Programação Web

Javascript

Patrícia Leite e Marta Martinho 2021

Organização da apresentação

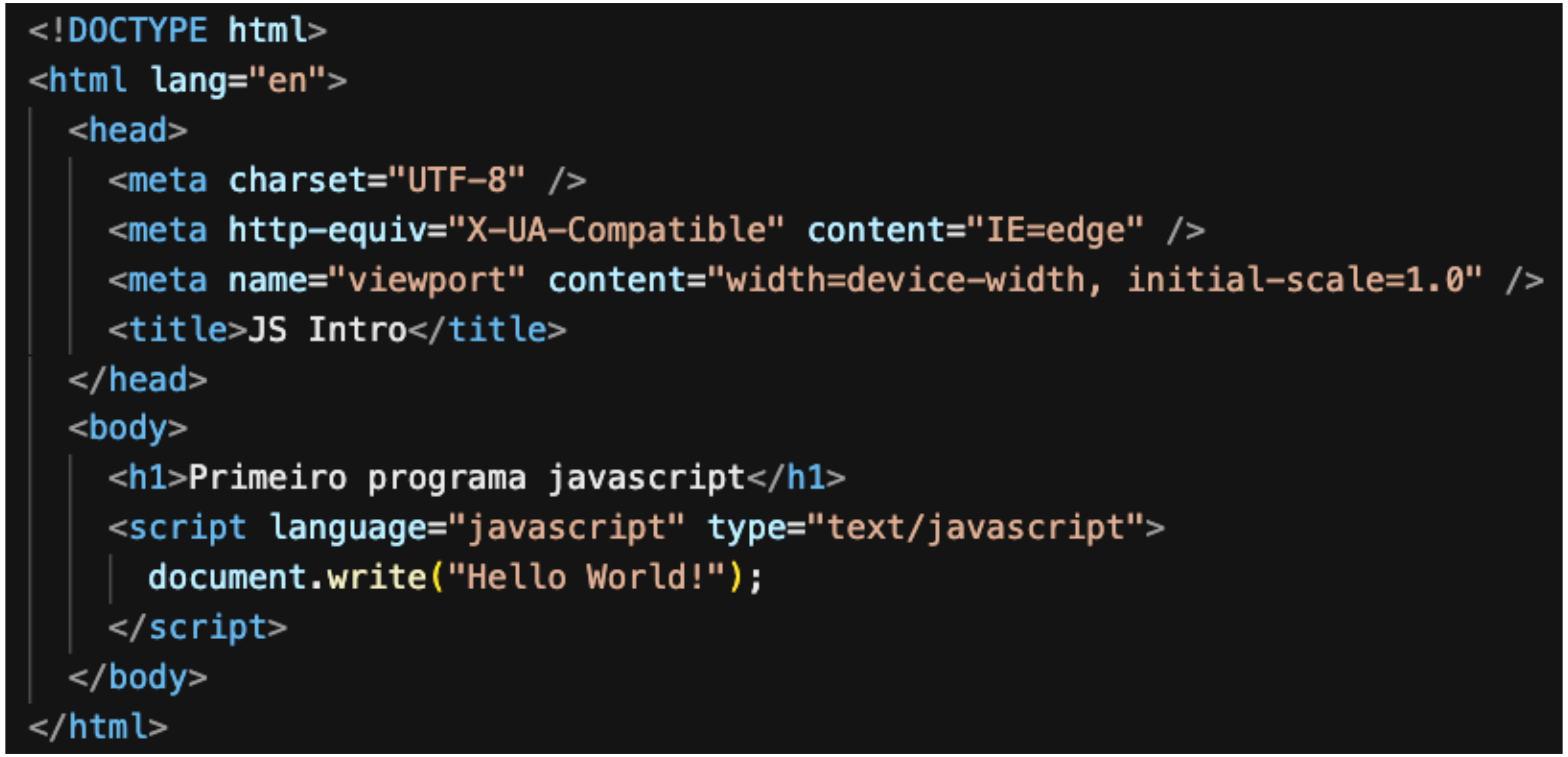
- Javascript;
- Primeiro programa javascript;
- Declaração de variáveis;
- Operadores aritméticos, relacionais e lógicos;
- Estruturas de decisão e repetição;
- Funções;
- Eventos;
- Objetos;
- Arrays.

Javascript

- É a linguagem mais popular do mundo
- É uma linguagem de programação interpretada.
- É uma linguagem para desenvolvimento web.
- É uma linguagem de front-end e back-end
- É suportada por todos os browsers modernos, não sendo necessário preparar qualquer ambiente de desenvolvimento.
- Permite criar páginas dinâmicas, manipulando o html em tempo real.
- Existem várias bibliotecas e frameworks javascript, e.g. Angular, React, jQuery, Pode.js, tem..

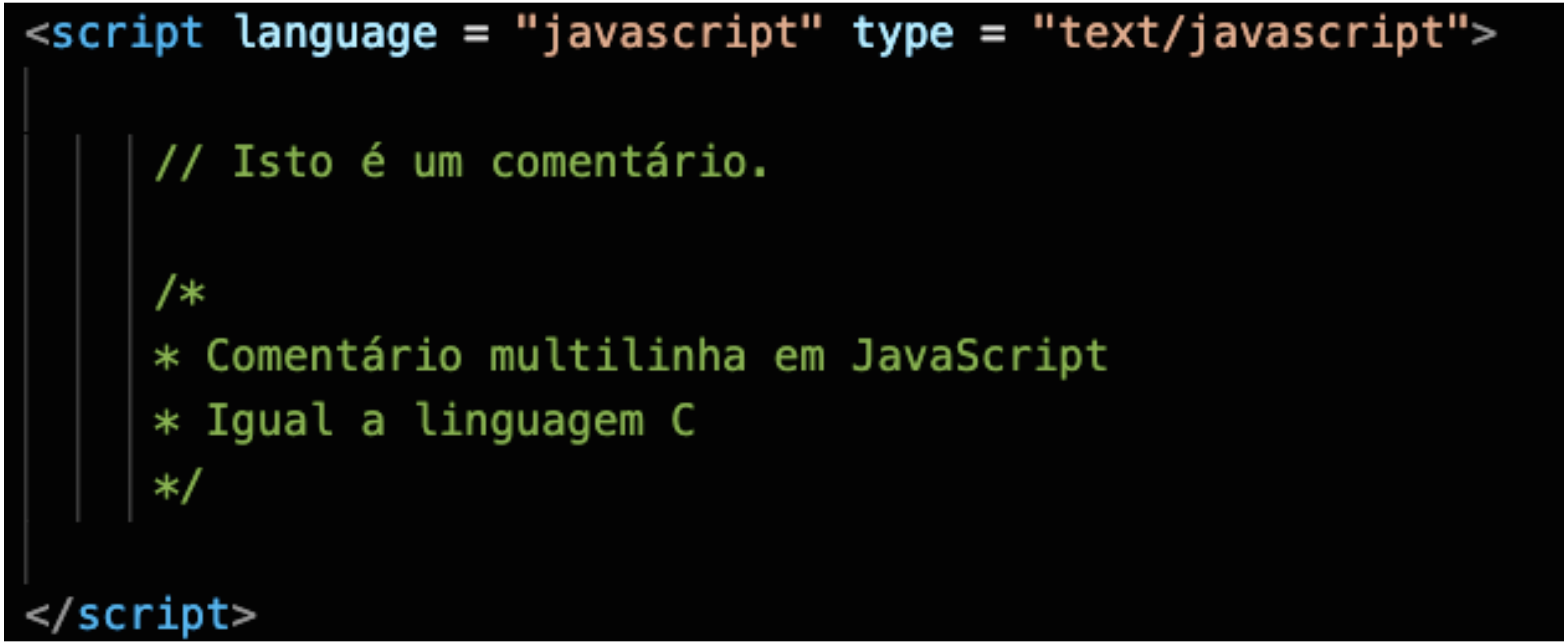
Javascript

- Pode ser implementado entre as tags HTML <script></script> emqualquer lado numa página web.



```html
<!DOCTYPE html>
<html lang="en">
  <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>JS Intro</title>
  </head>
  <body>
    <h1>Primeiro programa javascript</h1>
    <script language="javascript" type="text/javascript">
      document.write("Hello World!");
    </script>
  </body>
</html>
```

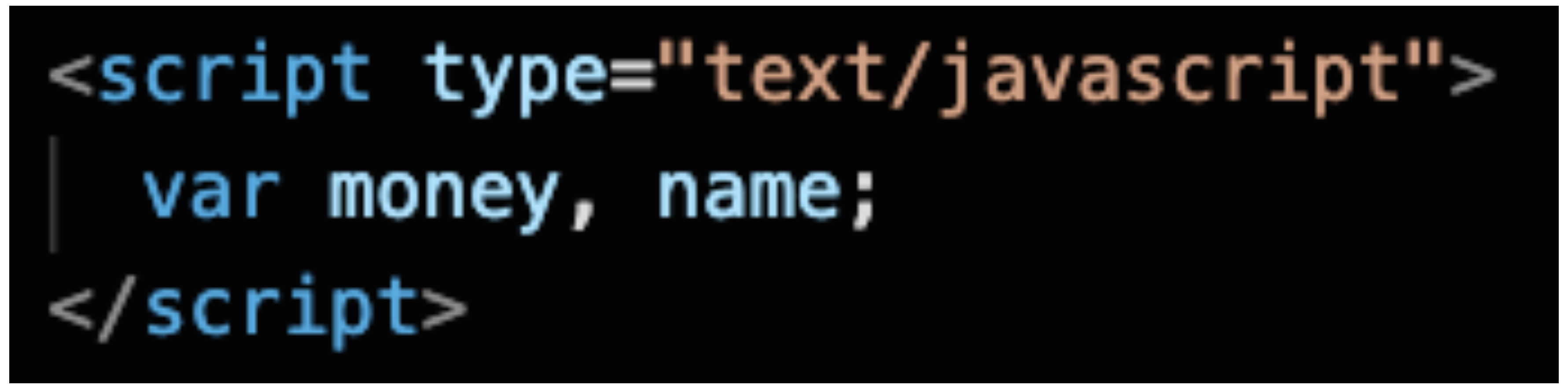
Comentários



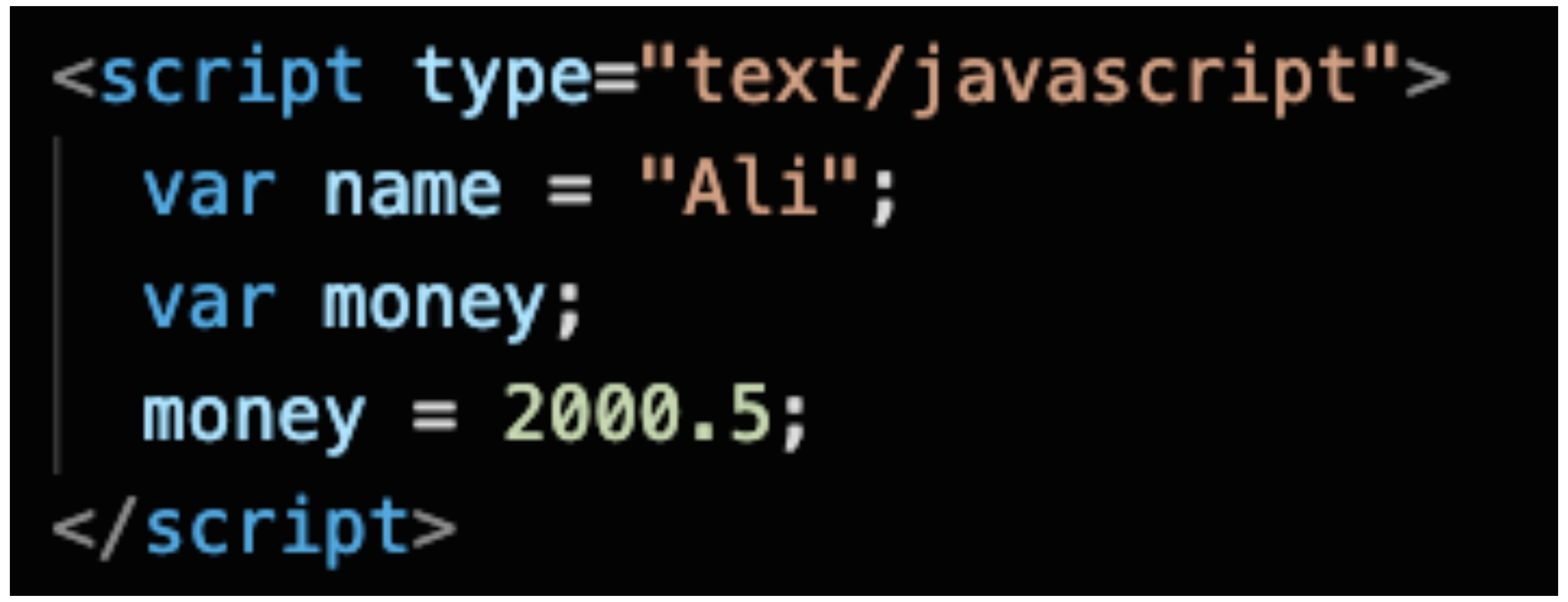
```
<script language = "javascript" type = "text/javascript">

        // Isto é um comentário.

        /*
        * Comentário multilinha em JavaScript
        * Igual a linguagem C
        */

</script>
```

Declaração e variáveis

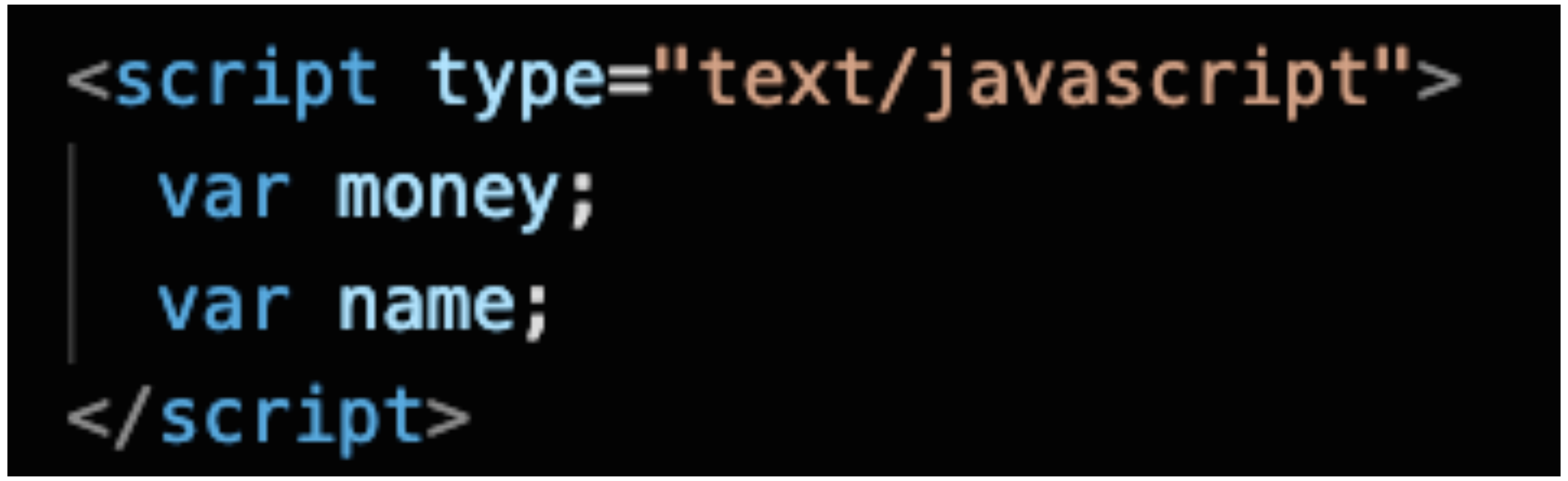
- O ponto e vírgula a separar as instruções não é obrigatório.
- É uma linguagem case-sensitive.



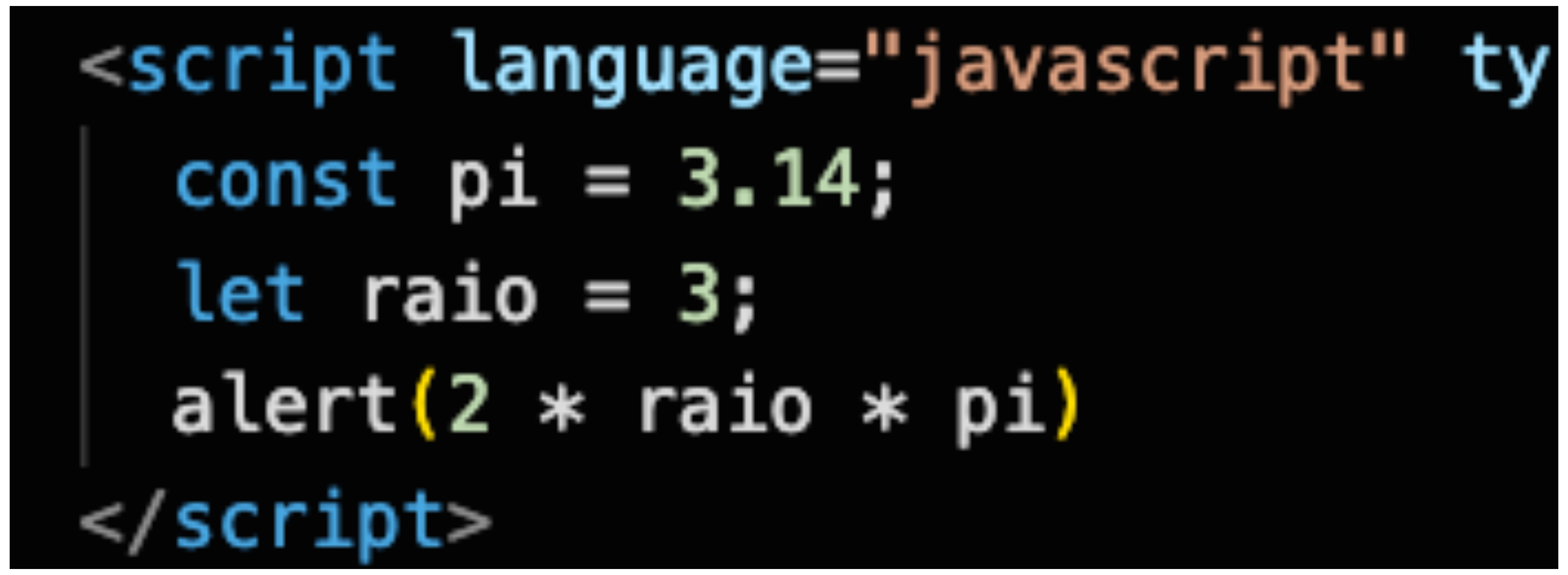
```
<script type="text/javascript">
  var money, name;
</script>
```



```
<script type="text/javascript">
  var name = "Ali";
  var money;
  money = 2000.5;
</script>
```

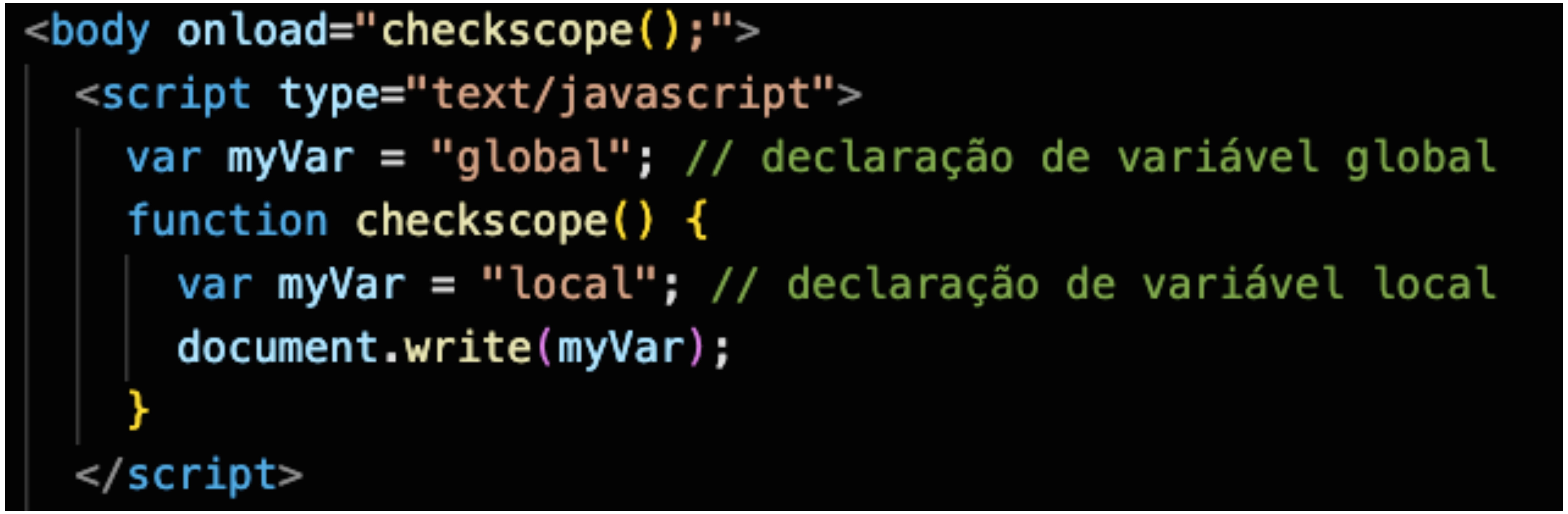


```
<script type="text/javascript">
  var money;
  var name;
</script>
```



