JavaScript

Lógica Condicional

- São utilizados para aplicar a lógica de condição de causa e consequência no site. Seja um evento, um aperto de botão ou um cálculo.
- São utilizados os seguintes comandos:
  - If, else if, else
  - switch case
- Os operadores de comparação são:

==	igual a
===	igual valor e igual tipo
!=	diferente
!==	diferente valor OU diferente tipo
>	maior que
<	menor que
>=	maior ou igual que
<=	menor ou igual que

Lógica Condicional

If, else if, else

- São utilizados para conferir eventos a partir de uma condição.
- O código é:

```
if (condição)
{
     // código caso a condição seja verdadeira
}
else if (condição)
{
     // código caso a condição anterior não seja verdadeira, testa de novo
}
else
{
     // código caso nenhuma condição anterior não seja verdadeira, faz esse código
}
```

Funções

- Utilizadas para realizar alguma ação ao serem chamadas.
  - Exemplo seria ao apertar o botão SUBMIT e antes de enviar os dados, conferir se eles estão ok.
- O jeito de escrever uma função é escrevendo `function` seguido por o nome da função e um parênteses para os parâmetros. Todo o código da função fica dentro de chaves e, caso haja retorno de algum parâmetro, se utiliza o `return` no final da função. Exemplo:

```
function NOME(Parametro1, Parametro2, Parametro3)
{
      // Código a ser executado
      return Variavel_ou_Valor
}
```

Funções

- Para chamar a função, dentro do código em HTML, basta chamar o NOME e preencher os parâmetros dentro dos parênteses. Caso se chame sem parênteses, ele recebe o objeto função com todo o código.

  Exemplo:

```
function Temp_Celsius(Temp_Fahrenheit)
{
     var Calculo = (5/9) * (Temp_Fahrenheit-32); (pode se criar uma variável para receber a operação de sua função, variável temporária, deixa de ocupar memória assim que o cálculo é executado, variável local, atentar para não repetir nome de variáveis que ja existem)
     return Calculo;
}
//chamada de função
document.getElementById("Temp").innerHTML = Temp_Celsius(NUMERO);
```

Formulário

- Dentro dos elementos dos HTML, é possível pegar os valores dos campos utilizando a propriedade do elemento chamada “**.value**”.

  Exemplo:

```
<input type="range" id="Quantidade_Jogadores" onchange="Jogadores()">
<p id="Num_Jogadores"> </p>

function Jogadores()
{
     var Valor_Num_Jogadores = document.getElementById("Quantidade_Jogadores").value;
     document.getElementById("Num_Jogadores").innerHTML = Valor_Num_Jogadores;
}
```

Formulário

- Vale lembrar que para o input realizar alguma ação, necessitamos utilizar os eventos do HTML.

  Exemplo:

```
<input type="range" id="Quantidade_Jogadores" onchange="Jogadores()">

function Jogadores()
{
     var Valor_Num_Jogadores = document.getElementById("Quantidade_Jogadores").value;
     document.getElementById("Num_Jogadores").innerHTML = Valor_Num_Jogadores;
}
```

Formulário - Eventos

- **`onblur`** - Ao perder o foco.
  - O evento dispara quando se tira o foco do input, ou seja, quando se clica fora do espaço dele.
- **`onchange`** - Ao sofrer alguma mudança.
  - O evento dispara quando se modifica o valor do input (seja digitado ou selecionado um novo valor).
  - Ao contrário do **`oninput`**, ele ativa após receber o valor e perder o foco (sair do campo do input).
- **`oncontextmenu`** - Ao utilizar o Menu de Contexto.
  - O evento dispara quase se usa o botão direito do mouse para abrir o Menu de Contexto.
- **`onfocus`** - Ao ganhar o foco.
  - O evento dispara quase se dá o foco no input, ou seja, quando se clica dentro do espaço dele.
- **`oninput`** - Ao receber um input.
  - O evento dispara quando se insere um valor do input.
  - Ao contrário do **`onchange`**, ele ativa imediatamente ao receber o valor.

Formulário - Eventos

- **oninvalid** - Ao input ser inválido.
  - O evento dispara quando o input não é válido (letra em telefone, campo em branco).
- **onreset** - Ao se clicar no botão RESET (inglês) ou REDEFINIR (português).
  - O evento dispara quando se aperta o botão definido como `<input type="reset">`.
- **onsearch** - Ao realizar a procura.
  - O evento dispara quase se aperta ENTER no campo de procura definido como `<input type="search">`.
- **onselect** - Ao se selecionar um texto em um input.
  - O evento dispara quando se seleciona um texto em algum campo de input.
- **onsubmit** - Ao se clicar no botão SUBMIT (inglês) ou ENVIAR (português).
  - O evento dispara quando se aperta o botão definido como `<input type="submit">`.

