\nnos métodos: \nalert, confirm e prompt!");
    }
</script>
</body>
</html>
```

MÓDULO V

Qual a compatibilidade com os navegadores?

Verificar a compatibilidade dos recursos com os navegadores é importante para evitar falhas que podem ocorrer caso a ferramenta não ofereça suporte ao método usado. Afinal, para as pessoas que acessam a página, é desagradável clicar em um botão e não obter nenhuma resposta. Os métodos **alert(), confirm()** e **prompt()** são suportados pelos principais navegadores.

Confira a lista de navegadores compatíveis:

- Chrome;
- Internet Explorer;
- Firefox;
- Safari;
- Opera.

Os métodos **JavaScript alert, confirm** e **prompt** são alguns dos recursos que facilitam a interação entre a pessoa usuária e a página web. Cada um deles é indicado para uma situação específica. Por isso, é importante conhecer as formas de aplicação e a maneira de acessar o resultado dessa interação.

MÓDULO V

Operadores Lógicos AND (&&), OR (||) e NOT (!) em JavaScript

Operador lógico AND: &&

And, em inglês, significa "**E**".
Exemplo: isso **e** aquilo

Ele funciona da seguinte maneira:
Unimos mais de uma expressão, usando o operador **&&**

```
if (condicao1 && condicao2){
    codigo;
}
```

MÓDULO V

Agora, esse **IF** só será verdadeiro se a **condição1** for verdadeira **E** a **condição2 TAMBÉM** for verdadeira. Se qualquer uma delas for falsa, todo o teste condicional será **falso**.
Não existem limites para o tanto de testes que podemos usar com o operador **&&**:

```
if (condicao1 && condicao2 && condicao3 &&...){
    codigo;
}
```

MÓDULO V

Tabela verdade: AND &&

Condição 1	Condição 2	Resultado
Verdade	Verdade	**Verdade**
Verdade	Falso	**Falso**
Falso	Verdade	**Falso**
Falso	Falso	**Falso**

Exemplo de uso do operador &&

Você foi contratado para fazer um site para uma empresa de consórcio.
No teste inicial, você deve perguntar para o usuário a idade dele e o seu salário.
*A pessoa só pode comprar um automóvel se for de **maior** e ganhar mais de R$ 2.000,00*

MÓDULO V

Primeiro, vamos pegar a idade e armazenar na variável **idade**.
Depois, pegar o salário da pessoa e armazenar na variável **salário**.

Precisamos fazer dois testes:

1. idade > 17
2. salário > 2000.00

Para ser aprovado para o consórcio, as **duas** condições devem ser **true**.
A primeira **AND** a segunda também!

Se qualquer uma delas for falsa, cai no **else**. Nosso teste condicional fica:

- **if** (idade>17 && salario>2000)

MÓDULO V

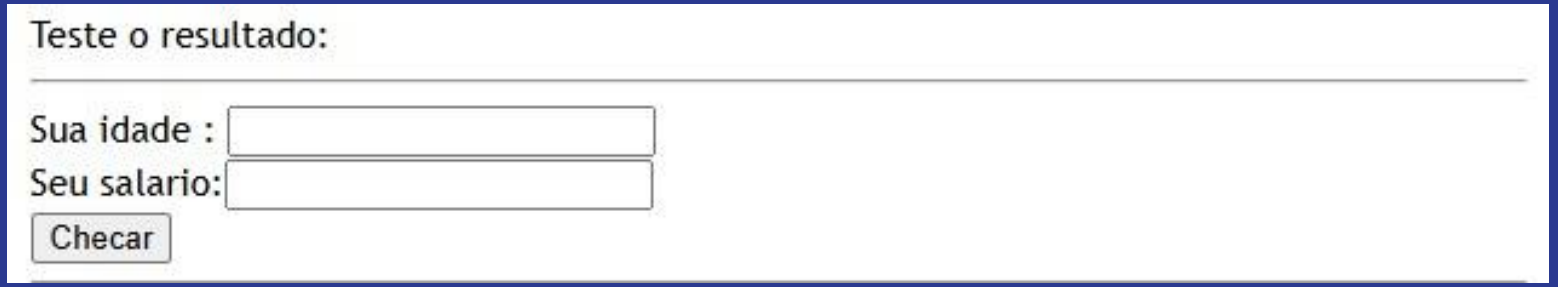
```
<!DOCTYPE html>
<html>
 <head>
   <title>Curso JavaScript Progressivo</title>
   Sua idade  : <input id="age" type="number"><br />
   Seu salario:<input id="salary" type="number"><br />
   <button onclick="checar()">Checar</button>