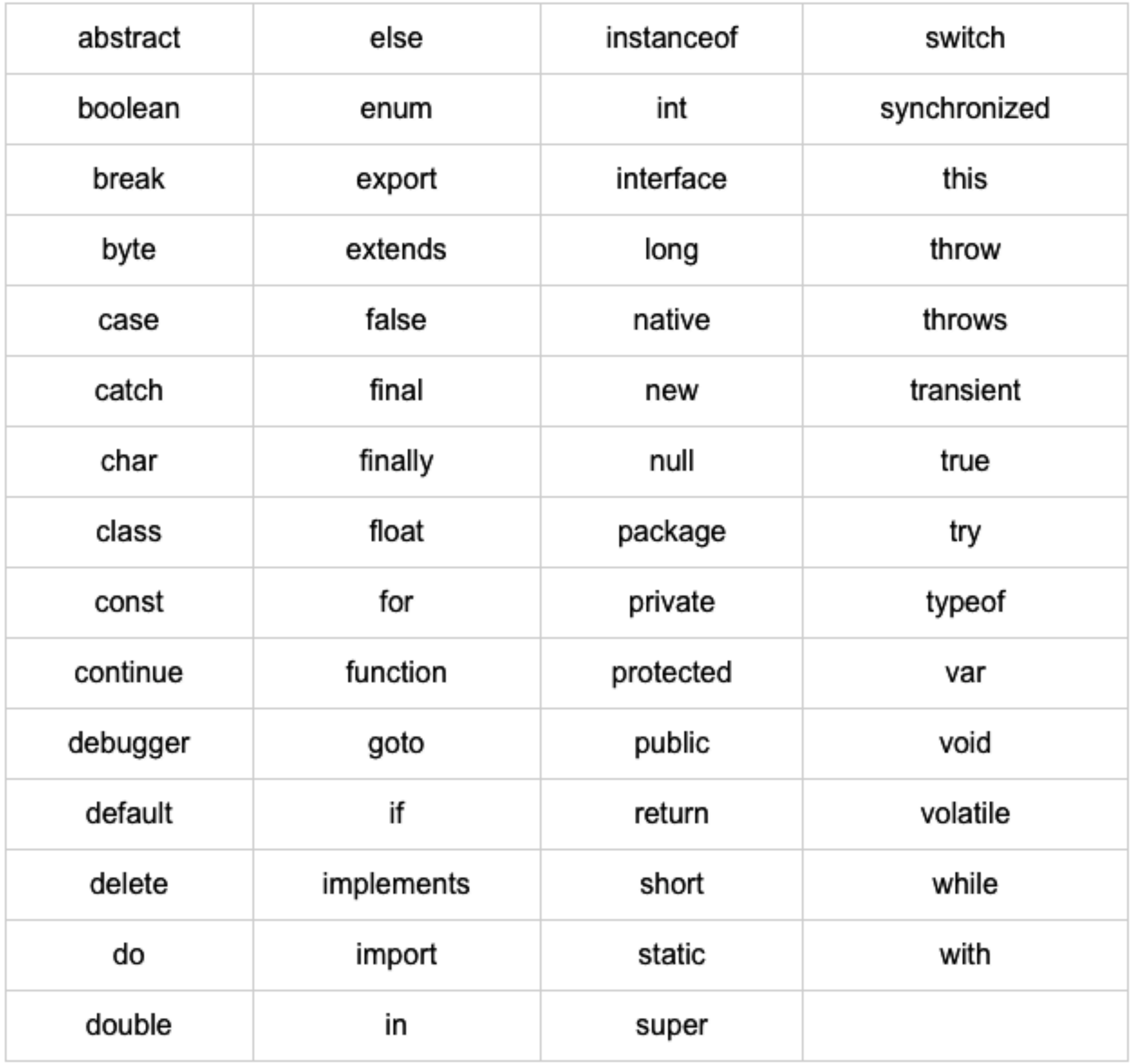
```
<script language="javascript" ty
  const pi = 3.14;
  let raio = 3;
  alert(2 * raio * pi)
</script>
```

Variáveis locais e globais



```
<body onload="checkscope();">
  <script type="text/javascript">
    var myVar = "global"; // declaração de variável global
    function checkscope() {
      var myVar = "local"; // declaração de variável local
      document.write(myVar);
    }
  </script>
```

Palavras reservadas

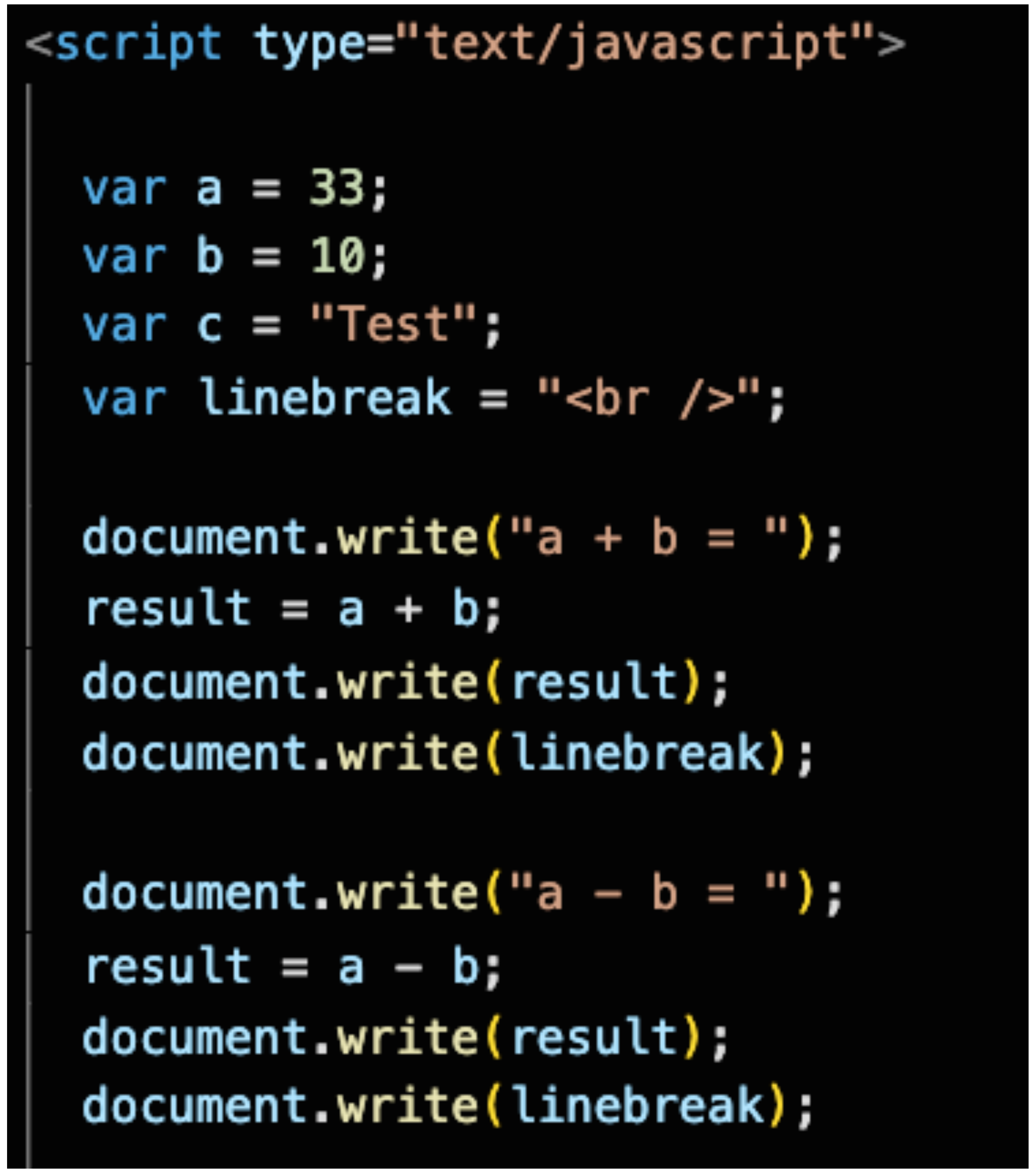


abstract	else	instanceof	switch
boolean	enum	int	synchronized
break	export	interface	this
byte	extends	long	throw
case	false	native	throws
catch	final	new	transient
char	finally	null	true
class	float	package	try
const	for	private	typeof
continue	function	protected	var
debugger	goto	public	void
default	if	return	volatile
delete	implements	short	while
do	import	static	with
double	in	super	

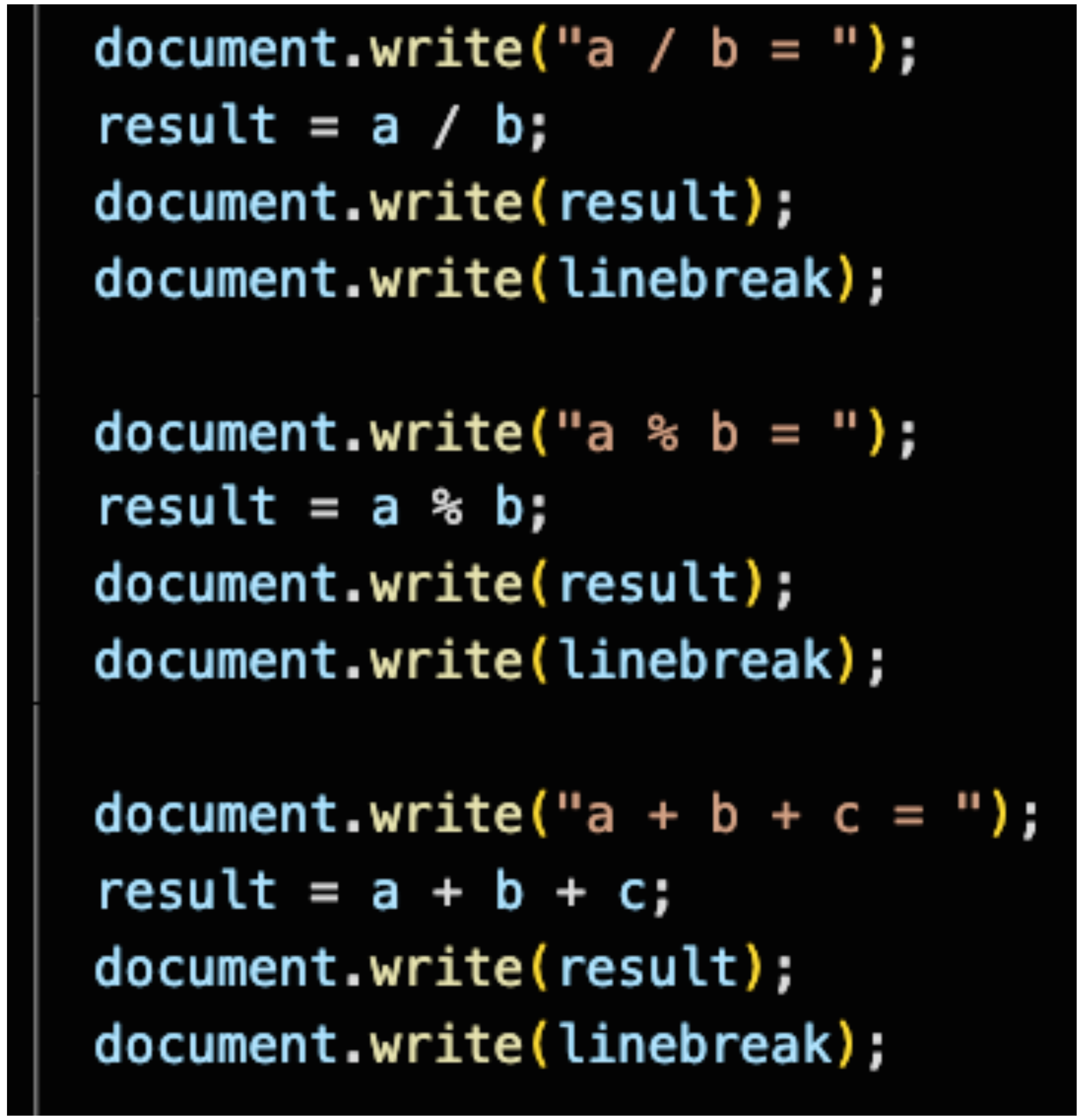
Operadores aritméticos

Operador	Descrição
+	(Adição) adiciona dois operandos
-	(Subtração) subtrai o segundo operando ao primeiro
*	(Multiplicação) Multiplica dois operandos
/	(Divisão) Divide o numerador ao denominador
%	Especifica como os controlos do audio são mostrados
++	Incrementa um a número inteiro
--	Decrementa um a número inteiro

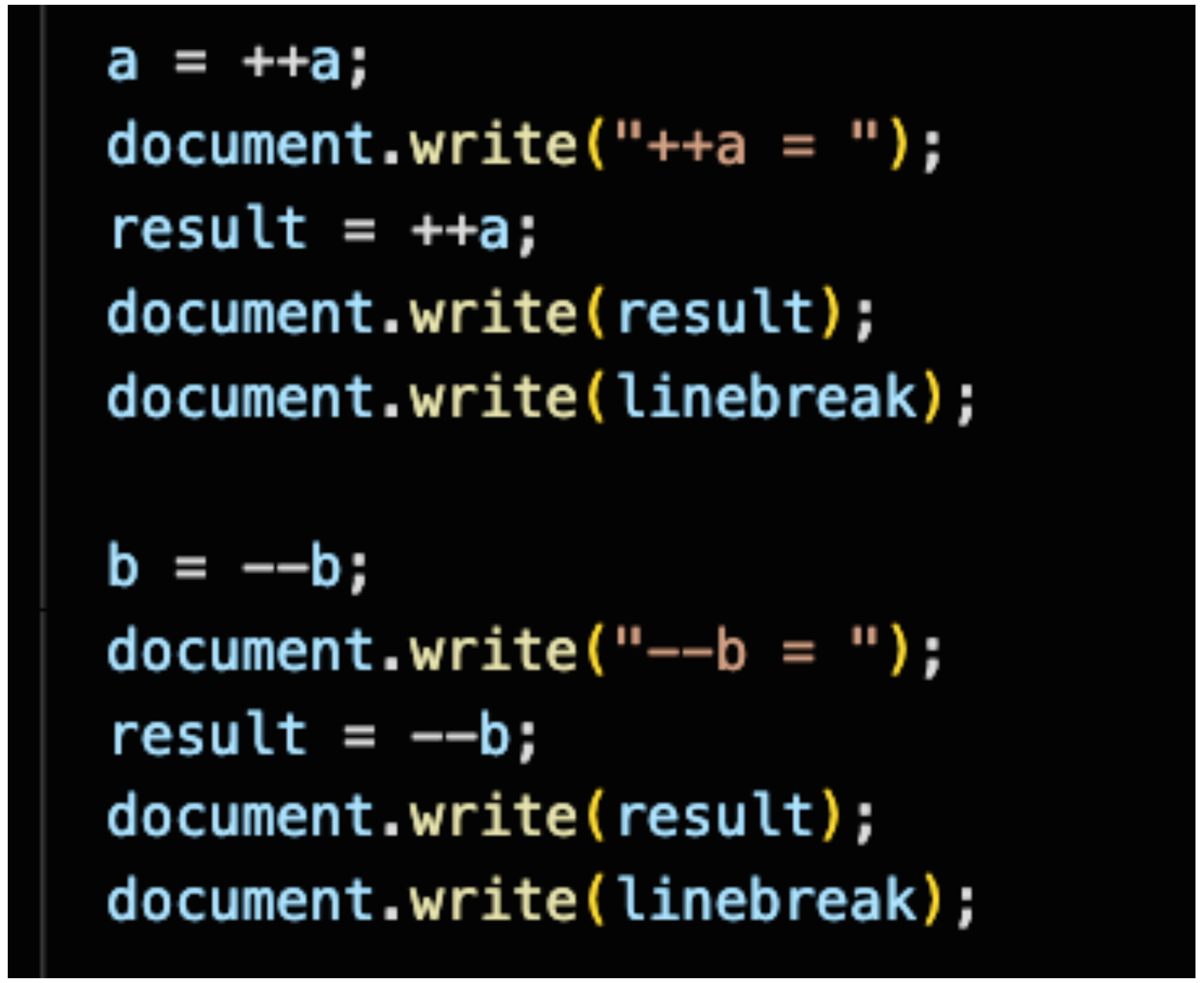
Operadores aritméticos



```
<script type="text/javascript">

  var a = 33;
  var b = 10;
  var c = "Test";
  var linebreak = "<br />";

  document.write("a + b = ");
  result = a + b;
  document.write(result);
  document.write(linebreak);

  document.write("a - b = ");
  result = a - b;
  document.write(result);
  document.write(linebreak);
```



```
  document.write("a / b = ");
  result = a / b;
  document.write(result);
  document.write(linebreak);

  document.write("a % b = ");
  result = a % b;
  document.write(result);
  document.write(linebreak);

  document.write("a + b + c = ");
  result = a + b + c;
  document.write(result);
  document.write(linebreak);
```

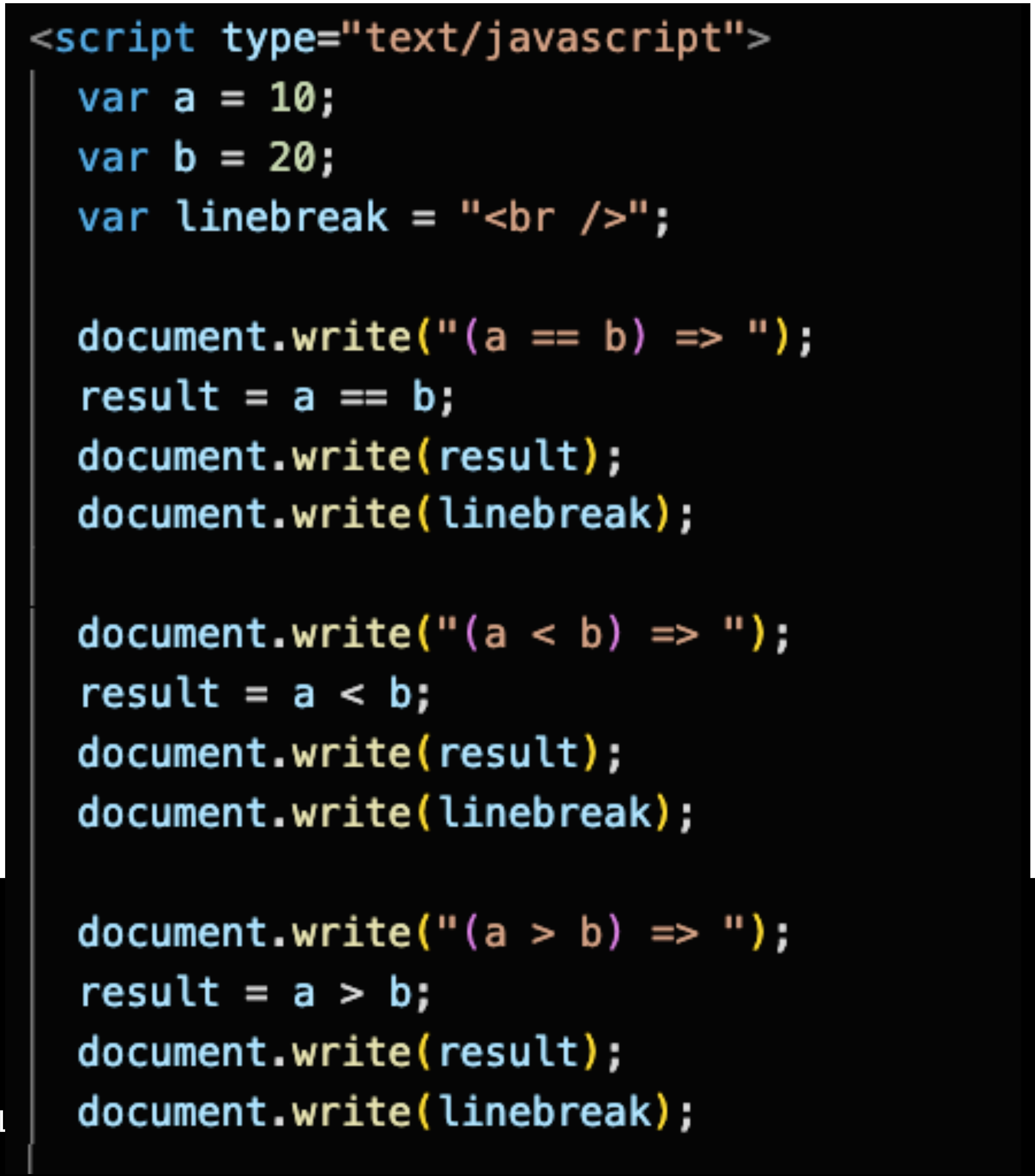


```
  a = ++a;
  document.write("++a = ");
  result = ++a;
  document.write(result);
  document.write(linebreak);

  b = --b;
  document.write("--b = ");
  result = --b;
  document.write(result);
  document.write(linebreak);
```