Outros eventos que não são específicos de formulário estão em:
< https://www.w3schools.com/tags/ref_eventattributes.asp >

Formulário - Máscara

- Para realização da máscara do formulário se utilizar os eventos em conjunto com as funções.
- Eventos mais utilizados:
  - `onchange` ;
  - `oninput` ;
  - `oninvalid` ;
- As propriedades de strings estão em <https://www.w3schools.com/jsref/jsref_obj_string.asp>

Formulário - Máscara

- Métodos mais utilizados (Procura nas Strings):
  - **.length**
    Mostra o tamanho da string;
  - **.indexOf(**`"termo_procurado"`**,** `n°_do_Indice_Inicial`**)**
    Retorna o índice da string onde aparece a primeira ocorrência do `"termo_procurado"`.
    - Procura do começo ao final.
  - **.lastIndexOf(**`"termo_procurado"`**,**`n°_do_Indice_Inicial`**)**
    Retorna o índice da string onde aparece a última ocorrência do `"termo_procurado"`.
    - Procura do final ao começo.
  - **.search(**`"termo_procurado"`**)**
    - Retorna o índice da string onde aparece a primeira ocorrência do `"termo_procurado"`.
      - Procura do começo ao final.
      - É diferente do **.indexOf()** pois não permite escolher da onde procurar, porém permite utilizar Expressões Regulares (`\`,`^`,`$`,`*`,`+`,`?`,`.`,`[\b]`,`\t`,`\n`,`\s`, e etc)
- Se iniciam com 0, que nem em Arrays e caso não achem uma ocorrência, retornam o valor -1;

Formulário - Máscara

- Métodos mais utilizados (Extração de termos na String por String):
  - **.slice(**`n°_do_Indice_Inicial`**,**`n°_do_Indice_Final`**)**
    Extrai parte de uma string e cria uma nova a partir do início e fim configurado.
    - Se não colocar o `n°_do_Indice_Final`, ele pega todo o resto do string.
    - Se os valores forem negativos, ele retira do fim pro começo.
  - **.substring(**`n°_do_Indice_Inicial`**,**`n°_do_Indice_Final`**)**
    Extrai parte de uma string e cria uma nova a partir do início e fim configurado.
    - Se não colocar o `n°_do_Indice_Final`, ele pega todo o resto do string.
    - É parecido com o **.slice()**, porém **NÃO** permite índices negativos.
  - **.substr(**`n°_do_Indice_Inicial`**,**`tamanho_do_recorte_da_string`**)**
    Extrai parte de uma string e cria uma nova a partir do início e o tamanho configurado.
    - Se não colocar o `tamanho_do_recorte_da_string`, ele pega todo o resto do string.
    - Se os valores forem negativos, ele retira do fim pro começo.
    - É parecido com o **.slice()**, porém o segundo parâmetro é o tamanho do recorte.
  - Lembre-se que iniciam com 0, que nem em Arrays

Formulário - Máscara

- Métodos mais utilizados (Extração de termos na String por Char):
  - **.charAt(**`n°_do_Indice`**)**

    Extrai o char da string, considerando ela como se fosse um Array.
  - **[**`n°_do_Indice`**]**

    Trata o String como um Array e extrai o char da string de acordo com a posição.
- Lembre-se que iniciam com 0, que nem em Arrays

Formulário - Máscara

- Métodos mais utilizados (Mudança na String):
  - **.replace(**`"termo_procurado"`**,**`"termo_novo"`**)**

    Substitui o `"termo_procurado"` que aparece na primeira ocorrência pelo `"termo_novo"`.
    - Procura do começo ao final.
    - Permite utilizar Expressões Regulares (`\`,`^`,`$`,`*`,`+`,`?`,`.`,`[\b]`,`\t`,`\n`,`\s`, e etc).
  - **.toUpperCase()**

    Coloca todas as letras em maiúsculo;
  - **.toLowerCase()**

    Coloca todas as letras em minúsculo;
  - **.concat(**`"termo1"`**,**`"termo2"`**)**

    Contanena os termos (o mesmo que utilizar o `+`). Fazem as mesmas coisas:

```
var TEXTO = "TEXTO 1" + " " + "TEXTO 2";
var TEXTO = "TEXTO 1".concat(" ","TEXTO 2");
```

Métodos (https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/String)