   <script type="text/javascript">
    function checar(){
     var idade = parseInt(document.getElementById("age").value);
     var salario = parseFloat(document.getElementById("salary").value);

     if (idade>17 && salario>2000)
      alert("Voce esta apto para participar do consorcio");
     else
      alert("Voce NAO esta apto para participar do consorcio");
    }
 </script>

 </head>
</html>
```

MÓDULO V



Tente fazer esse script sem o operador **&&.** Vai dar um trabalhão, vários **IFs** e **ELSEs** dentro de **IFs ELSEs**.

MÓDULO V

Operador lógico OR: ||

OR em inglês significa **OU**.
A sintaxe para usar o operador || é:
Unimos mais de uma expressão, usando o operador **&&**

```
if (condicao1 || condicao2){
    codigo;
}
```

MÓDULO V

Traduza, para o português mesmo:
Unimos mais de uma expressão, usando o operador **&&**

```
if (condicao1 OU condicao2){
    codigo;
}
```

Se a *condição1* **OU** *condição2* for verdadeira, todo o teste condicional será verdadeiro.
Faz, sentido, não é?

Se um for **falso** e o outro for **verdade**, o resultado é **verdade** (**true**).

MÓDULO V

Tabela verdade: OR ||

Condição 1	Condição 2	Resultado
Verdade	Verdade	**Verdade**
Verdade	Falso	**Verdade**
Falso	Verdade	**Verdade**
Falso	Falso	**Falso**

Exemplo de uso do operador **OR** || em JS

*Você foi contratado pelo governo para criar um script para definir as filas prioritárias. Seu site deve exibir um menu de opções, perguntando se a pessoa é **Gestante**, **Idosa**, **Deficiente** ou Nenhuma das alternativas.*

*Se ela for **Deficiente**, **Idosa** ou **Gestante**, tem direito a fila prioritária. Se não, não.*

MÓDULO V

Bom, vamos lá.
O resultado final vai ser o seguinte (pode testar):

Primeiro, criamos um site simples, exibindo um texto com as opções:

1. Gestante
2. Idoso(a)
3. Deficiente
4. Nenhuma das opções acima

MÓDULO V

Depois um formulário de input, do tipo ***number*** e ***id option.***
Por fim, um botão, que ao ser clicado chama a função **verificar()**.

Dentro do JS, vamos apenas definir a função **verificar()**.
Ela declara uma variável de nome **op** que vai receber a opção escolhida pelo usuário. Vale lembrar que o número digitado é capturado com o comando ***document.getElementById("option").value*** na forma de *string*.

Então, colocamos esse comando dentro da função **parseInt()** para transformar string em um inteiro.
Com o inteiro em mãos, vamos aos testes!

MÓDULO V

Vamos fazer três testes: verificar se o número digitado é 1, se é 2 ou se é 3.
Isso é feito com a seguinte linha de código:

```
if( (op == 1) || (op == 2) || (op == 3) )
```

Note que usamos o operador OR (OU) || .
Ou seja, basta um deles, qualquer ser verdadeiro, que o teste condicional **if** vai retornar **true** e ser executado!

Caso não seja 1, 2 ou 3 o valor digitado, o paciente não é prioritário e cai no **else**.

Esse é um exemplo real de código, coisas usadas no dia-a-dia, de verdade.

MÓDULO V

```
<!DOCTYPE html>
<html>
 <head>
   <title>Curso JavaScript Progressivo</title>
   Voce e:<br />
   1. Gestante<br />
   2. Idoso(a)<br />
   3. Dificiente<br />
   4. Nenhuma das opcoes acima<br />

   <input id="option" type="number"> <br />
   <button onclick="verificar()">Verificar</button>

   <script type="text/javascript">
    function verificar(){
     var op = parseInt(document.getElementById("option").value);

     if( (op == 1) || (op == 2) || (op == 3) )
          alert("Voce tem direito a fila preferencial");
     else
          alert("Voce NAO tem direito a fila preferencial");
    }
   </script>

 </head>
</html>
```

MÓDULO V

Operador lógico NOT: !

NOT em inglês significa **não**, é a negação.
Sempre que colocamos esse operador perto de alguma condição, ele altera o valor dela de **verdadeiro** para **falso** (**true** para **false**) ou o contrário, de **falso** para **verdadeiro**.