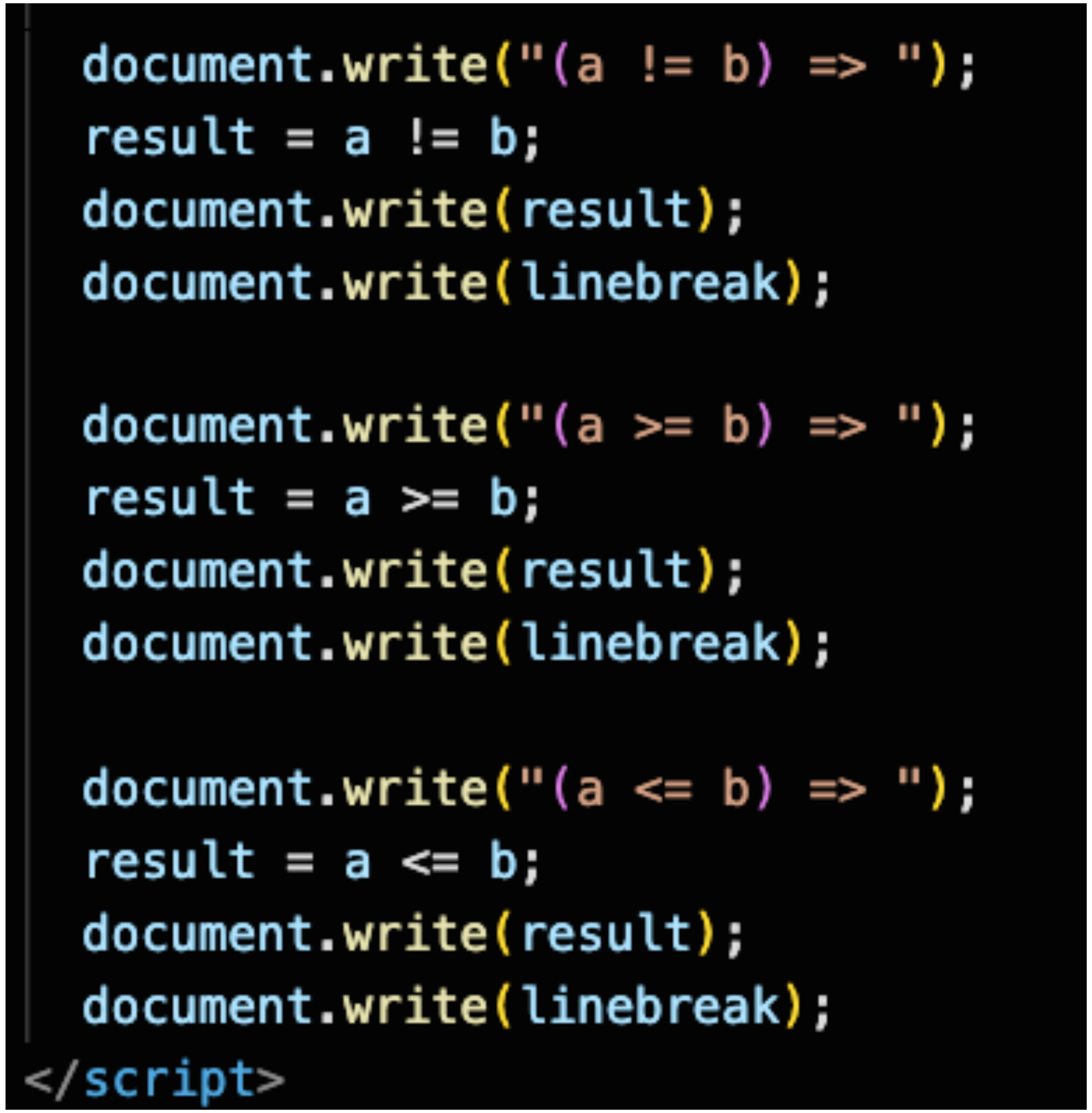
Operadores relacionais

Operador	Descrição
`==`	(Igual) Avalia se os operandos são iguais ou não.Se sim a condição é verdadeira.
`!=`	(Diferente) Avalia se os operandos são diferentes ou não.Se sim a condição é verdadeira.
`>`	(Maior) Avalia se o operando da esquerda é maior que o da direita.Se sim a condição é verdadeira.
`<`	(Menor) Avalia se o operando da esquerda é menor que o da direita.Se sim a condição é verdadeira.
`>=`	(Maior ou igual) Avalia se o operando da esquerda é maior ou igual que o da direita.Se sim a condição é verdadeira
`<=`	(Maior ou igual) Avalia se o operando da esquerda é menor ou igual que o da direita.Se sim a condição é verdadeira

Operadores relacionais



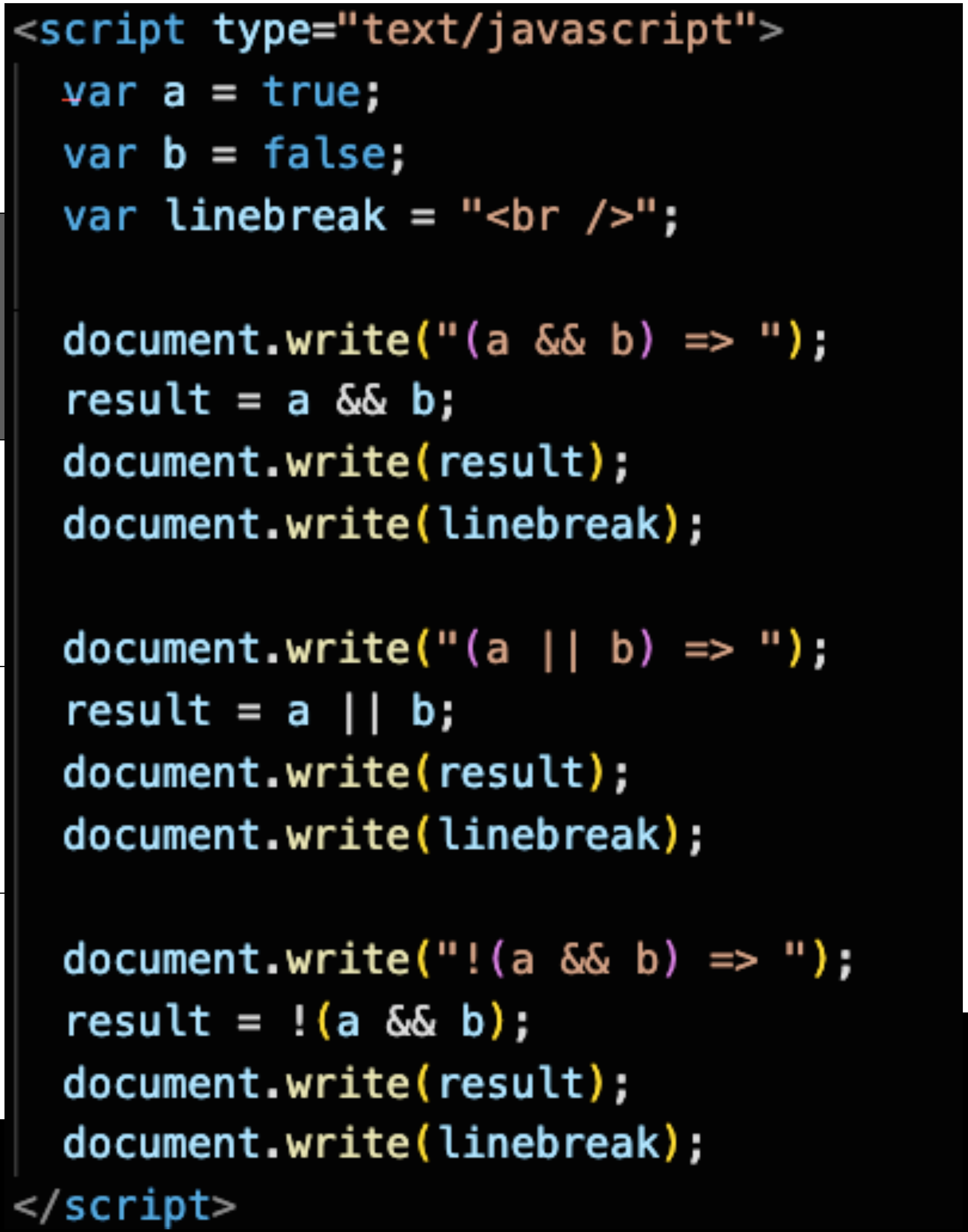
```
<script type="text/javascript">
  var a = 10;
  var b = 20;
  var linebreak = "<br />";

  document.write("(a == b) => ");
  result = a == b;
  document.write(result);
  document.write(linebreak);

  document.write("(a < b) => ");
  result = a < b;
  document.write(result);
  document.write(linebreak);

  document.write("(a > b) => ");
  result = a > b;
  document.write(result);
  document.write(linebreak);
```



```
  document.write("(a != b) => ");
  result = a != b;
  document.write(result);
  document.write(linebreak);

  document.write("(a >= b) => ");
  result = a >= b;
  document.write(result);
  document.write(linebreak);

  document.write("(a <= b) => ");
  result = a <= b;
  document.write(result);
  document.write(linebreak);
</script>
```

Operadores lógicos

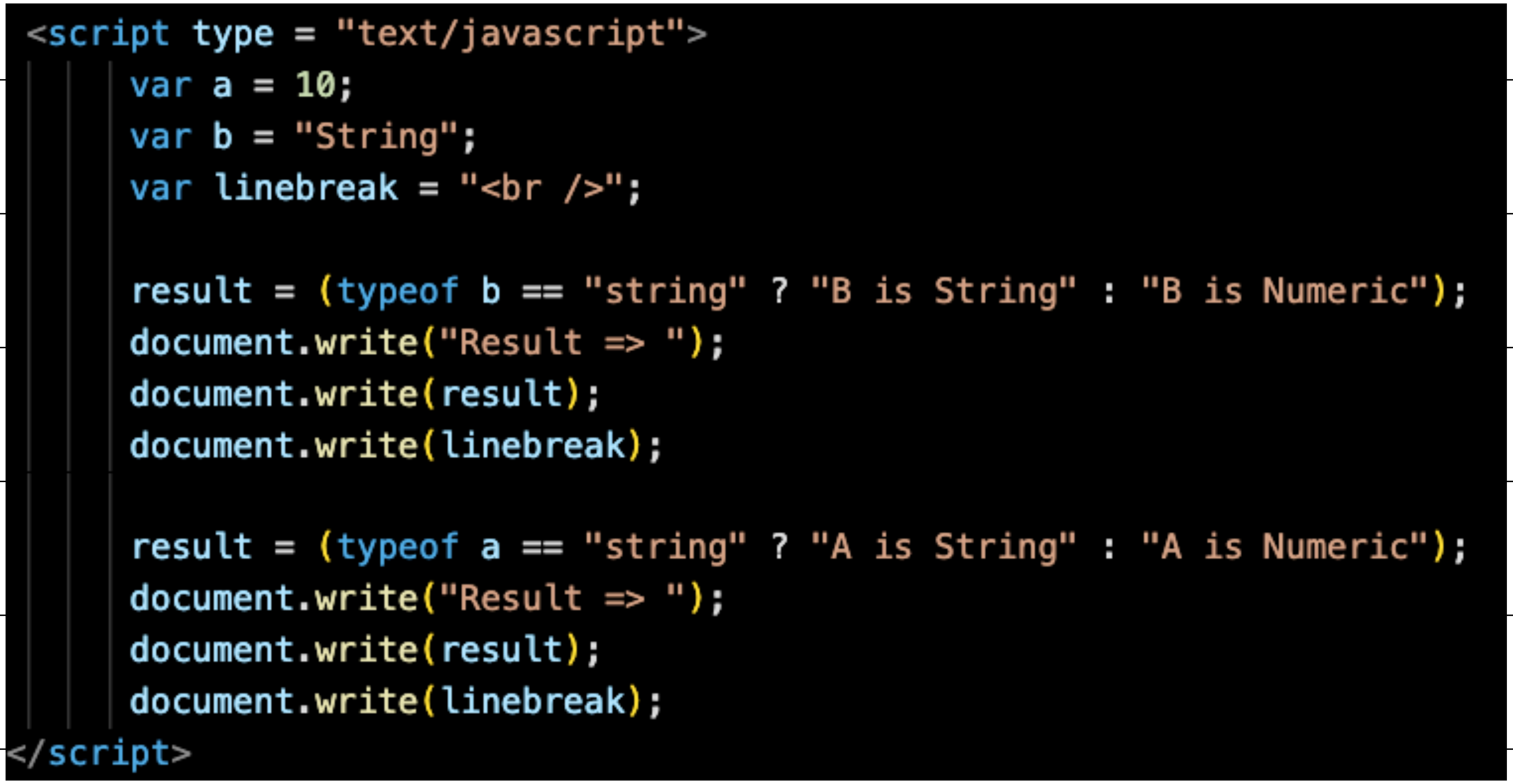
Operador	Descrição
&&	(AND lógico).
||	(OR lógico).
!	(NOT lógico).



```
<script type="text/javascript">
  var a = true;
  var b = false;
  var linebreak = "<br />";

  document.write("(a && b) => ");
  result = a && b;
  document.write(result);
  document.write(linebreak);

  document.write("(a || b) => ");
  result = a || b;
  document.write(result);
  document.write(linebreak);

  document.write("!(a && b) => ");
  result = !(a && b);
  document.write(result);
  document.write(linebreak);
</script>
```

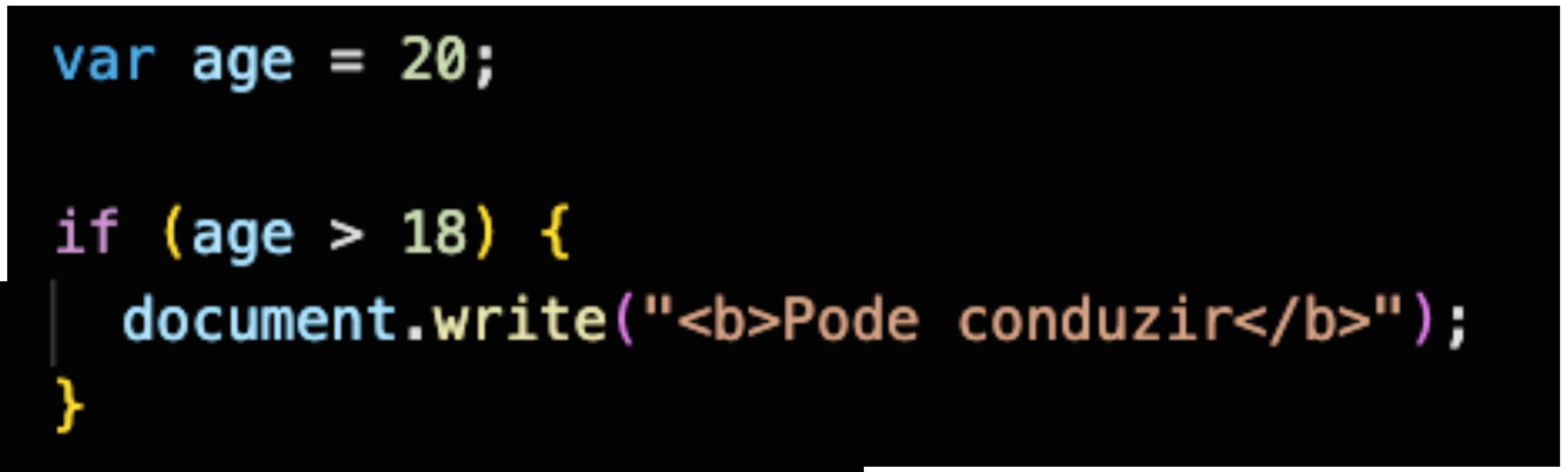
Operador typeof

Tipo	Valor retornado
Number	“number”
String	“string”
Boolean	“boolean”
Object	“object”
Funcion	“function”
Undefined	“undefined”
Null	“Object”

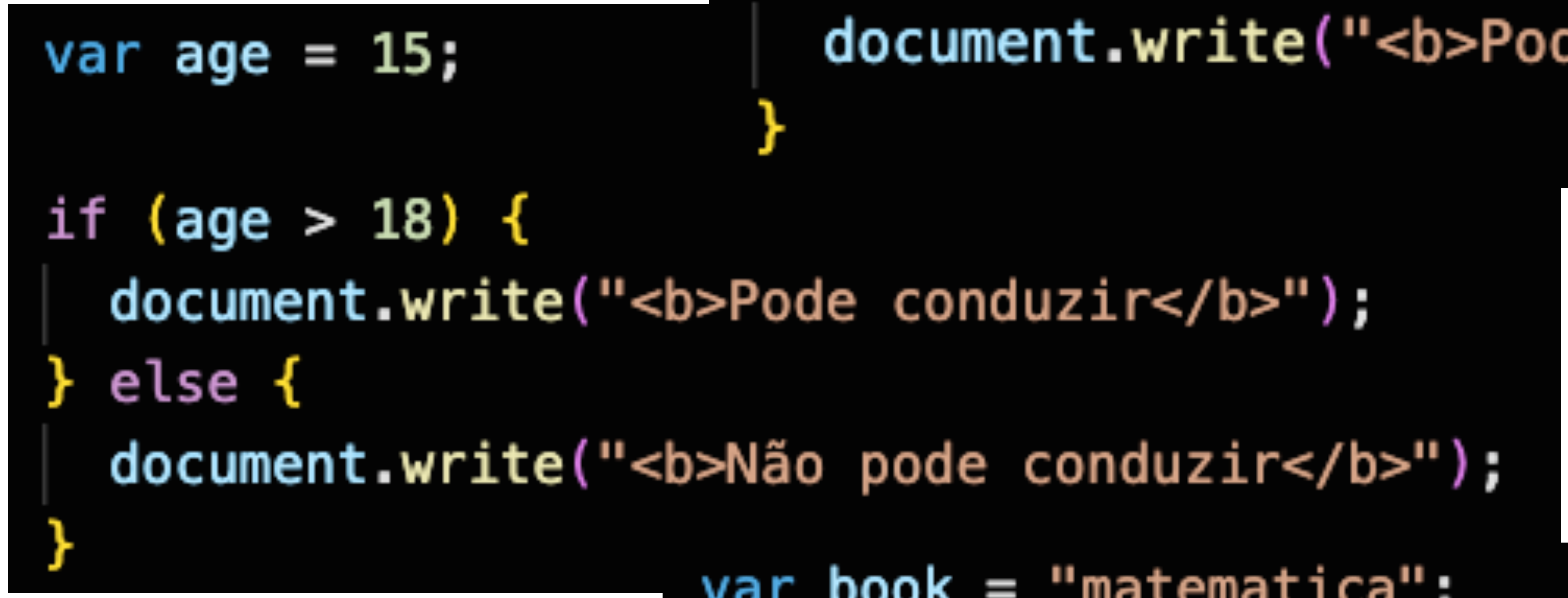


```
<script type = "text/javascript">
        var a = 10;
        var b = "String";
        var linebreak = "<br />";

        result = (typeof b == "string" ? "B is String" : "B is Numeric");
        document.write("Result => ");
        document.write(result);
        document.write(linebreak);

        result = (typeof a == "string" ? "A is String" : "A is Numeric");
        document.write("Result => ");
        document.write(result);
        document.write(linebreak);
</script>
```

