

JavaScript

Aula 2



Comandos condicionais e repetição

Instrução While

A instrução while realiza uma ação enquanto determinada condição for satisfeita.

Sintaxe é:

```
while (<condição>)  
{  
    <comando>;  
    <comando>;  
}
```

JS

Veja no exemplo seguinte a utilização do laço while que é repetido por total de 10 vezes:

```
<script>
```

```
    num = 0;
```

```
    while (num < 10)
```

```
    {
```

```
        console.log("Número: "+num+"<br>");
```

```
        num++;      /* faz o incremento para a variável  
                     num*/
```

```
    }
```

```
</script>
```

JS

The screenshot shows a web browser window with the address bar displaying the URL: `objetos%20Acadêmicos/Instituições/Recode%20Pro/JS/whileconsole.html`. The browser's developer tools are open, and the 'Console' tab is selected. The console shows the output of a JavaScript while loop, displaying numbers from 0 to 9, each followed by a line break (`
`). The source of each log entry is `whileconsole.html:4`.

Browser tabs and bookmarks are visible at the top. The developer tools interface includes tabs for Elements, Console, Sources, Network, Performance, Memory, Application, and Security. The Console panel shows a list of log entries with a filter input and a 'Default levels' dropdown.

Log Entry	Source
Número: 0 	<code>whileconsole.html:4</code>
Número: 1 	<code>whileconsole.html:4</code>
Número: 2 	<code>whileconsole.html:4</code>
Número: 3 	<code>whileconsole.html:4</code>
Número: 4 	<code>whileconsole.html:4</code>
Número: 5 	<code>whileconsole.html:4</code>
Número: 6 	<code>whileconsole.html:4</code>
Número: 7 	<code>whileconsole.html:4</code>
Número: 8 	<code>whileconsole.html:4</code>
Número: 9 	<code>whileconsole.html:4</code>

Atividade Prática 1

1. Fazer um programa para que o usuário informe um número e a tabuada daquele valor seja calculada e exibida na tela. As multiplicações começam no valor 1 até o valor 9.

Exemplo da informação: $2 \times 1 = 2$

- O programa deve ter um título;
- No `<body>` deve ter um `<h2>` Com uma informação que represente o que o programa irá fazer;
- No `<body>` deve ter um parágrafo com orientações sobre a estrutura de repetição. É preciso utilizar o `<style>` para dar uma forma mais bonita no parágrafo.

JS

Do While

O do while é uma variante do loop while. Este loop executará o bloco de código uma vez, antes de verificar se a condição é verdadeira e, em seguida, repetirá o loop enquanto a condição for verdadeira.

Sintaxe:

```
do
{
    <commando>;
}
while (condition);
```

JS

Exemplo:

```
<!DOCTYPE html>
<html>
  <body>
    <h2>JavaScript Do While</h2>
    <script>
      let text = ""
      let i = 0;
      do {
        text += "<br>O número é: " + i;
        i++;
      }
      while (i < 10);
      document.write(text);
    </script>
  </body>
</html>
```

Atividade Prática 2

1. Fazer um programa que mostre todos os números pares de 1 até 100 em ordem decrescente.
- O programa deve ter um título;
 - No `<body>` deve ter um `<h2>` Com uma informação que represente o que o programa irá fazer;
 - Inserir uma imagem que represente uma estrutura de repetição. A imagem deve estar centralizada na tela;
 - No `<body>` deve ter um parágrafo com orientações sobre a estrutura de repetição. É preciso utilizar o `<style>` para dar uma forma mais bonita no parágrafo.

Instrução for

A instrução for realiza uma ação até que determinada condição seja satisfeita.

Sua sintaxe básica é:

```
for (início; condição; incremento)
{
    <comando>;
}
```

O início determina o valor inicial do laço for. Normalmente é 0 ou 1, porém poderá ser especificado qualquer outro valor. O valor especificado é atribuído em uma variável, por exemplo `i=0`, `count=1`.