```
if(true){
    [esse codigo sempre executa];
}

if(!true){
    [esse codigo nunca executa];
}
```

MÓDULO V

Exemplo de uso do operador NOT em JavaScript

Crie um menu em JavaScript com diversas opções para o usuário, e caso ele digite 0, o sistema seja encerrado.

MÓDULO V

```
<!DOCTYPE html>
<html>
 <head>
   <title>Curso JavaScript Progressivo</title>
   Menu:<br />
   0. Sair<br />
   1. Comprar<br />
   2. Vender<br />
   3. Trocar<br />
   4. Etc<br />

   <input id="option" type="number"> <br />
   <button onclick="verificar()">Verificar</button>

   <script type="text/javascript">

    function verificar(){
     var op = parseInt(document.getElementById("option").value);

     if(!op)
  alert("Saindo do sistema...");
 else
      alert("Opcao escolhida: "+op);
    }
 </script>

 </head>
</html>
```

MÓDULO V

Esse é um típico exemplo de uso do operador de negação **NOT !** , que inverte o valor **booleano**.
A única coisa diferente nesse código é o:

if(!op)

Ele funciona assim:

1. Se você digitar **0**, o teste **if(op)** daria sempre falso, pois **0** é **false**. Porém, a gente colocou o operador **!**, então inverteu. Sempre que o usuário digitar **0**, o teste **!op** vira **true**
2. Qualquer outro número que o usuário digitar para a variável **op**, o teste **if(op)** seria **true**, então o teste **if(!op)** vai dar falso para qualquer coisa diferente de **0** que a pessoa digitar, por isso vai cair no **ELSE**.

MÓDULO V

JavaScript Switch

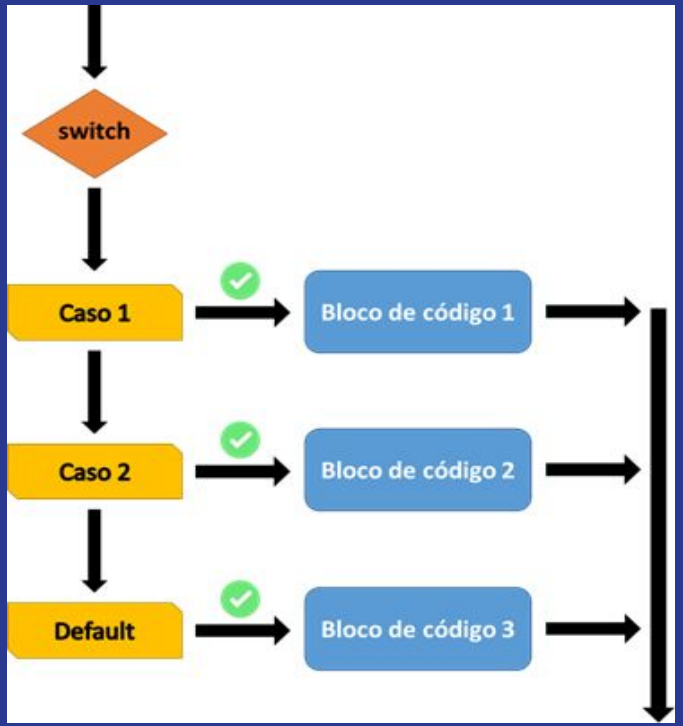
A estrutura condicional **switch** permite executar um bloco de código diferente de acordo com cada opção (cada case) especificada. Seu uso é indicado quando os valores a serem analisados nessas condições são pré-definidos.

```
var tipoUsuario = "Gerente";

switch (tipoUsuario) {
    case "Admin":
        mensagem = "*|*| Feliz Natal, chefe! |*|*";
        break;
    case "Gerente":
        mensagem = "Boas festas, meu amigo!";
        break;
    default:
        mensagem = "Boas festas!";
}
```

MÓDULO V

O valor atribuído à variável mensagem será: “Boas festas, meu amigo!”.
A **Figura 1** demonstra o funcionamento do condicional switch.



MÓDULO V

Sintaxe