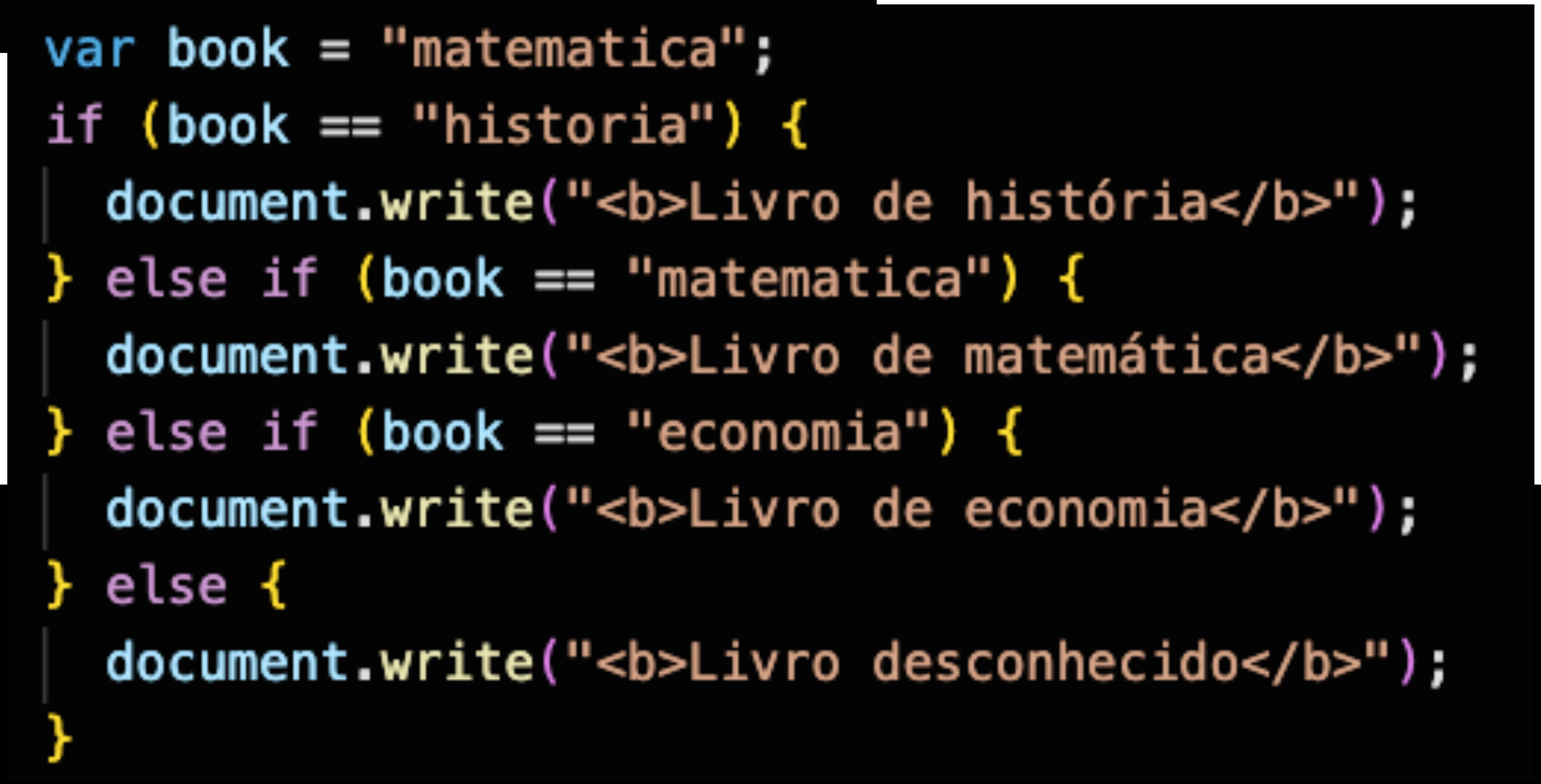
Estruturas de decisão



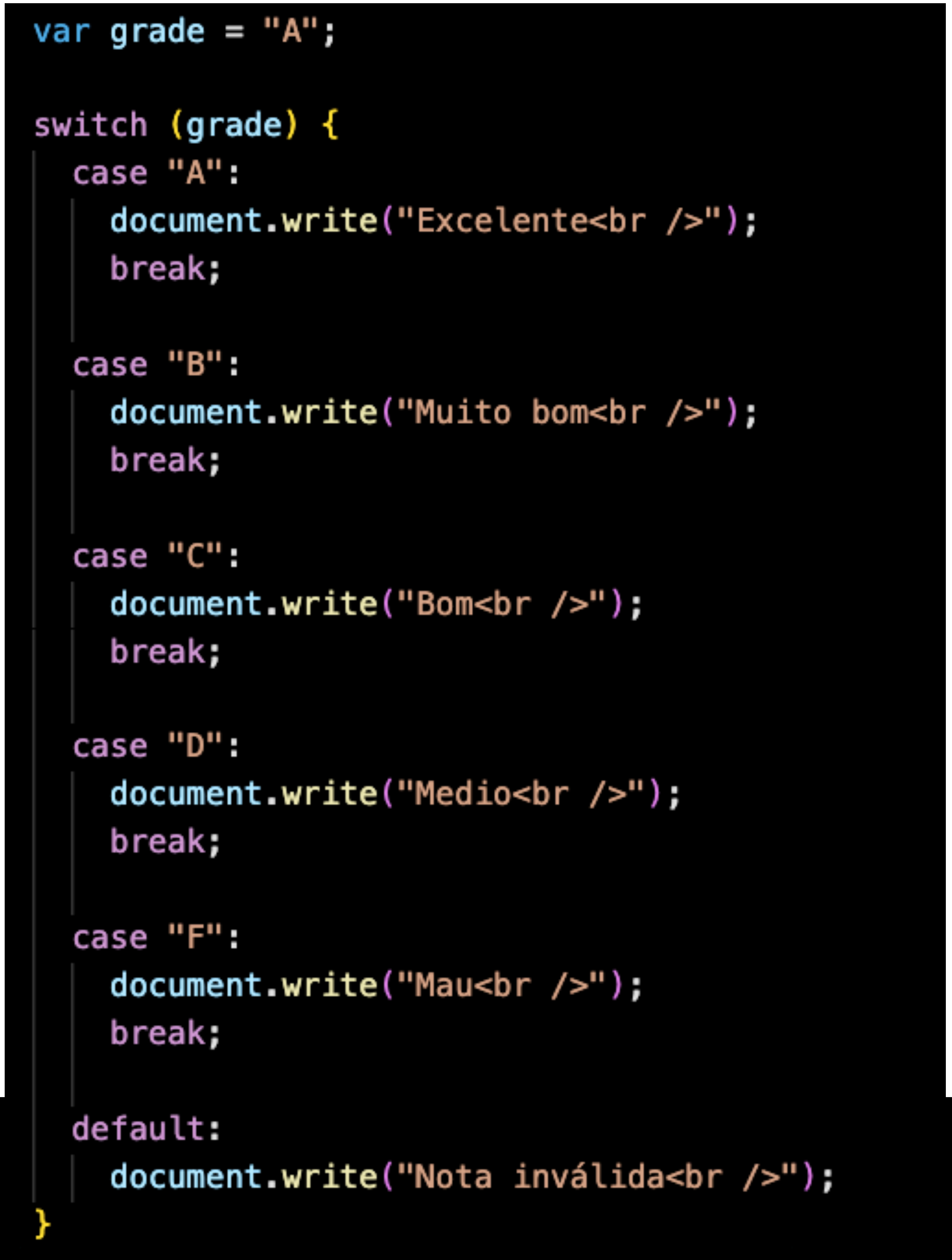
```
var age = 20;

if (age > 18) {
  document.write("<b>Pode conduzir</b>");
}
```



```
var age = 15;

if (age > 18) {
  document.write("<b>Pode conduzir</b>");
} else {
  document.write("<b>Não pode conduzir</b>");
}
```

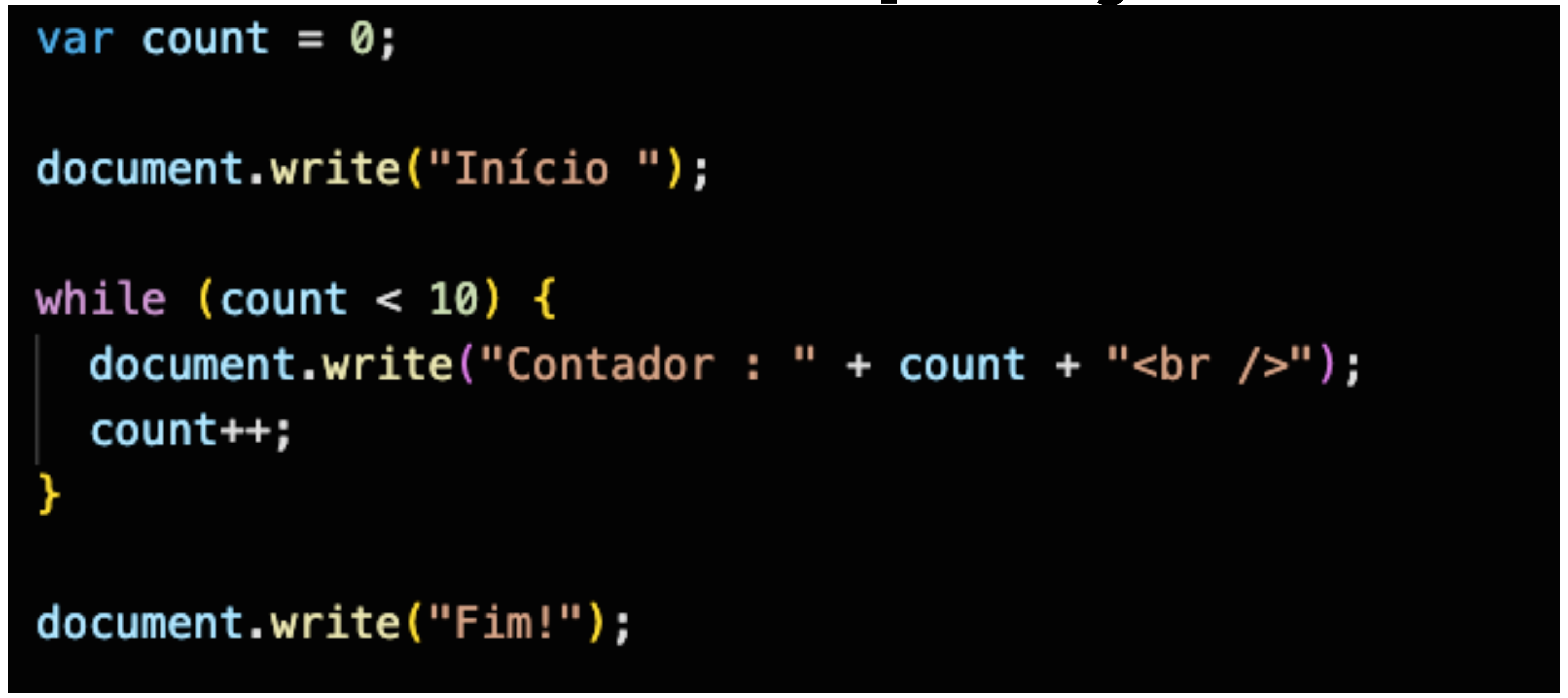


```
var book = "matematica";
if (book == "historia") {
  document.write("<b>Livro de história</b>");
} else if (book == "matematica") {
  document.write("<b>Livro de matemática</b>");
} else if (book == "economia") {
  document.write("<b>Livro de economia</b>");
} else {
  document.write("<b>Livro desconhecido</b>");
}
```

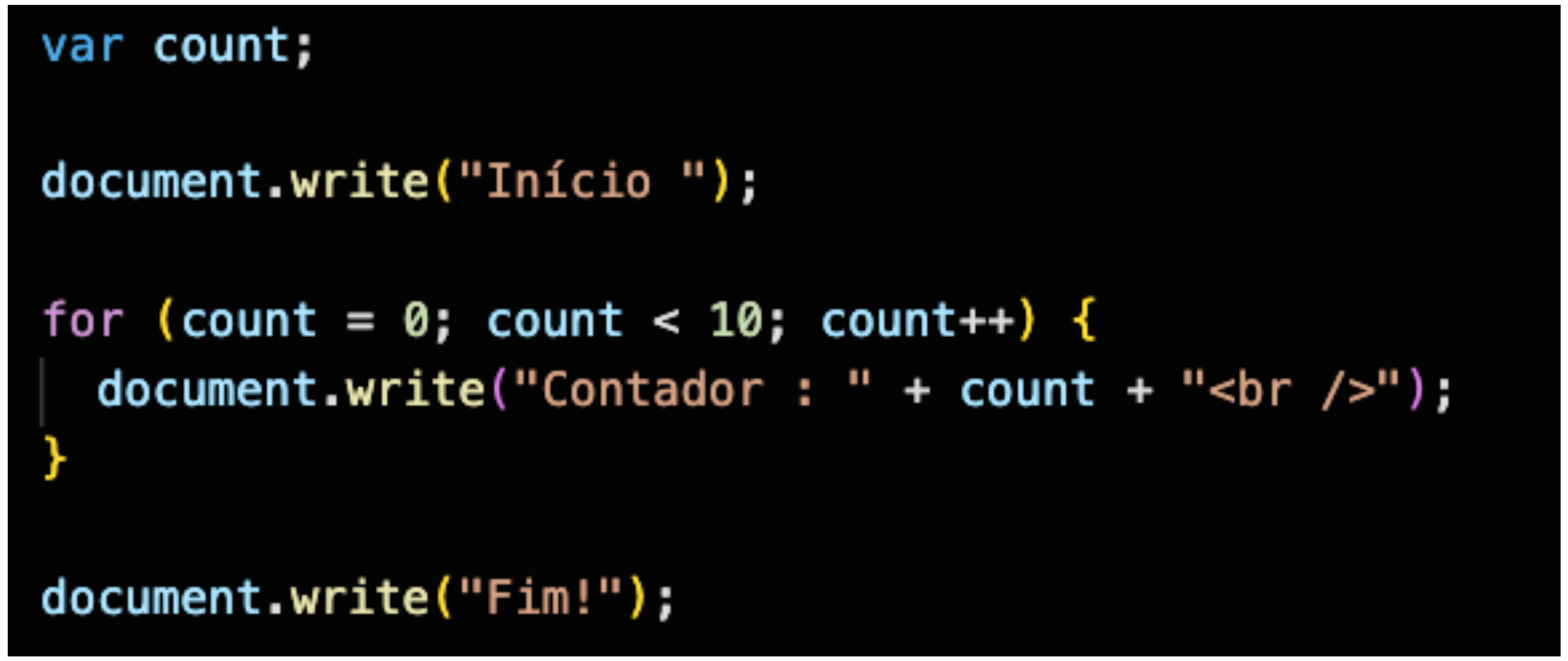


```
var grade = "A";

switch (grade) {
  case "A":
    document.write("Excelente<br />");
    break;

  case "B":
    document.write("Muito bom<br />");
    break;

  case "C":
    document.write("Bom<br />");
    break;

  case "D":
    document.write("Medio<br />");
    break;

  case "F":
    document.write("Mau<br />");
    break;

  default:
    document.write("Nota inválida<br />");
}
```

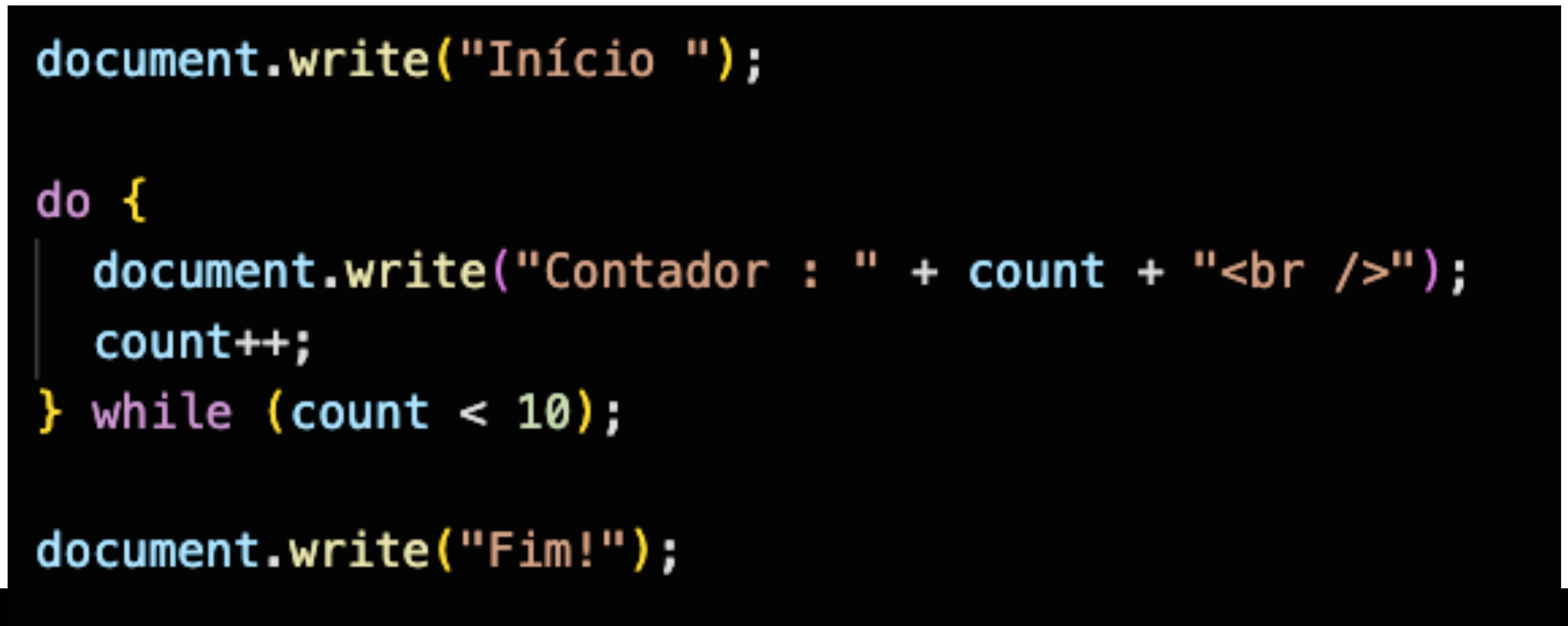
Estruturas de repetição



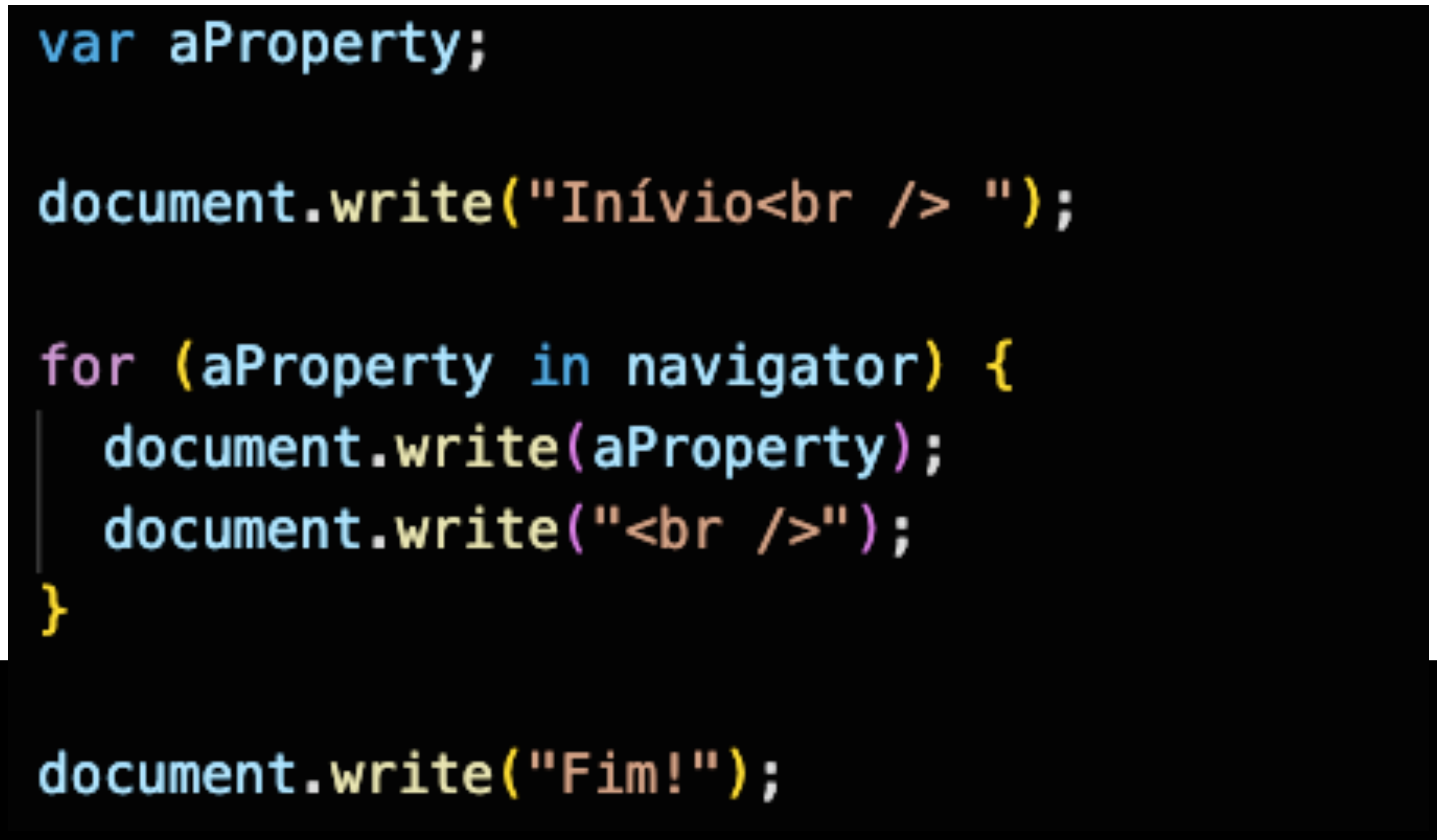
```
var count = 0;

document.write("Início ");

while (count < 10) {
  document.write("Contador : " + count + "<br />");
  count++;
}

document.write("Fim!");
```



```
var count;

document.write("Início ");

for (count = 0; count < 10; count++) {
  document.write("Contador : " + count + "<br />");
}

document.write("Fim!");
```