JS

A condição determina a expressão que irá controlar o número de repetições do laço. Enquanto esta expressão for verdadeira, o laço continuará, caso o laço seja falso, o laço terminará. Por exemplo: `i<20`. Enquanto o valor de `i` for menor que 20, a condição é verdadeira.

O incremento determina como o laço irá contar, de 1 em 1, 2 em 2, 5 em 5, 10 em 10, enfim.

Exemplo: `i++`. Será aumentado o valor da variável `i` a cada repetição. Diferente de todas as outras linguagens, em JavaScript, a instrução `for`, utiliza ponto e vírgula para separar os argumentos ao invés de vírgula.

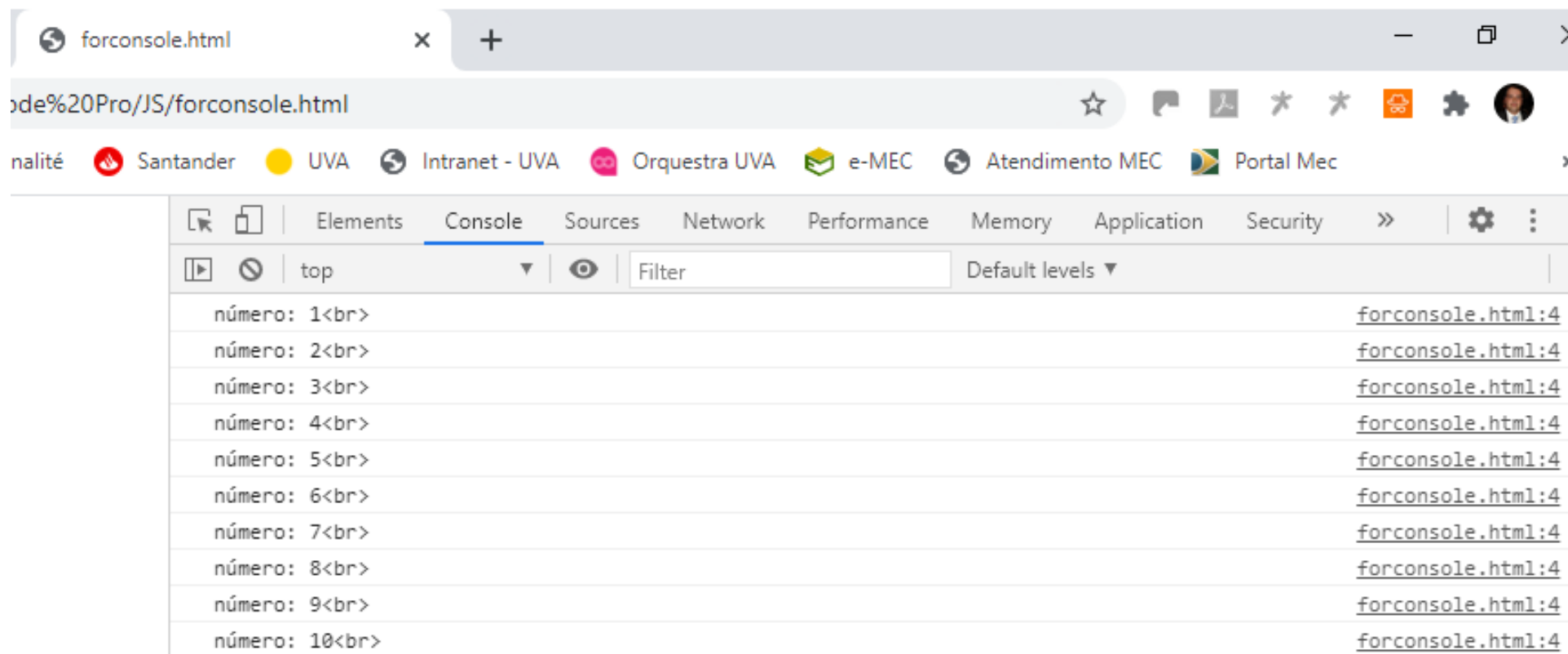
JS

Vejamos um exemplo prático de utilização do laço for que conta valores de 1 até 10, acrescentando um valor de cada vez:

Exemplo:

```
<script>  
    for (i=1 ; i<=10 ; i++)  
    {  
        console.log("número: "+ i + "<br>");  
    }  
</script>
```

JS



Atividade Prática 3

1. Fazer um programa para digitar 5 números e após o término da repetição é preciso exibir na página o valor total (somatório) dos números informados. É preciso exibir também após o término da repetição quantos números digitados são ímpares e pares.
- O programa deve ter um título;
 - No `<body>` deve ter um `<h2>` Com uma informação que represente o que o programa irá fazer;
 - Inserir uma imagem que represente uma estrutura de repetição. A imagem deve estar centralizada na tela;
 - No `<body>` deve ter um parágrafo com orientações sobre a estrutura de repetição. É preciso utilizar o `<style>` para dar uma forma mais bonita no parágrafo.

JS

Array

Os arrays JavaScript são usados para armazenar vários valores em uma única variável.

Array: `carros = ["Fiat", "Volvo", "BMW ", "Mercedes"];`
`carros:` Nome do array

Exemplo:

```
<html>
  <body>
    <h2>JavaScript Arrays</h2>
    <script>
      const carros = ["Fiat", "Volvo", "BMW", "Mercedes"];
      console.log(carros[1]);
    </script>
  </body>
</html>
```

JS

Inserindo Informações no Array

Os arrays JavaScript são usados para armazenar vários valores em uma única variável.

Exemplo:

```
<html>
  <body>
    <h2>JavaScript Arrays</h2>
    <script>
      carros = ["Fiat"];
      document.writeln(carros);
      carros[1] = "Ford";    ← Atribui Ford ao índice 1 do array carros
      document.writeln(carros);
      carros[2] = "Chevrolet"; ← Atribui Chevrolet ao índice 2 do array carros
      document.writeln(carros);
    </script>
  </body>
</html>
```

Atividade Prática 4

1. Fazer um programa para criar um array com 3 elementos que representem frutas.
2. O programa deverá inserir mais dois elementos no array.
3. O programa deverá exibir a informação do terceiro elemento do array.
4. O programa também deverá exibir a informação do último elemento do array.

Iterando sobre os itens de um array

A melhor maneira de percorrer um Array é usando um loop "for":

```
<script>
    var frutas, text, fLen, i;
    frutas = ["Banana", "Laranja", "Manga", "Morango"];
    fLen = frutas.length;
    for (i = 0; i < fLen; i++)
    {
        document.write(frutas[i]);
    }
</script>
```

Atividade Prática 5

1. Fazer um programa para criar um array frutas = ["Banana", "Laranja", "Manga", "Manga", "Pera", "Morango"]; .
2. O programa deverá dentro de uma estrutura de repetição exibir as frutas cujo índice tem valor ímpar.

JS

Método push()

Para **adicionar** elementos em um Array com JavaScript, podemos utilizar o **método push()**, que significa “empurrar”. Desta forma, o próximo dado será sempre inserido ao final e não precisamos nos preocupar com a posição do índice.

Exemplo:

```
<script>  
  let frutas = [];  
  frutas.push("carambola");  
  frutas.push("maça");  
  frutas.push("amora");  
  console.log(frutas);  
</script>
```

Exibição do Array

frutas	carambola	maça	Amora
índices	0	1	2

JS

Método pop(), splice() e shift()

Podemos também **remover** um elemento do Array. Para isso, o primeiro método de remoção que usamos é o **pop()**. Ele remove simplesmente e apenas o último elemento de um array.

Exemplo:

```
<script>  
  let frutas = [];  
  frutas.push("carambola");  
  frutas.push("maça");  
  frutas.push("amora");  
  frutas.pop();  
  console.log(frutas);  
</script>
```

Exibição do Array

frutas	carambola	Maça
índices	0	1

JS

Existem outros métodos de remoção como o **splice()** e o **shift()**. O **splice()** remove um elemento específico do Array e recebe **2 parâmetros**: o primeiro é a posição inicial, o segundo é a quantidade de elementos. Portanto, se quisermos excluir a fruta “maça”, faríamos a seguinte declaração:

Exemplo:

```
<script>
```

```
let frutas = [];  
frutas.push("carambola");  
frutas.push("maça");  
frutas.splice(1,1);  
console.log(frutas);
```

```
</script>
```

frutas	carambola	maça
índices	0	1

Exibição do Array

carambola

JS

O método **shift()**, que, ao contrário do **pop()**, remove o primeiro elemento de um Array.