```
switch(expressão){
 case n1:
    bloco de código 1
    break;

 case n2:
   bloco de código 2
   break;
 default:
   bloco de código 3
}
```

MÓDULO V

Expressão é a expressão a ser comparada com cada **case** declarado dentro do **switch**. Caso o valor obtido na expressão seja a igual ao que for declarado no **case**, o bloco de código é executado.

case é o valor que será comparado à expressão.

break é a palavra reservada que finaliza a execução do **switch**. Caso não especificada no final do bloco de código em execução, as linhas dos blocos de código seguintes também serão executadas.

Default é a palavra reservada que define o bloco de código a ser executado se nenhum dos **cases** atenderem à expressão declarada no **switch**.

MÓDULO V

Exemplos de switch

Exemplo 1

No exemplo a seguir demonstramos como exibir diferentes alimentos de acordo com a necessidade do usuário:

```
var alimento = "Gordura";

switch (alimento) {
    case "Proteína":
        mensagem = "Carne, leite, aveia, amêndoas";
        break;
    case "Carboidrato":
        mensagem = "Banana, batata doce, feijão, pão";
        break;
    default:
        mensagem = "Cuidado com a alimentação!";
}
```

O valor atribuído a mensagem é: “Cuidado com a alimentação! ".

MÓDULO V

Exemplo 2

Neste exemplo a seguir demonstramos que o uso do break não é necessário, porém ao não declará-lo e a depender da lógica de negócio, um erro pode ser inserido no projeto.

```
var cargo = "gerente";
var salario = 2000;

switch (cargo) {
    case "gerente":
        salario *= 1.15;
    case "supervisor":
        salario *= 1.10;
    default:
        salario *= 1.05;
}
```

MÓDULO V

O valor atribuído a **salario** é **2.656,50**. A ausência do **break** em cada bloco de código faz com que o código declarado dentro de cada **caso** seja executado independentemente do **case** atender à condição especificada no **switch**.

MÓDULO V

Exemplo 3

No exemplo a seguir demonstramos uma situação em que a ausência do **break** evita a escrita de linhas de código desnecessárias:

O valor impresso no console é "Outono".

MÓDULO V

```
var mes = "Maio";

switch (mes) {
    case "Janeiro":
    case "Fevereiro":
    case "Março":
        console.log("Verão!");
        break;
    case "Abril":
    case "Maio":
    case "Junho":
        console.log("Outono!");
        break;
    case "Julho":
    case "Agosto":
    case "Setembro":
        console.log("Inverno!");
        break;
    case "Outubro":
    case "Novembro":
    case "Dezembro":
        console.log("Primavera!");
}
```

MÓDULO V

Exemplo 4

No exemplo a seguir demonstramos como customizar uma mensagem de boas-vindas com switch:

```
sexo = "feminino";

switch (sexo) {
    case "feminino":
        console.log("Bem-vinda!");
        break;
    case false:
        console.log("Bem-vindo!");
}
```

O valor impresso no console é "Bem-vinda!"

MÓDULO V

Exemplo 5

No exemplo a seguir demonstramos como obter o nome do mês a partir do seu respectivo número com **switch**:

O valor impresso no console é "Abril". Caso o valor seja diferente dos especificados em cada **case**, é impresso "Mês inexistente".

MÓDULO V

```
var mes = 4;
var nomeMes = "";

switch (mes) {
    case 1:
        nomeMes = "Janeiro";
        break;
    case 2:
        nomeMes = "Fevereiro";
        break;
    case 3:
        nomeMes = "Março";
        break;
    case 4:
        nomeMes = "Abril";
        break;
    case 5:
        nomeMes = "Maio";
        break;
    case 6:
        nomeMes = "Junho";
        break;
```

MÓDULO V

```
    case 7:
        nomeMes = "Julho";
        break;
    case 8:
        nomeMes = "Agosto";
        break;
    case 9:
        nomeMes = "Setembro";
        break;
    case 10:
        nomeMes = "Outubro";
        break;
    case 11:
        nomeMes = "Novembro";
        break;
    case 12:
        nomeMes = "Dezembro";
        break;
    default:
        nomeMes = "Mês inexistente";
}
console.log(nomeMes);
```

MÓDULO V

Exemplo 6

No exemplo a seguir demonstramos como redirecionar um usuário para uma página diferente com switch:

```
var tipoUsuario = "Adolescente";

switch (tipoUsuario) {
    case "Adolescente":
        window.location.href = "homejovem.html";
        break;
    case "Adulto":
        window.location.href = "home.html";
}
```

O usuário será redirecionado para "homejovem.html".

MÓDULO V

EXERCÍCIOS...

A Educação é o primeiro passo para um futuro melhor...

MÓDULO V

FIM...

A Educação é o primeiro passo para um futuro melhor...