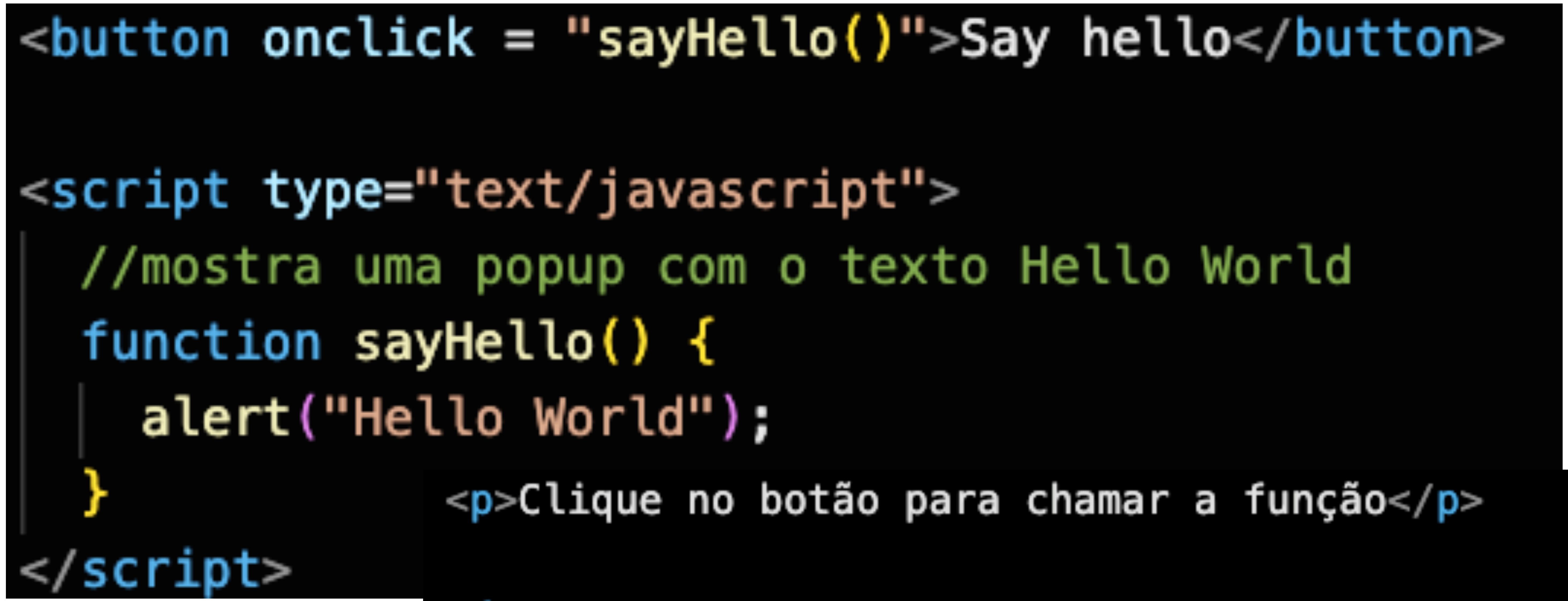


```
document.write("Início ");

do {
  document.write("Contador : " + count + "<br />");
  count++;
} while (count < 10);

document.write("Fim!");
```

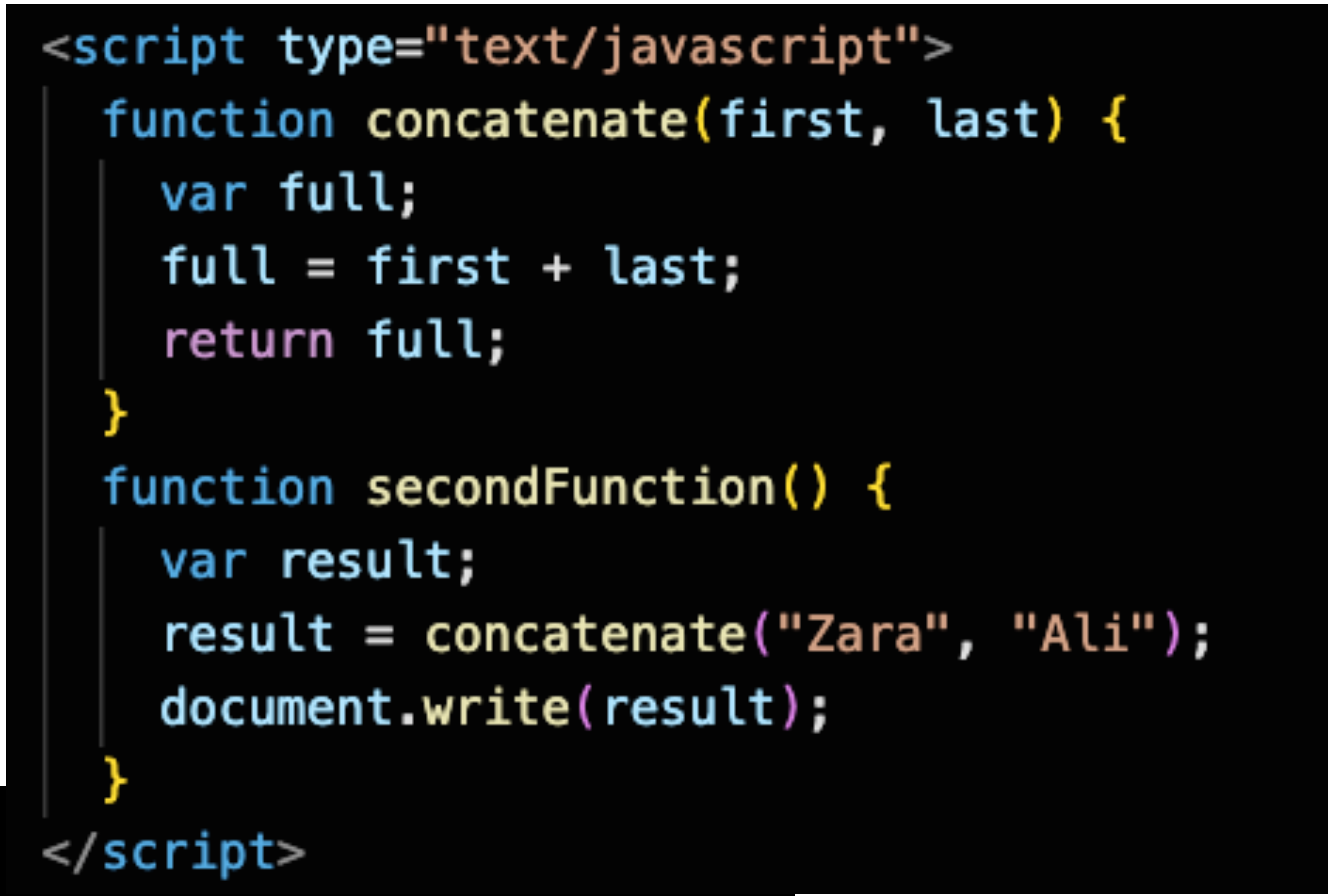


```
var aProperty;

document.write("Inívio<br /> ");

for (aProperty in navigator) {
  document.write(aProperty);
  document.write("<br />");
}

document.write("Fim!");
```

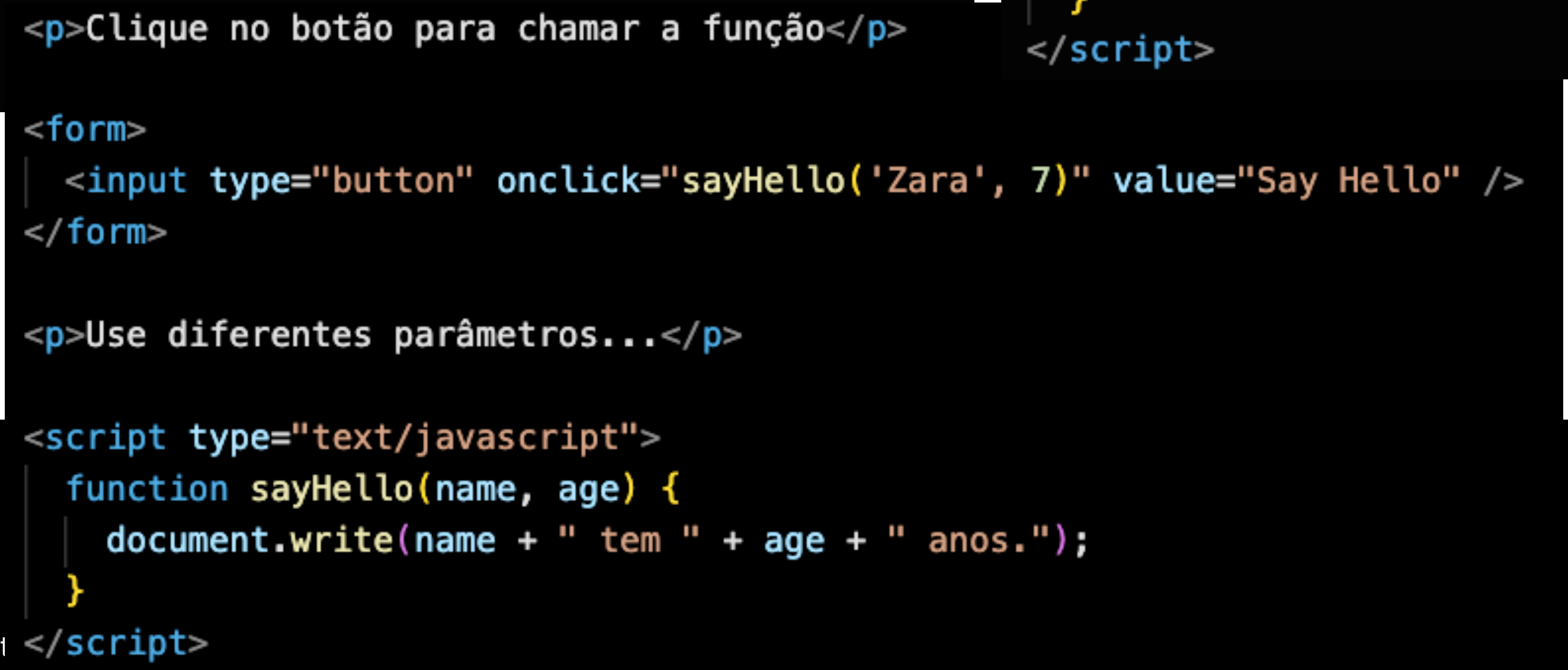
Funções



```html
<button onclick = "sayHello()">Say hello</button>

<script type="text/javascript">
  //mostra uma popup com o texto Hello World
  function sayHello() {
    alert("Hello World");
  }
</script>
```

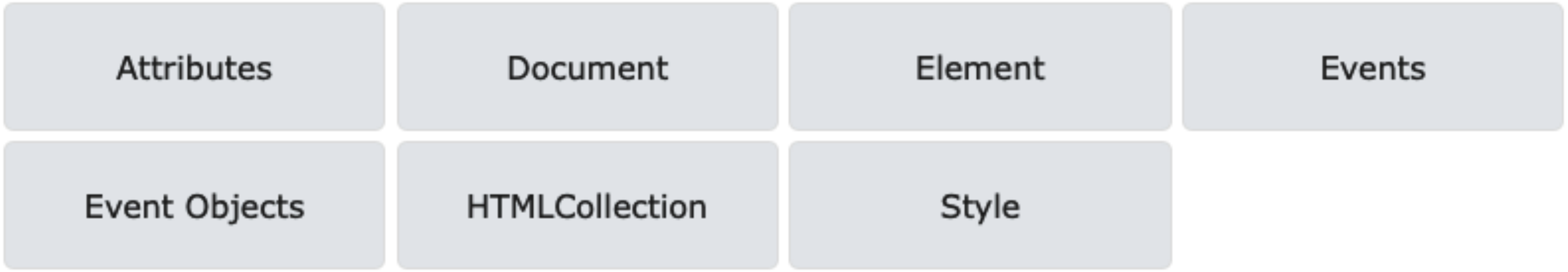


```html
<script type="text/javascript">
  function concatenate(first, last) {
    var full;
    full = first + last;
    return full;
  }
  function secondFunction() {
    var result;
    result = concatenate("Zara", "Ali");
    document.write(result);
  }
</script>
```



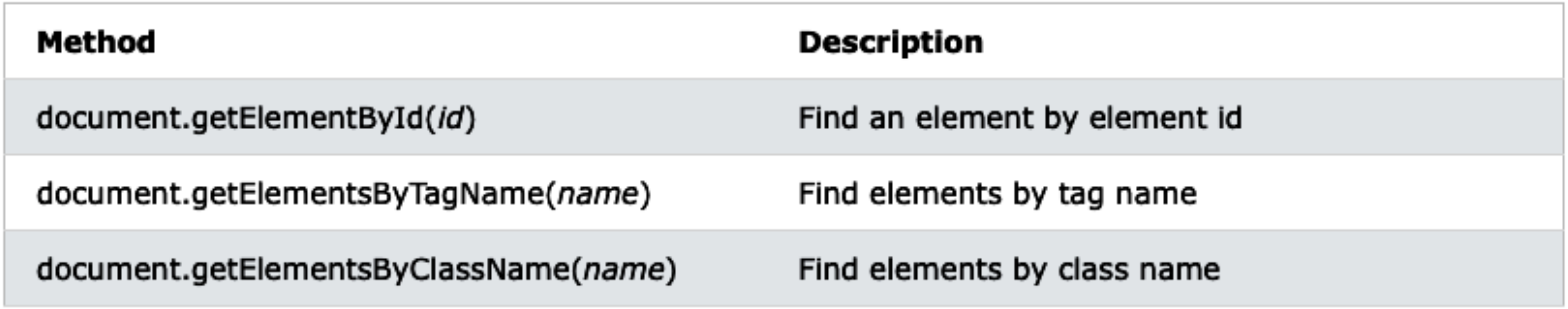
```html
<p>Clique no botão para chamar a função</p>

<form>
  <input type="button" onclick="sayHello('Zara', 7)" value="Say Hello" />
</form>

<p>Use diferentes parâmetros...</p>

<script type="text/javascript">
  function sayHello(name, age) {
    document.write(name + " tem " + age + " anos.");
  }
</script>
```

Referencia DOM HTML



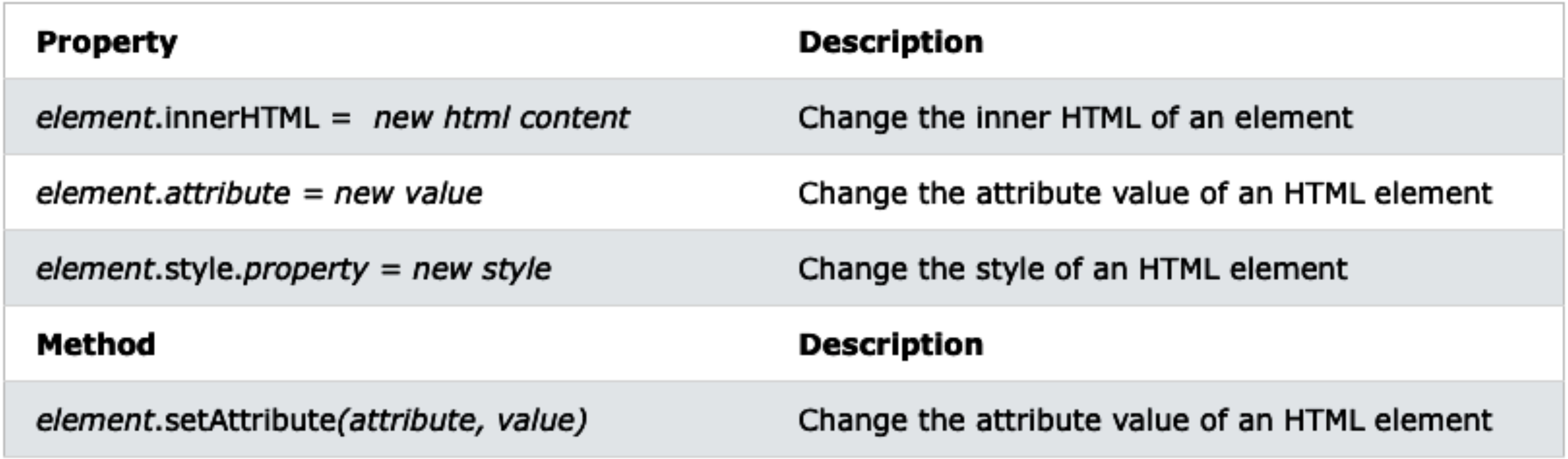
https://www.w3schools.com/jsref/default.asp

Referencia DOM - Procurar elementos



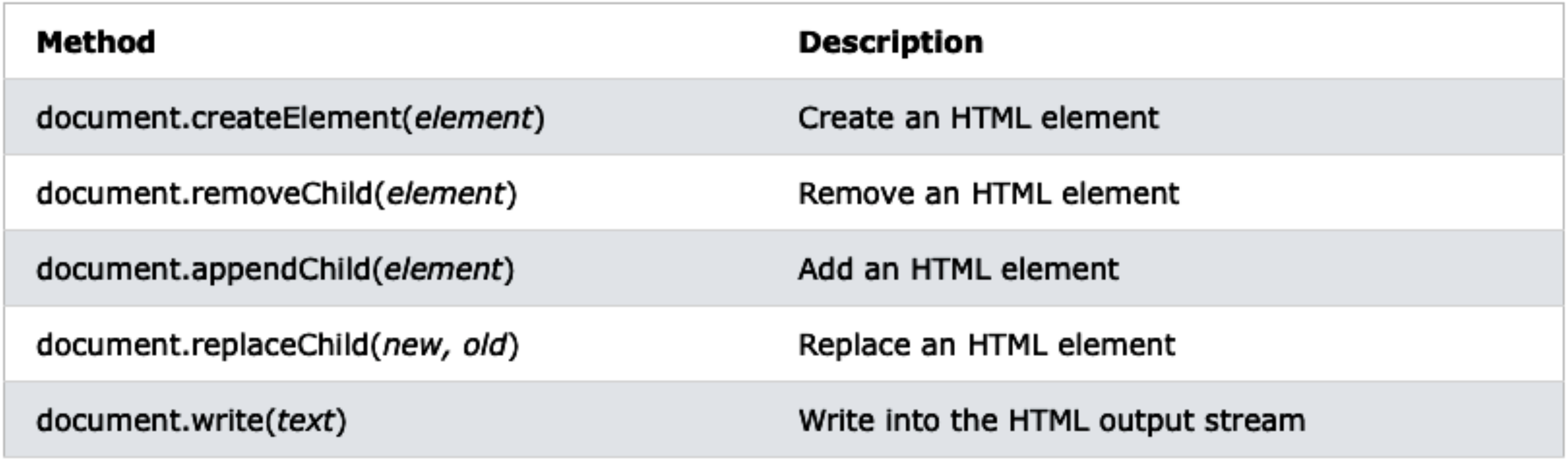
Method	**Description**
document.getElementById(*id*)	Find an element by element id
document.getElementsByTagName(*name*)	Find elements by tag name
document.getElementsByClassName(*name*)	Find elements by class name

Referencia DOM - Manipular elementos



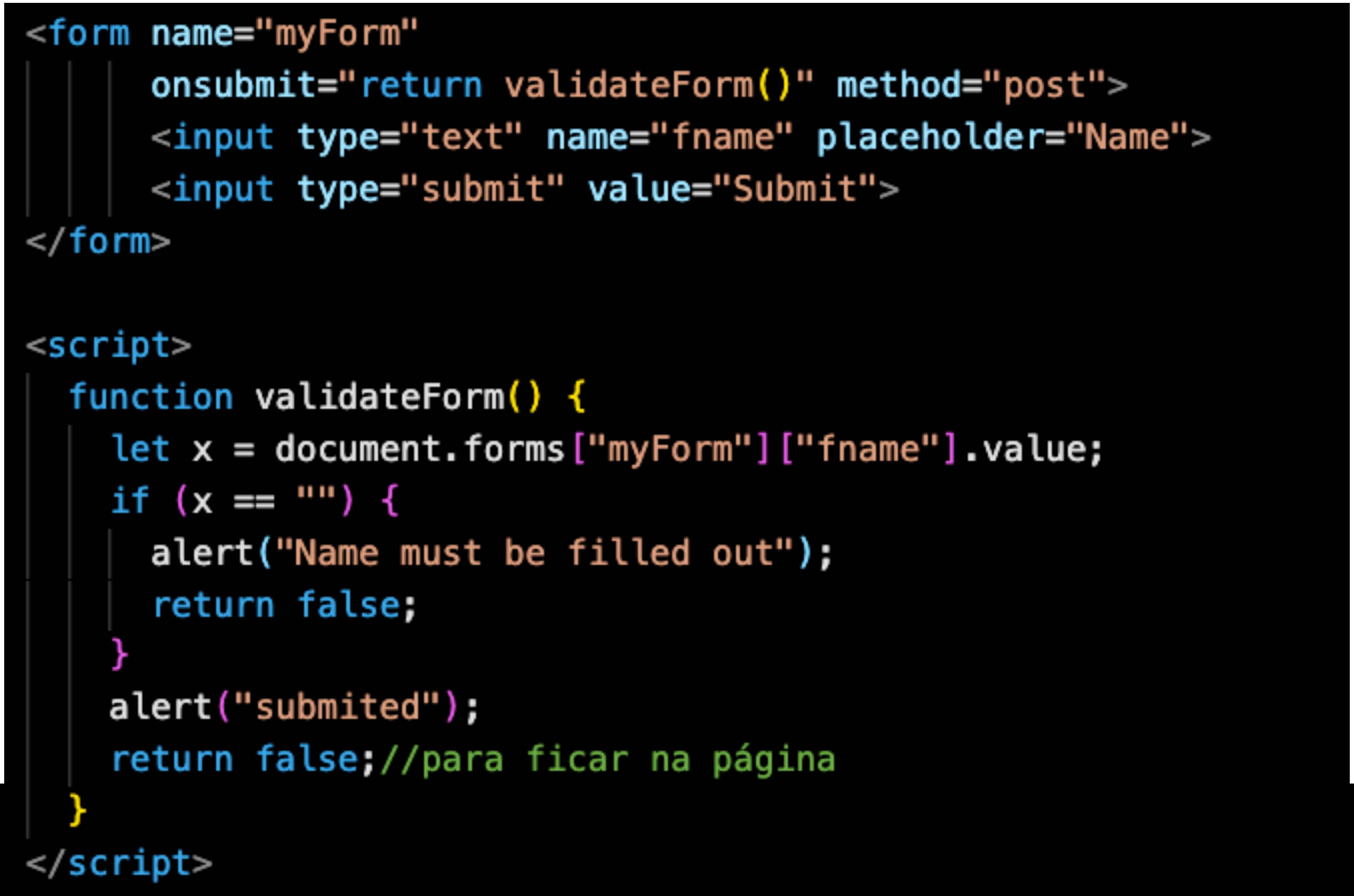
Property	**Description**
element.innerHTML = *new html content*	Change the inner HTML of an element
element.attribute = new value	Change the attribute value of an HTML element
element.style.*property = new style*	Change the style of an HTML element
Method	**Description**
element.setAttribute*(attribute, value)*	Change the attribute value of an HTML element