Exemplo:

```
<script>
  let frutas = [];
  frutas.push("carambola");
  frutas.push("maça");
  frutas.shift();
  console.log(frutas);
</script>
```

frutas	carambola	maça
índices	0	1

Exibição do Array

maça

Propriedade e Métodos do Array

A força real dos Arrays de JavaScript são as propriedades e métodos do Array incorporada:

- `let x = carros.length;` → A propriedade `length` retorna o número de elementos.
- `elementos let y = carros.sort();` → O método `sort()` classifica os arrays.

A Propriedade length para o array

A propriedade length de um Array retorna o comprimento de um Array (o número de elementos do Array).

let frutas = ["Banana", "Laranja", "Maça", "Manga"];
frutas.length; → o length de frutas é 4 (quatro dados no array).

Exemplo:

```
<script>  
  let frutas = ["Banana", "Laranja", "Maça", "Manga"];  
  console.log(frutas.length);  
</script>
```

A propriedade length é sempre mais do que o índice do Array mais alto, pois o índice inicia com 0 e o length inicia com 1.

JS

A Propriedade sort() para o array

A propriedade sort() de um Array faz a ordenação das informações em ordem decrescente.

```
let frutas = ["Banana", "Laranja", "Maça", "Manga"];
```

Exemplo:

```
<script>
```

```
let frutas = ["Banana", "Carambola", "Amora", "Manga"];
```

```
frutas.sort();
```

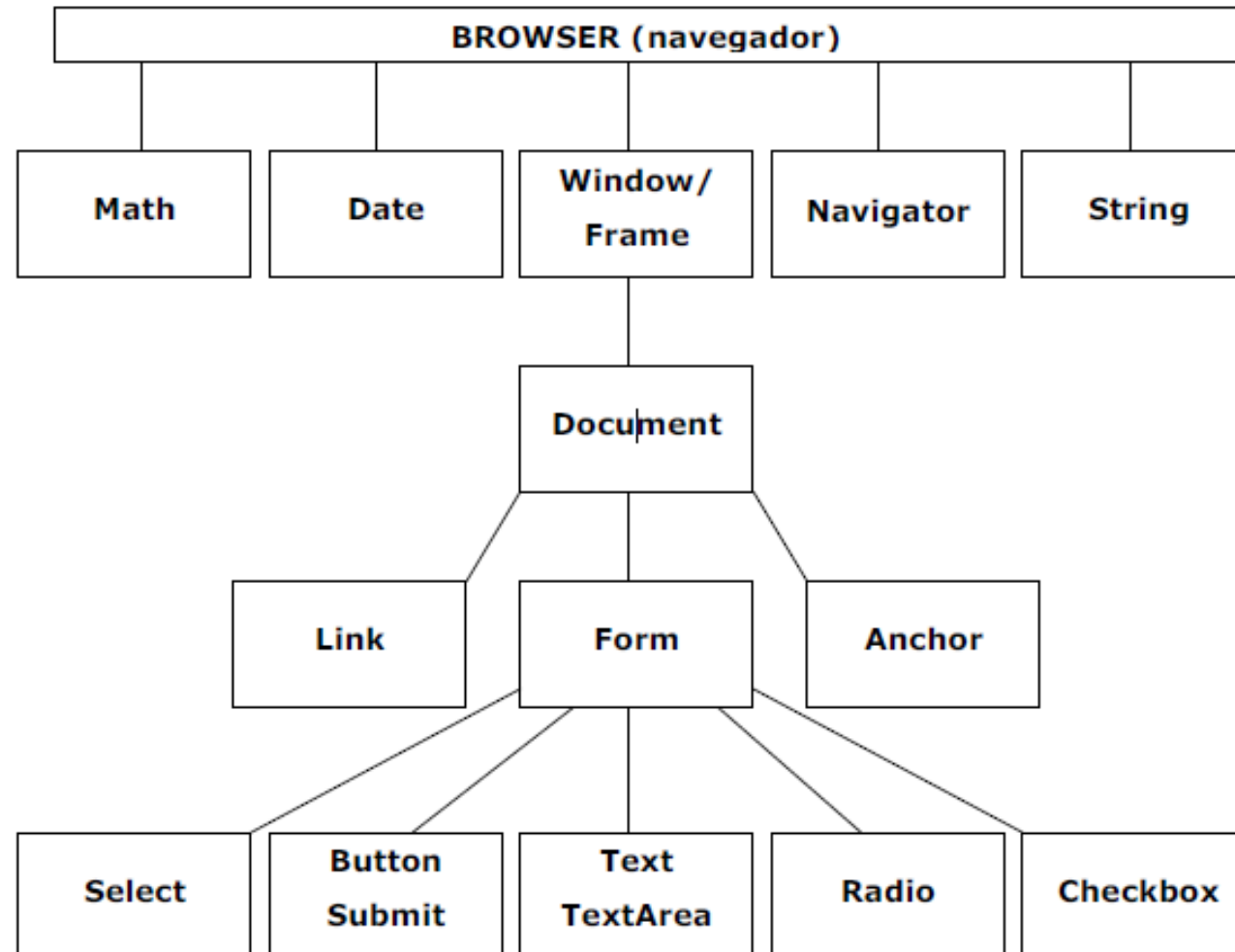
```
console.log(frutas);
```

```
</script>
```