Referencia DOM - Manipular elementos



Method	**Description**
document.createElement(*element*)	Create an HTML element
document.removeChild(*element*)	Remove an HTML element
document.appendChild(*element*)	Add an HTML element
document.replaceChild(*new, old*)	Replace an HTML element
document.write(*text*)	Write into the HTML output stream

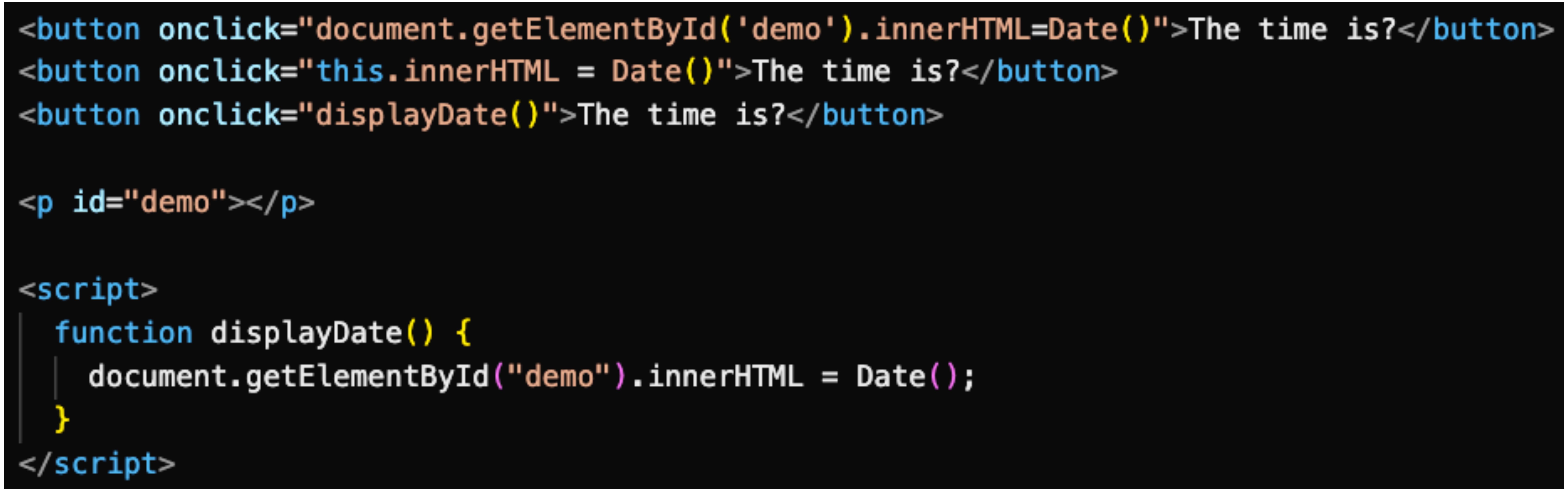
Referencia DOM - Forms



```
<form name="myForm"
      onsubmit="return validateForm()" method="post">
      <input type="text" name="fname" placeholder="Name">
      <input type="submit" value="Submit">
</form>

<script>
  function validateForm() {
    let x = document.forms["myForm"]["fname"].value;
    if (x == "") {
      alert("Name must be filled out");
      return false;
    }
    alert("submited");
    return false;//para ficar na página
  }
</script>
```

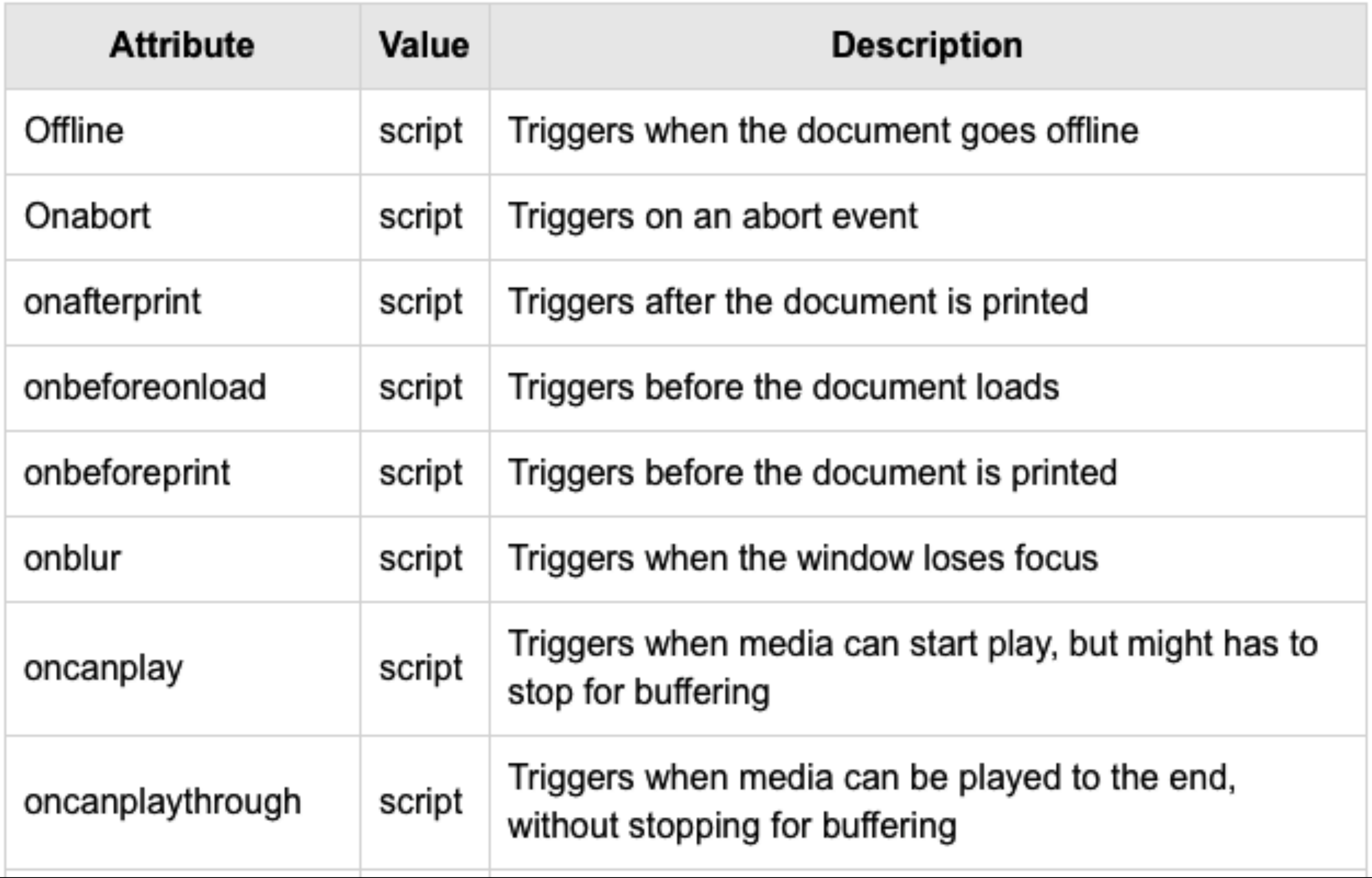
Eventos

- Os eventos são despoletados quando o utilizador manipula a página (e.g. clicar num botão).



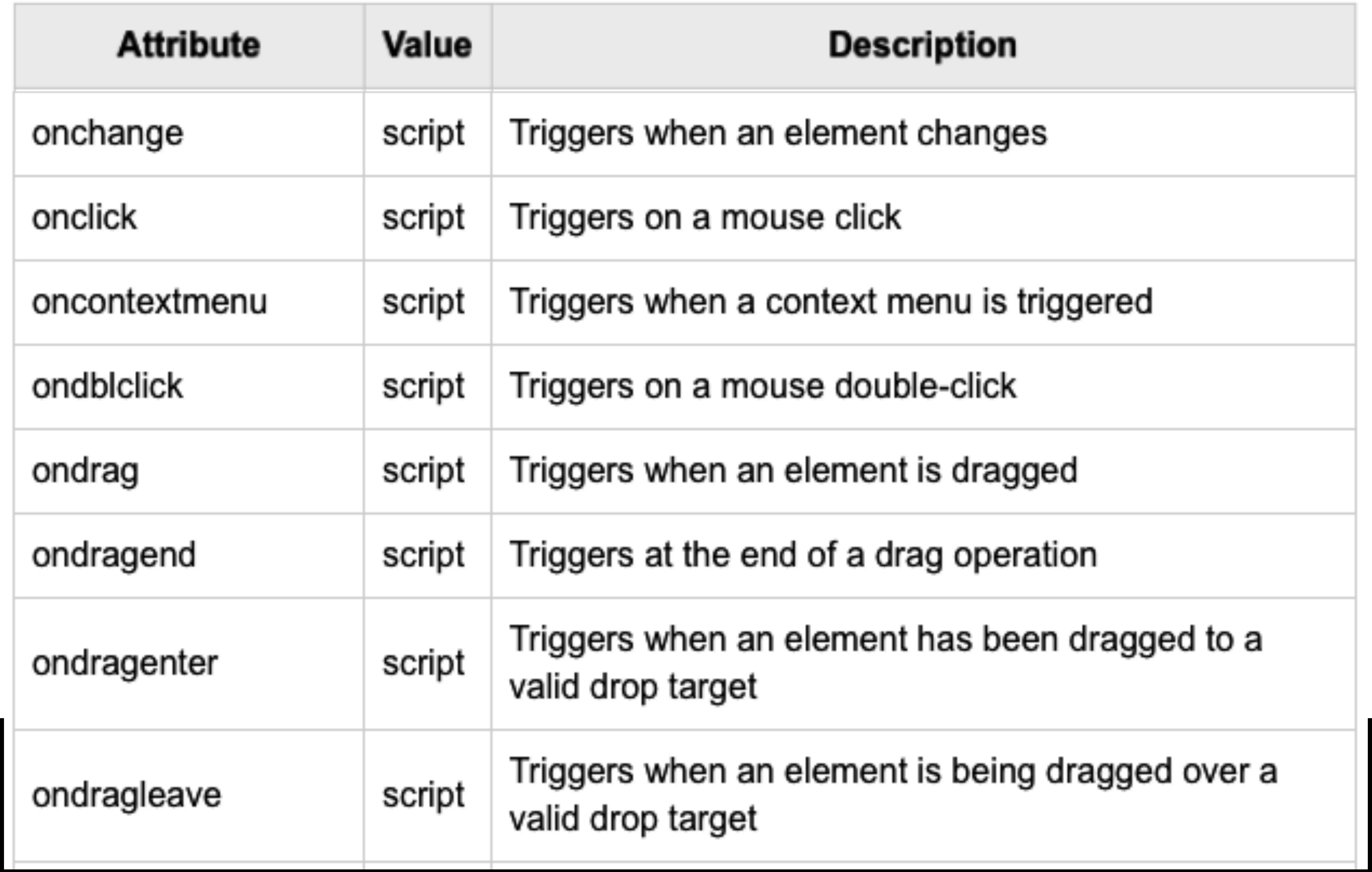
```html
<button onclick="document.getElementById('demo').innerHTML=Date()">The time is?</button>
<button onclick="this.innerHTML = Date()">The time is?</button>
<button onclick="displayDate()">The time is?</button>

<p id="demo"></p>

<script>
  function displayDate() {
    document.getElementById("demo").innerHTML = Date();
  }
</script>
```

Eventos



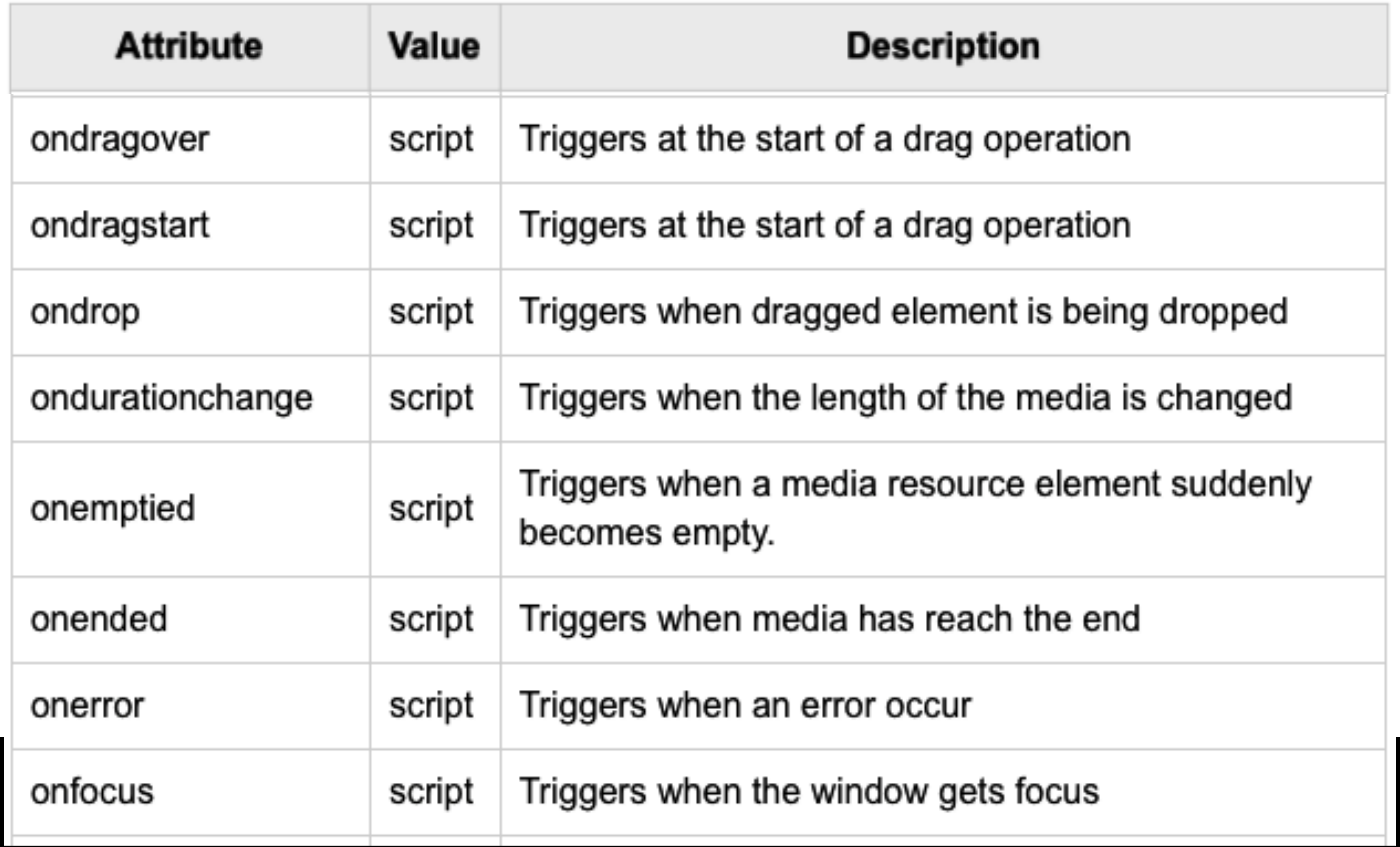
Attribute	Value	Description
Offline	script	Triggers when the document goes offline
Onabort	script	Triggers on an abort event
onafterprint	script	Triggers after the document is printed
onbeforeonload	script	Triggers before the document loads
onbeforeprint	script	Triggers before the document is printed
onblur	script	Triggers when the window loses focus
oncanplay	script	Triggers when media can start play, but might has to stop for buffering
oncanplaythrough	script	Triggers when media can be played to the end, without stopping for buffering

Eventos



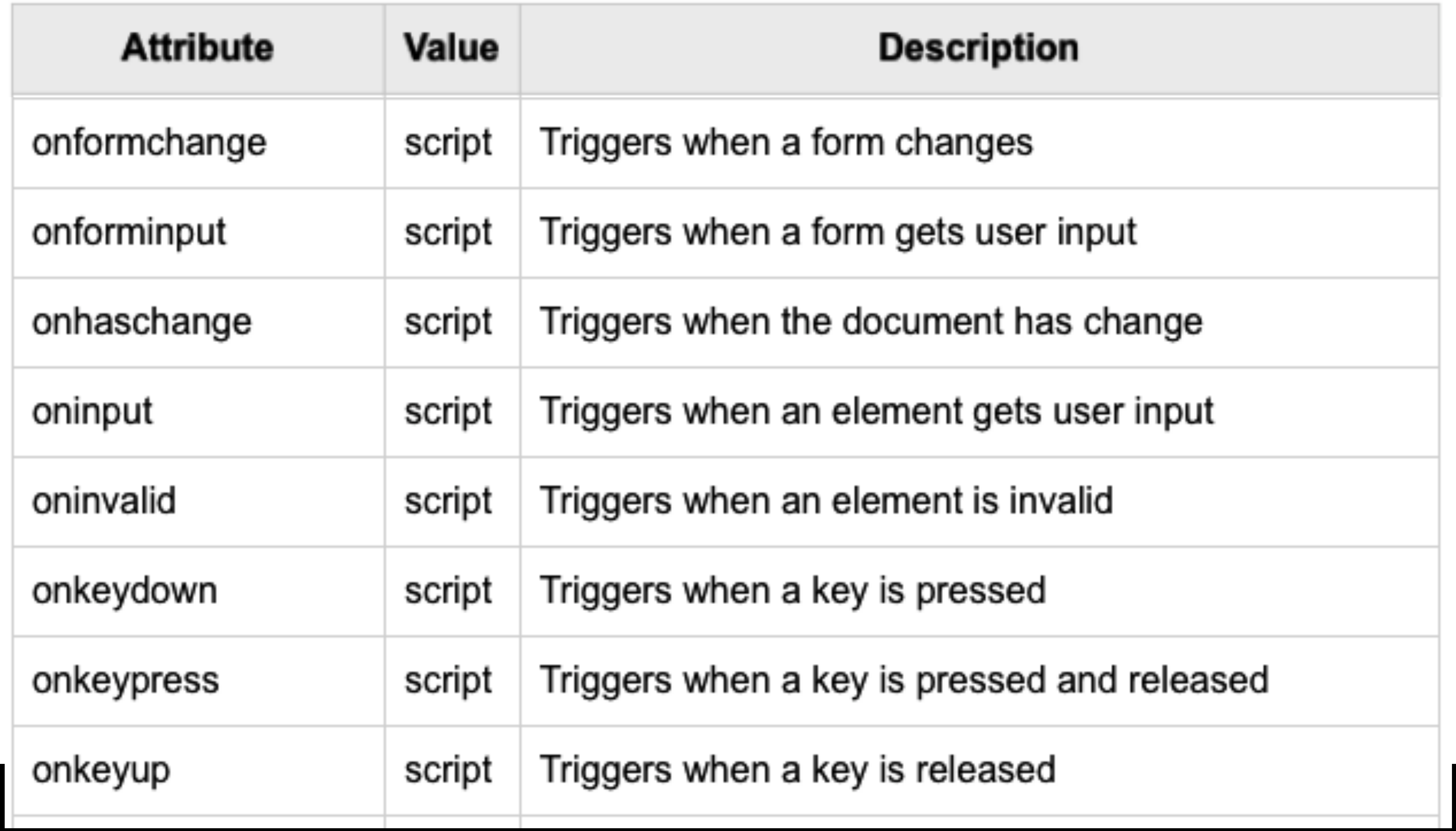
Attribute	Value	Description
onchange	script	Triggers when an element changes
onclick	script	Triggers on a mouse click
oncontextmenu	script	Triggers when a context menu is triggered
ondblclick	script	Triggers on a mouse double-click
ondrag	script	Triggers when an element is dragged
ondragend	script	Triggers at the end of a drag operation
ondragenter	script	Triggers when an element has been dragged to a valid drop target
ondragleave	script	Triggers when an element is being dragged over a valid drop target

Eventos



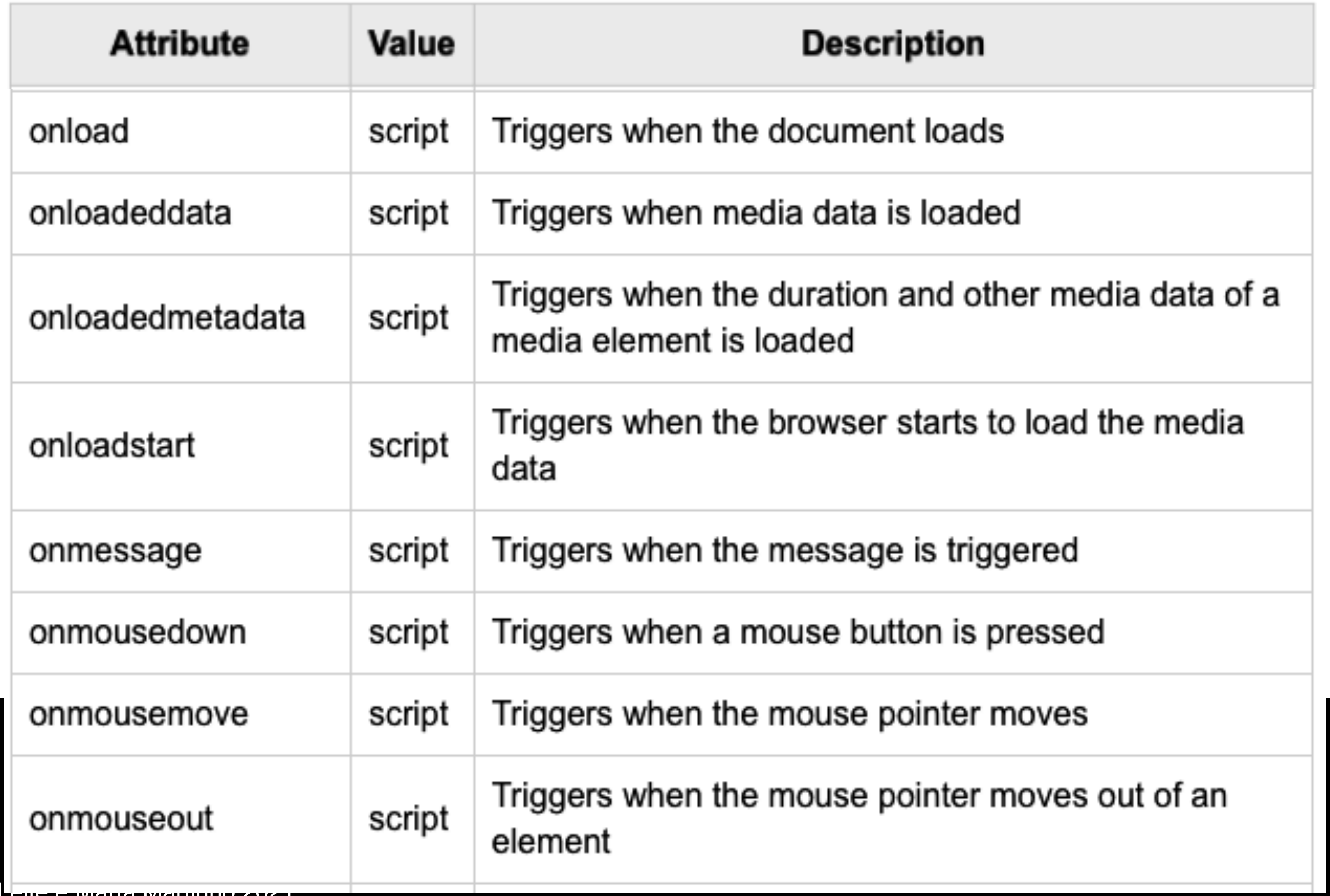
Attribute	Value	Description
ondragover	script	Triggers at the start of a drag operation
ondragstart	script	Triggers at the start of a drag operation
ondrop	script	Triggers when dragged element is being dropped
ondurationchange	script	Triggers when the length of the media is changed
onemptied	script	Triggers when a media resource element suddenly becomes empty.
onended	script	Triggers when media has reach the end
onerror	script	Triggers when an error occur
onfocus	script	Triggers when the window gets focus

Eventos



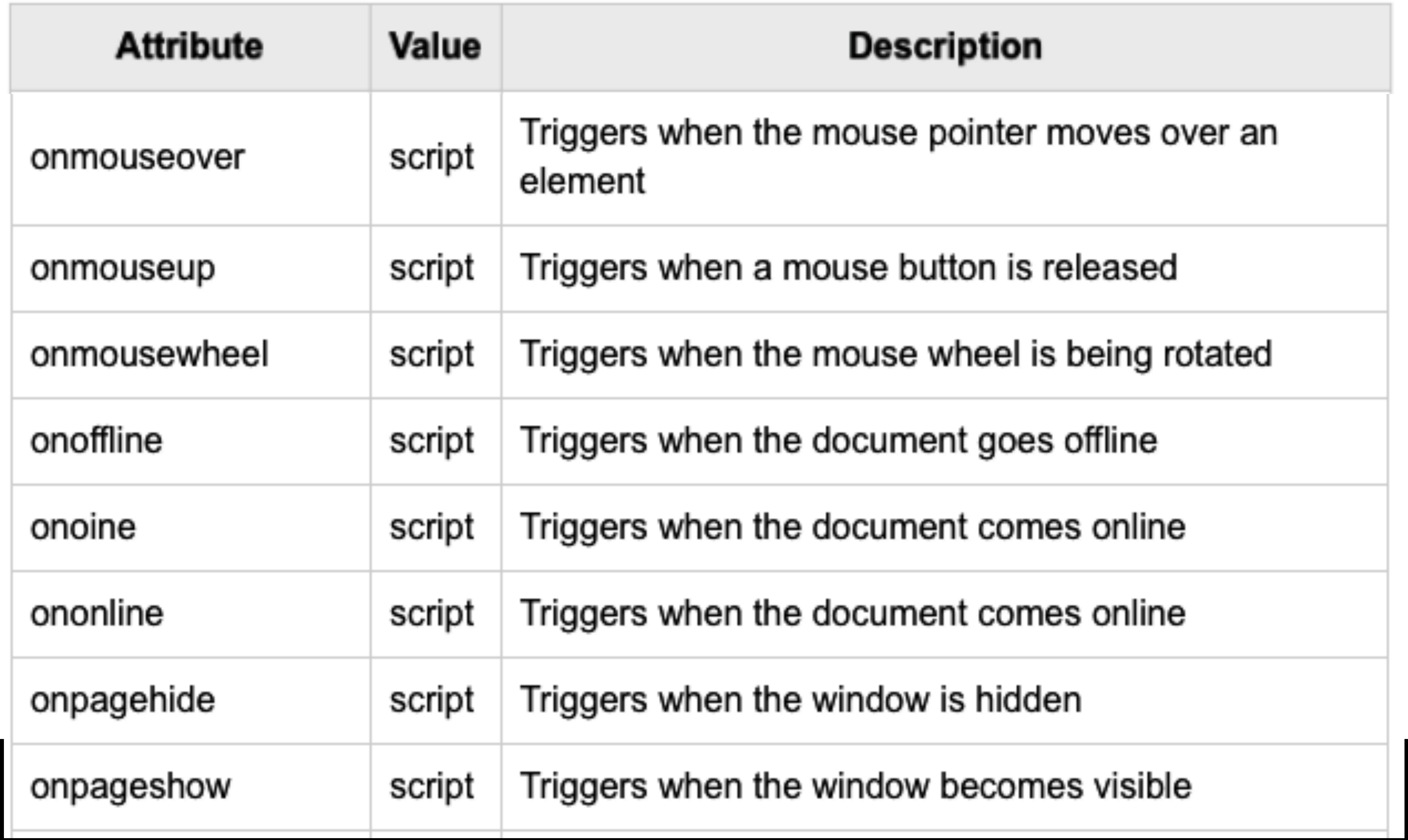
Attribute	Value	Description
onformchange	script	Triggers when a form changes
onforminput	script	Triggers when a form gets user input
onhaschange	script	Triggers when the document has change
oninput	script	Triggers when an element gets user input
oninvalid	script	Triggers when an element is invalid
onkeydown	script	Triggers when a key is pressed
onkeypress	script	Triggers when a key is pressed and released
onkeyup	script	Triggers when a key is released

Eventos



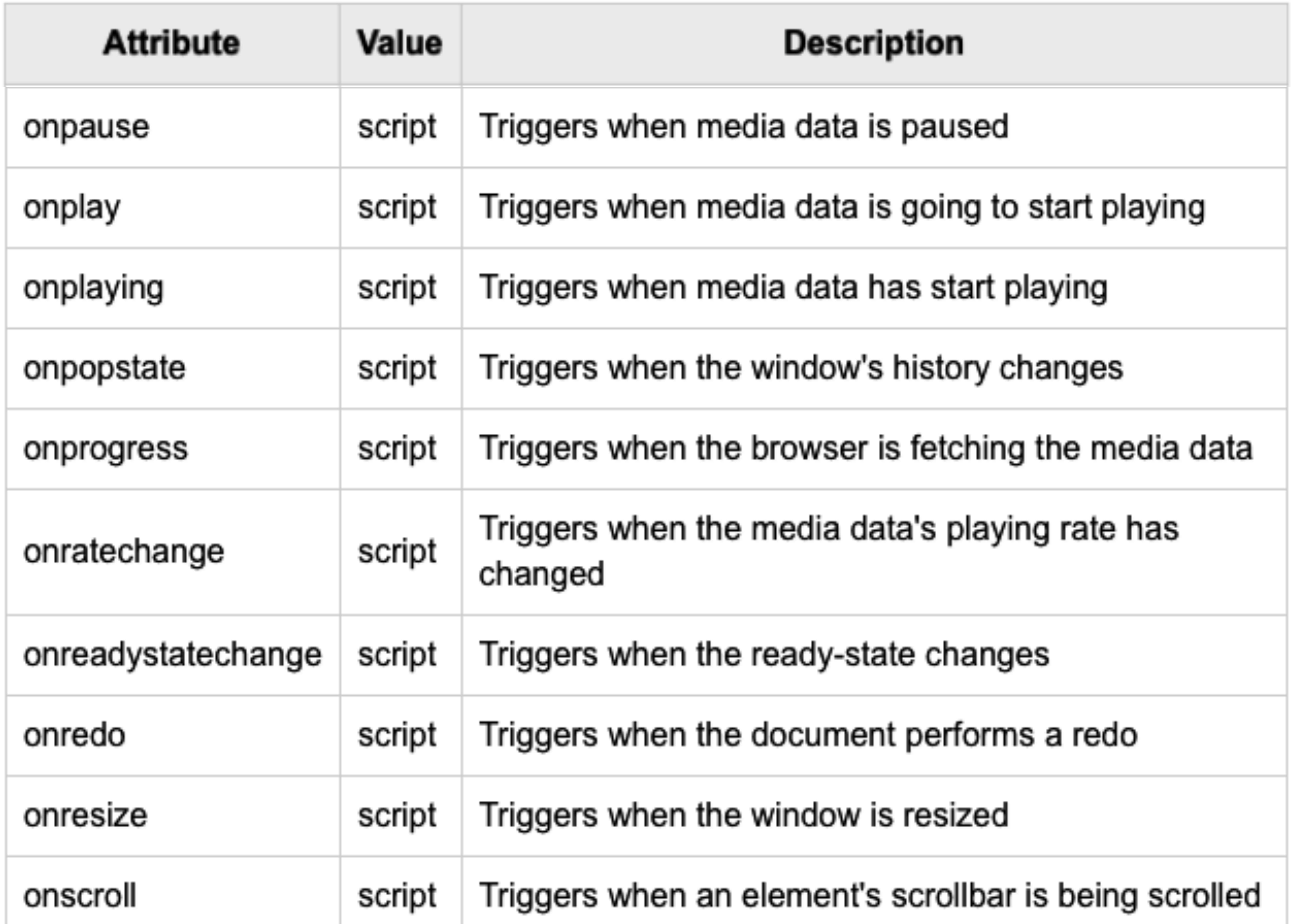
Attribute	Value	Description
onload	script	Triggers when the document loads
onloadeddata	script	Triggers when media data is loaded
onloadedmetadata	script	Triggers when the duration and other media data of a media element is loaded
onloadstart	script	Triggers when the browser starts to load the media data
onmessage	script	Triggers when the message is triggered
onmousedown	script	Triggers when a mouse button is pressed
onmousemove	script	Triggers when the mouse pointer moves
onmouseout	script	Triggers when the mouse pointer moves out of an element

Eventos



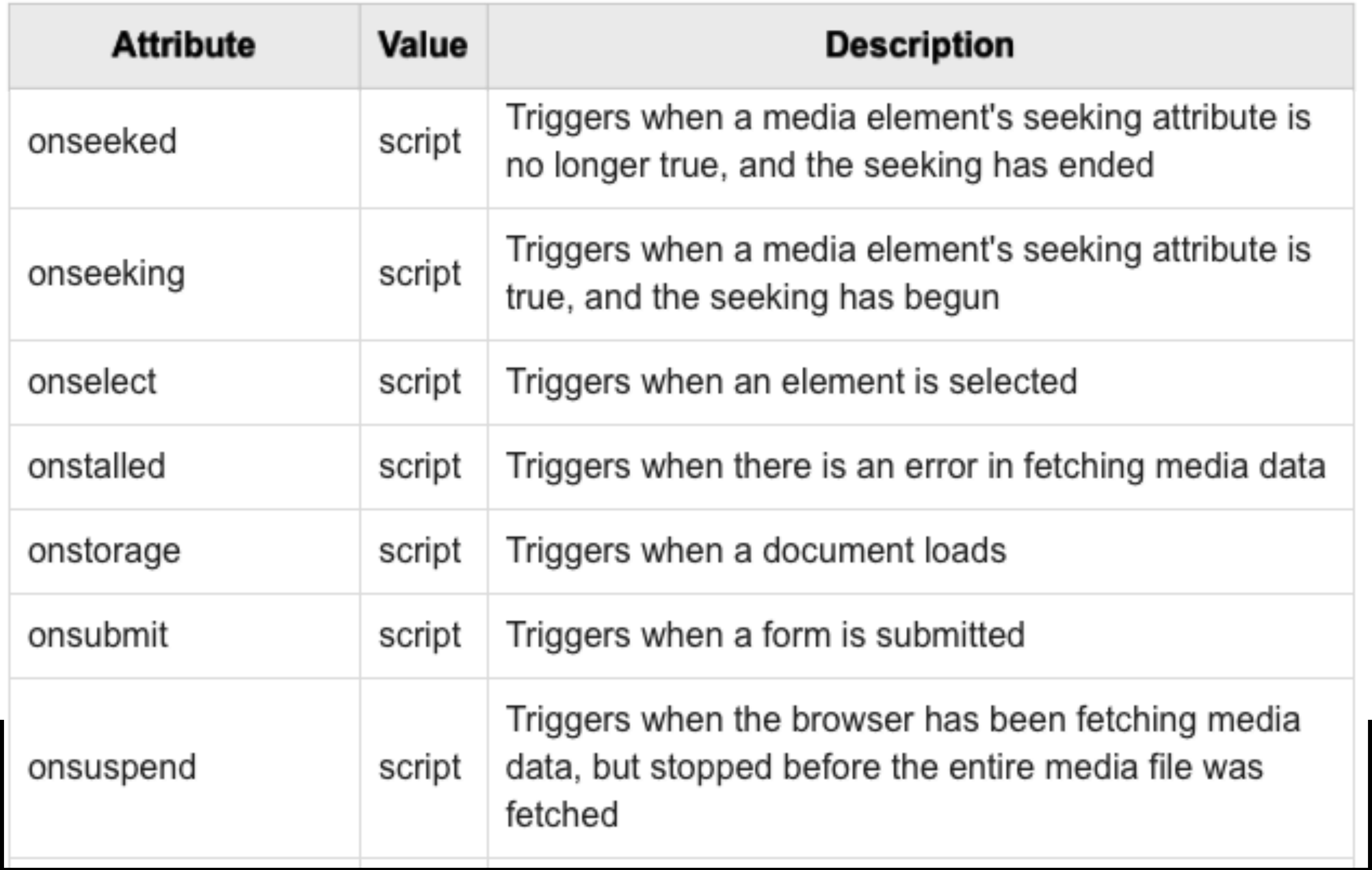
Attribute	Value	Description
onmouseover	script	Triggers when the mouse pointer moves over an element
onmouseup	script	Triggers when a mouse button is released
onmousewheel	script	Triggers when the mouse wheel is being rotated
onoffline	script	Triggers when the document goes offline
onoine	script	Triggers when the document comes online
ononline	script	Triggers when the document comes online
onpagehide	script	Triggers when the window is hidden
onpageshow	script	Triggers when the window becomes visible

Eventos



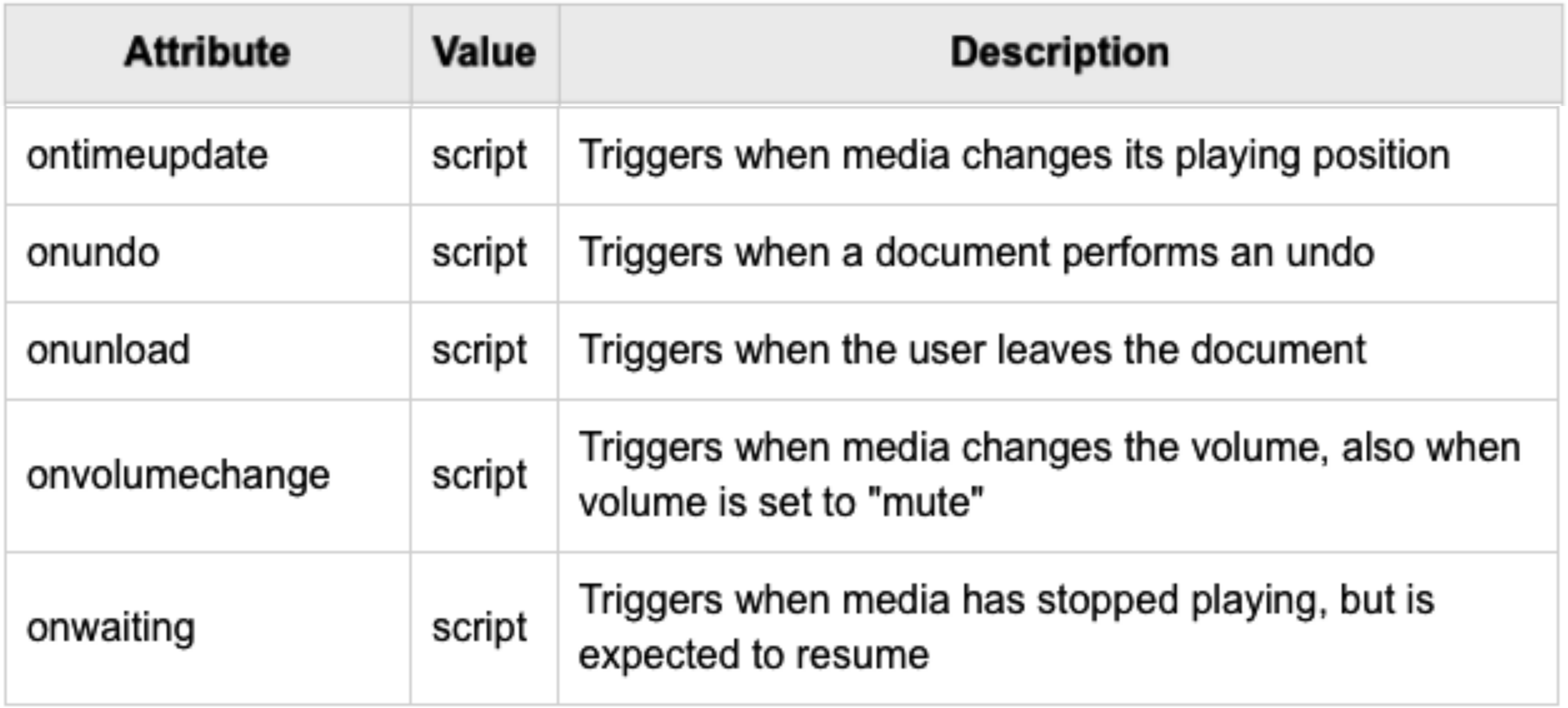
Attribute	Value	Description
onpause	script	Triggers when media data is paused
onplay	script	Triggers when media data is going to start playing
onplaying	script	Triggers when media data has start playing
onpopstate	script	Triggers when the window's history changes
onprogress	script	Triggers when the browser is fetching the media data
onratechange	script	Triggers when the media data's playing rate has changed
onreadystatechange	script	Triggers when the ready-state changes
onredo	script	Triggers when the document performs a redo
onresize	script	Triggers when the window is resized
onscroll	script	Triggers when an element's scrollbar is being scrolled