frutas	amora	banana	carambola	maça
índices	0	1	2	3

Hierarquia dos objetos do JavaScript

A linguagem JavaScript manipula vários tipos de objetos através do uso de suas propriedades e métodos. Estes objetos são representados por uma hierarquia, fazendo com que alguns objetos se tornem propriedades de outros.



window → O objeto mais acima na hierarquia, contém propriedades que se aplicam a toda a janela. Há também um objeto desta classe para todas as "sub-janelas" de um documento com frames `location`: Contém as propriedades da URL actual.

history → Contém as propriedades das URLs visitadas anteriormente.

document → Contém as propriedades do documento contido na janela, tais como o seu conteúdo, título, cores, etc.

Orientação a objetos

Diferente da Linguagem HTML, a linguagem JavaScript corresponde a programação orientada a objetos, isto significa que todos os elementos de uma página da Web são tratados como objetos.

Estes objetos são agrupados de acordo com seu tipo ou finalidade.

Dentro da linguagem JavaScript, são criados automaticamente objetos que permitem que o usuário possa criar novos objetos de acordo com sua conveniência.

Propriedade de objetos

Cada objeto existente na manipulação do JavaScript possui **propriedades (características)**.

Sabemos que um documento HTML possuem título e corpo, estas características do documento podemos chamar de propriedades que existem neste documento.

Estas propriedades existem de dois tipos, algumas são os objetos propriamente ditos enquanto outras não.

Exemplo: o **objeto form (formulário)** que é uma propriedade do **objeto document (documento)**, conforme mostrado no organograma apresentado anteriormente.

JS

Já a **propriedade de título da página (title)**, é pertencente ao objeto document não havendo nenhuma propriedade sobre ela. Concluindo, podemos dizer que a propriedade form do objeto document é um objeto-filho e o objeto document é o objeto-pai.

Em geral, as propriedades podem conter valores (string, números, entre outros tipos).

A utilização de propriedades se dá acompanhada de seu objeto sendo separados por um ponto apenas.


JS

propriedades e valores próprios que são ajustados pelo conteúdo da própria página.

Todos eles seguem uma hierarquia que reflete toda a estrutura de uma página HTML.

A linguagem JavaScript pode ser utilizada para a criação de scripts tanto do lado cliente como do lado servidor. Seguindo a hierarquia de objetos da linguagem JavaScript, são criados os seguintes objetos ao ser carregada uma página.