Eventos



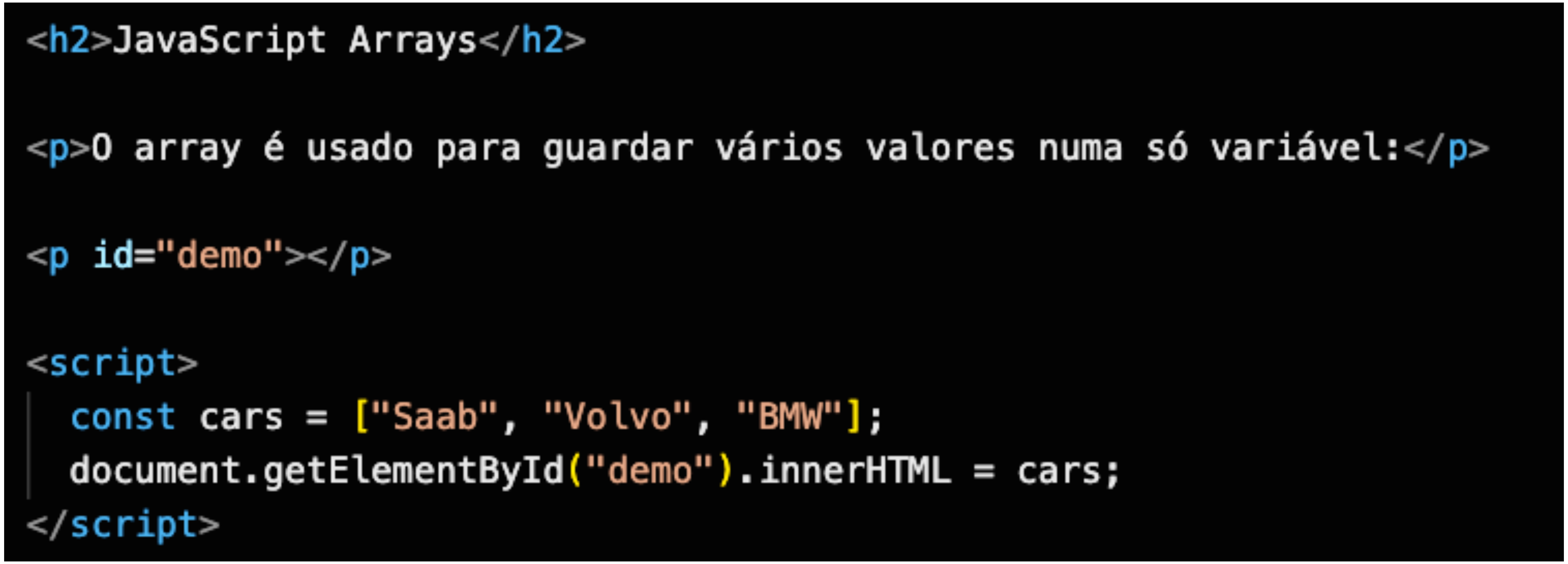
Attribute	Value	Description
onseeked	script	Triggers when a media element's seeking attribute is no longer true, and the seeking has ended
onseeking	script	Triggers when a media element's seeking attribute is true, and the seeking has begun
onselect	script	Triggers when an element is selected
onstalled	script	Triggers when there is an error in fetching media data
onstorage	script	Triggers when a document loads
onsubmit	script	Triggers when a form is submitted
onsuspend	script	Triggers when the browser has been fetching media data, but stopped before the entire media file was fetched

Eventos



Attribute	Value	Description
ontimeupdate	script	Triggers when media changes its playing position
onundo	script	Triggers when a document performs an undo
onunload	script	Triggers when the user leaves the document
onvolumechange	script	Triggers when media changes the volume, also when volume is set to "mute"
onwaiting	script	Triggers when media has stopped playing, but is expected to resume

Arrays



```
<h2>JavaScript Arrays</h2>

<p>O array é usado para guardar vários valores numa só variável:</p>

<p id="demo"></p>

<script>
  const cars = ["Saab", "Volvo", "BMW"];
  document.getElementById("demo").innerHTML = cars;
</script>
```

Arrays - métodos

Operador	Descrição
concat()	Joins two or more arrays, and returns a copy of the joined arrays
copyWithin()	Copies array elements within the array, to and from specified positions
entries()	Returns a key/value pair Array Iteration Object
every()	Checks if every element in an array pass a test
fill()	Fill the elements in an array with a static value
filter()	Creates a new array with every element in an array that pass a test
find()	Returns the value of the first element in an array that pass a test

Arrays - métodos

Operador	Descrição
forEach()	Calls a function for each array element
from()	Creates an array from an object
includes()	Check if an array contains the specified element
indexOf()	Search the array for an element and returns its position
isArray()	Checks whether an object is an array
join()	Joins all elements of an array into a string
keys()	Returns a Array Iteration Object, containing the keys of the original array

Arrays - métodos

Operador	Descrição
lastIndexOf()	Search the array for an element, starting at the end, and returns its position
map()	Creates a new array with the result of calling a function for each array element
pop()	Removes the last element of an array, and returns that element
push()	Adds new elements to the end of an array, and returns the new length
reduce()	Reduce the values of an array to a single value (going left-to-right)
reduceRight()	Reduce the values of an array to a single value (going right-to-left)
lastIndexOf()	Search the array for an element, starting at the end, and returns its position

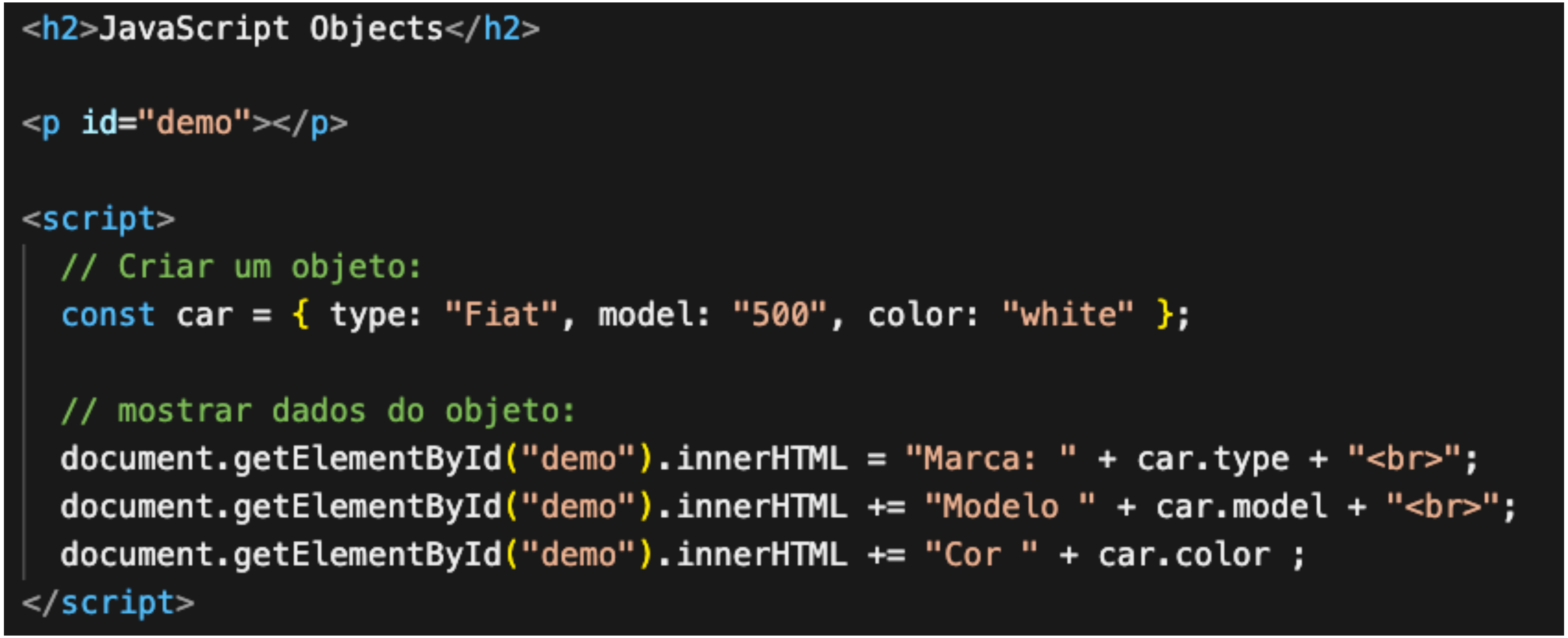
Arrays - métodos

Operador	Descrição
reverse()	Reverses the order of the elements in an array
shift()	Removes the first element of an array, and returns that element
slice()	Selects a part of an array, and returns the new array
some()	Checks if any of the elements in an array pass a test
sort()	Sorts the elements of an array
splice()	Adds/Removes elements from an array
reverse()	Reverses the order of the elements in an array

Arrays - métodos

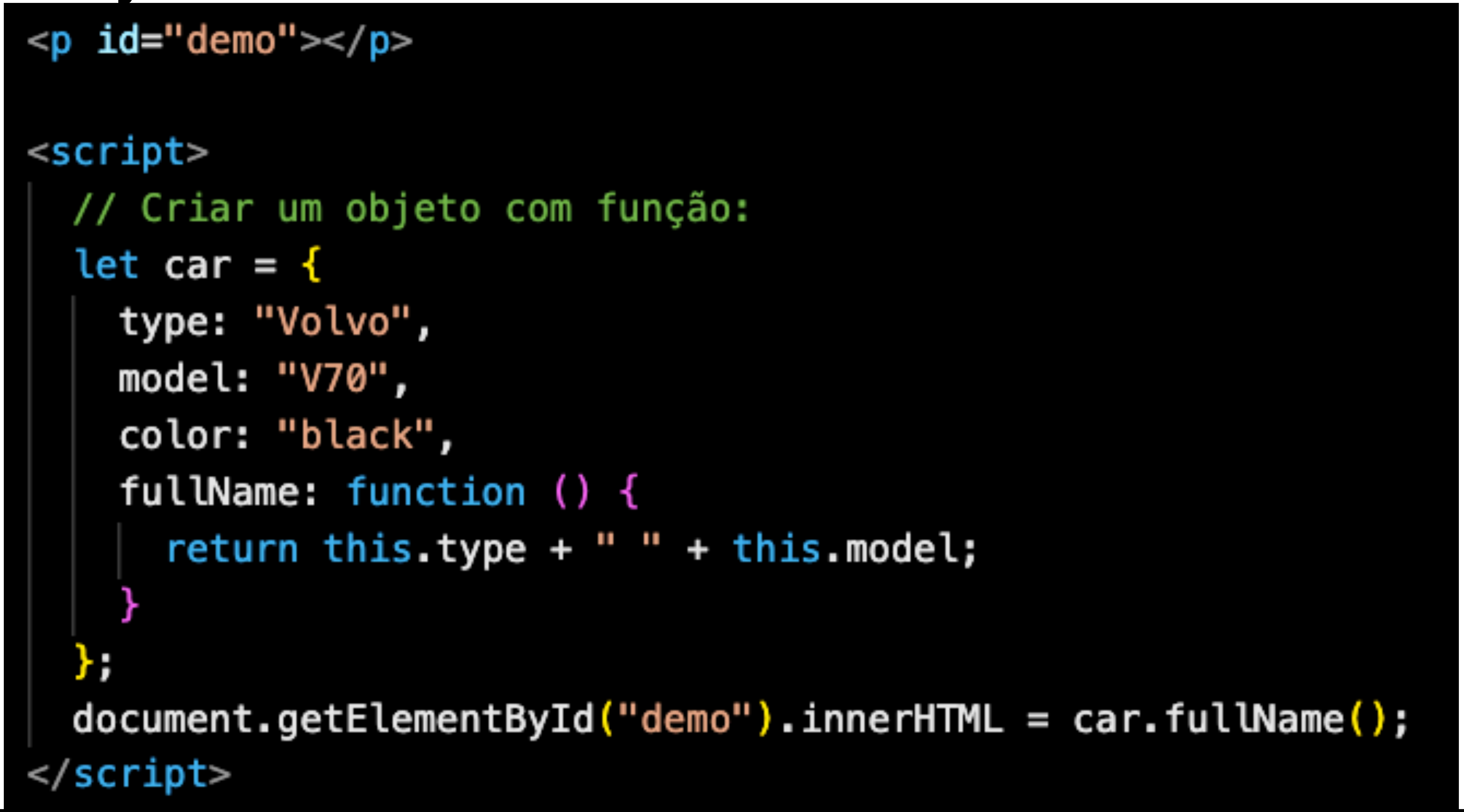
Operador	Descrição
toString()	Converts an array to a string, and returns the result
unshift()	Adds new elements to the beginning of an array, and returns the new length
valueOf()	Returns the primitive value of an array

Objectos



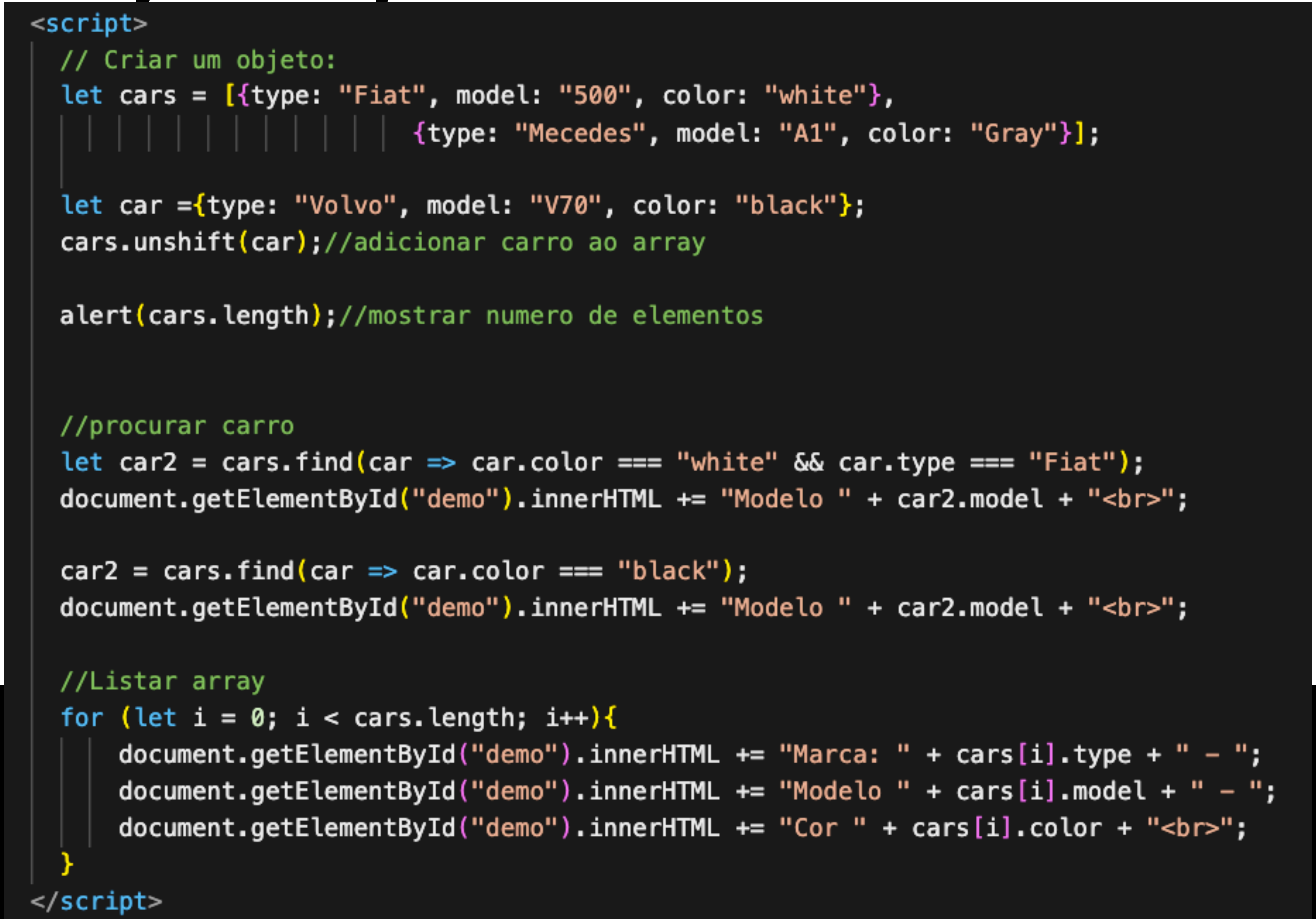
```
<h2>JavaScript Objects</h2>

<p id="demo"></p>

<script>
  // Criar um objeto:
  const car = { type: "Fiat", model: "500", color: "white" };

  // mostrar dados do objeto:
  document.getElementById("demo").innerHTML = "Marca: " + car.type + "<br>";
  document.getElementById("demo").innerHTML += "Modelo " + car.model + "<br>";
  document.getElementById("demo").innerHTML += "Cor " + car.color ;
</script>
```

Objectos



```
<p id="demo"></p>

<script>
  // Criar um objeto com função:
  let car = {
    type: "Volvo",
    model: "V70",
    color: "black",
    fullName: function () {
      return this.type + " " + this.model;
    }
  };
  document.getElementById("demo").innerHTML = car.fullName();
</script>
```

Arrays de objetos



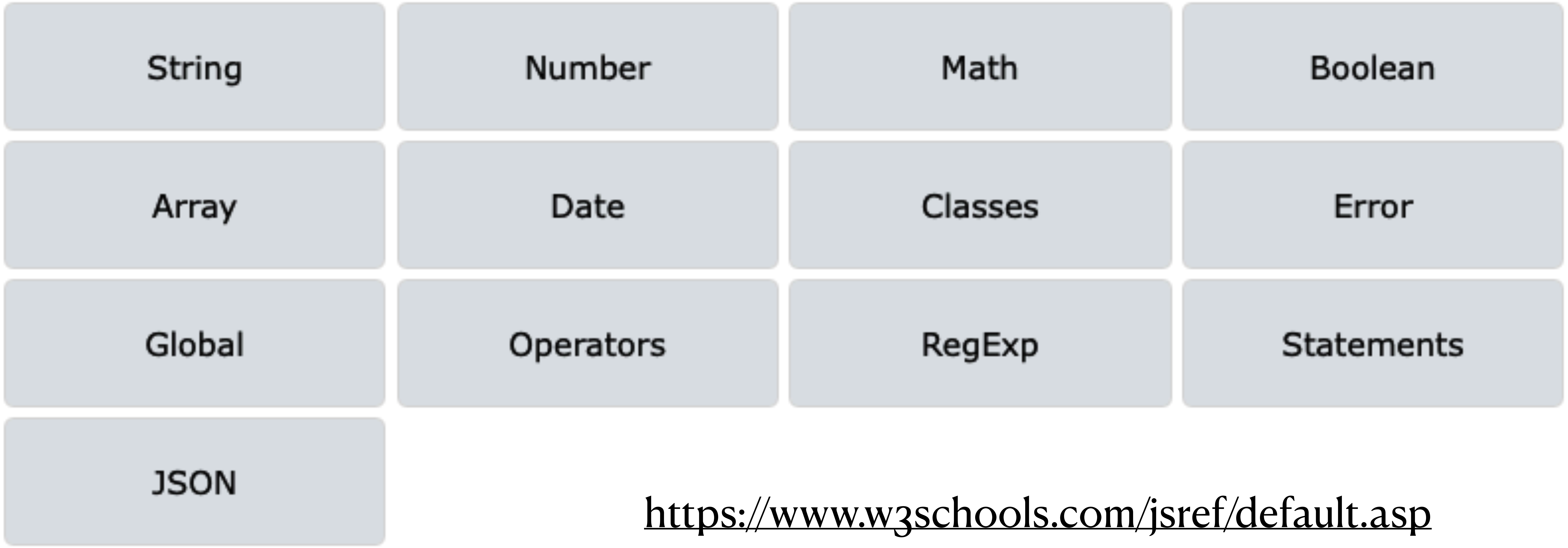
```
<script>
  // Criar um objeto:
  let cars = [{type: "Fiat", model: "500", color: "white"},
                          {type: "Mecedes", model: "A1", color: "Gray"}];

  let car ={type: "Volvo", model: "V70", color: "black"};
  cars.unshift(car);//adicionar carro ao array

  alert(cars.length);//mostrar numero de elementos


  //procurar carro
  let car2 = cars.find(car => car.color === "white" && car.type === "Fiat");
  document.getElementById("demo").innerHTML += "Modelo " + car2.model + "<br>";

  car2 = cars.find(car => car.color === "black");
  document.getElementById("demo").innerHTML += "Modelo " + car2.model + "<br>";

  //Listar array
  for (let i = 0; i < cars.length; i++){
      document.getElementById("demo").innerHTML += "Marca: " + cars[i].type + " - ";
      document.getElementById("demo").innerHTML += "Modelo " + cars[i].model + " - ";
      document.getElementById("demo").innerHTML += "Cor " + cars[i].color + "<br>";
  }
</script>
```

Cookies

- O protocolo HTTP é um protocolo sem estado, i.e. não mantém informação entre páginas.
- O servidor envia dados em forma de cookie para o browser do utilizador.
- A cookie é guardada em formato texto no disco do utilizador
- Quando o utilizador visita outra página, o browser envia a mesma cookie para o servidor para recuperação.

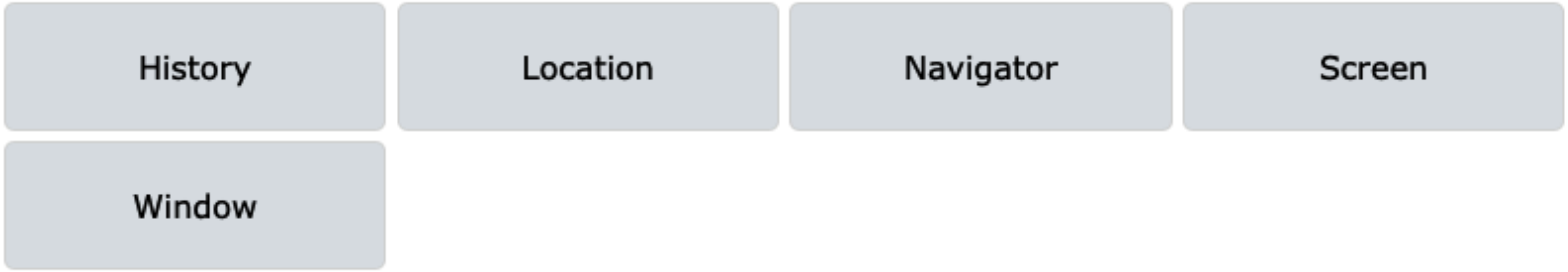
Cookies

Objetos javascript



https://www.w3schools.com/jsref/default.asp

Objetos Window



https://www.w3schools.com/jsref/default.asp

Programação Web

Javascript

Patrícia Leite e Marta Martinho 2021