Objetos, Propriedade e Métodos

Objeto	Propriedades	Métodos
	<code>car.name = Fiat</code> <code>car.model = 500</code> <code>car.weight = 850kg</code> <code>car.color = white</code>	<code>car.start ()</code> <code>car.drive ()</code> <code>car.brake ()</code> <code>car.stop ()</code>

Todos os carros têm as mesmas **propriedades**, mas os **valores** das propriedades diferem de carro para carro.

Todos os carros têm os mesmos **métodos**, mas os métodos são executados **em momentos diferentes**.

Acessando Propriedades do Objeto

Você pode acessar as propriedades do objeto de duas maneiras:

- `objectName.propertyName;`
- `objectName["propertyName"];`

JS

Exemplo:

```
<html>
  <body>
    <h2>Criação de Objetos no Javascript</h2>
    <script>
      let pessoa =
      {
        primNome: "marcus",
        ultNome : "antunes",
        idade  : 36
      };
      document.write (pessoa.primNome + " " + pessoa.ultNome);
    </script>
  </body>
</html>
```

Objeto

Propriedade

JS

Exemplo:

```
<html>
  <body>
    <h2>Criação de Objetos no Javascript</h2>
    <script>
      var meuCarro = new Object();
      meuCarro.fabricacao = "Ford";
      meuCarro.modelo = "New Fiesta";
      meuCarro.ano = 2020;
      document.write(meuCarro.fabricacao + "<br>");
      document.write(meuCarro.modelo + "<br>");
      document.write(meuCarro.ano);
    </script>
  </body>
</html>
```

Objeto

Propriedade

Métodos dos objetos

Além das propriedades, os objetos podem conter métodos que são funções pré-definidas pela linguagem JavaScript que irão executar determinada operação.

Dentro de um documento o usuário poderá utilizar o método de escrever neste documento para exibir um texto qualquer.

Os métodos estarão sempre associados à algum objeto presente no documento e cada método faz parte de um objeto específico.

JS

Não tente usar métodos em objetos que não o utilizam, isto faz com que a linguagem JavaScript cause erro na execução do script. Na maioria das vezes os métodos são usados para alterar o valor de uma propriedade ou executar uma tarefa específica.

Veja a sintaxe de utilização dos métodos:

nomeobjeto.metodo(argumento)

Na sintaxe apresentada, **nomeobjeto** faz referência ao objeto a ser utilizado e o qual sofrerá uma ação do método, já **método** é o nome de identificação do método usado e entre parênteses, **(argumento)** é a expressão ou valor opcional que será usada para alterar sobre o objeto.

JS

Exemplo:

const pessoa =

{

pName: "Ana",

sName : "Clara",

ident : 286732184,

nomeCompleto : function()

{

return this.pName + " " + this.sName;

}

};

Objeto

Propriedades

Método

Em uma definição de função, **this** refere-se ao "proprietário" da função.

No exemplo acima, **this** é o **objeto pessoa** que "possui" a função nomeCompleto.

Em outras palavras, **this.pName** significa propriedade pName deste objeto .

JS

Exemplo:

```
<!DOCTYPE html>
<html>
  <body>
    <h2 style = color:crimson>Método de um Objeto</h2>
    <p><b>O método de um objeto é definido pela função</b></p>
    <script>
      // criação do objeto
      const pessoa = {
        pName: "Ana",
        sName: "Clara",
        ident: 236789352,
        nomeCompleto: function()
        {
          return this.pName + " " + this.sName;
        }
      };
      document.write(pessoa.nomeCompleto());
    </script>
  </body>
</html>
```

Atividade Prática 6

1. Fazer um programa para criar um objeto:
 1. Carro
 1. fabricante
 2. modelo
 3. ano
 4. qtdPortas
 5. retronaDados ()
2. A função deve retornar três informações quais quer (você irá decidir).
3. Exibir as informações de retorno da função;
4. É importante que haja um <h2> com um título;
5. É importante que tenha um parágrafo com <style>;