

WINDOW OBJECT	2
DOM (DOCUMENT OBJECT MODEL)	2
MÉTODOS E PROPRIEDADES DOM	3
LOCALIZANDO ELEMENTOS HTML DOM	3
ALTERANDO ELEMENTOS HTML DOM	6
DOM EVENTS.....	7
DOM EVENTLISTENER	9
SINTAXE	9
THIS KEYWORD.....	10
THIS EM UM MÉTODO	10
THIS SOZINHO	10
THIS EM UMA FUNÇÃO (DEFAULT)	10
THIS EM UMA FUNÇÃO (STRICT)	10
THIS EM EVENT HANDLERS.	10
EMPRÉSTIMO DE FUNÇÃO BIND().....	11
ORDEM DE PRECEDENCIA THIS	11
BIBLIOTECA MATH.....	12
MATH PROPERTIES (CONSTANTS).....	12
MATH METHODS.....	12
NUMEROS INTEIROS	13
JSON.....	14
JSON PARSE	14
JSON STRINGIFY	14
LOCAL STORAGE	15
SINTAXE	15
JAVASCRIPT TIMING EVENTS	17
SETTIMEOUT	17
SETINTERVAL	18
HTML DOM EVENTS REFERENCE.....	19
OBJETOS DE EVENTO HTML DOM	24
PROPRIEDADES E MÉTODOS DO EVENTO HTML DOM	25

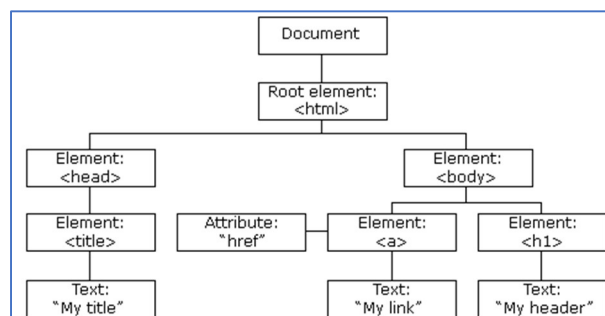
WINDOW OBJECT

O objeto **window** é suportado em todos os tipos de navegadores, ele representa a “janela” do navegador. Em JavaScript todos os global objects (objetos globais), funções e variáveis automaticamente fazem parte do window object. Global variables são propriedades do window object, global functions são métodos do window object.

Até mesmo o objeto document (do HTML DOM) é uma propriedade do objeto window.

DOM (DOCUMENT OBJECT MODEL)

Quando uma página web é carregada, o navegador cria um **Document Object Model** da página. O modelo HTML DOM é construído de maneira similar a uma árvore.



Através desse modelo o JavaScript pode ganhar poder para criar um HTML mais dinâmico:

- JavaScript pode **alterar** todos os **elementos** HTML na página
- JavaScript pode **alterar** todos os **atributos** HTML na página
- JavaScript pode **alterar** todos os **estilos** CSS na página
- JavaScript pode **remover elementos e atributos** HTML existentes
- JavaScript pode **adicionar novos elementos** e atributos HTML
- JavaScript pode **reagir a** todos os **eventos** HTML existentes
- JavaScript pode **criar novos eventos** HTML na página

DOM é um padrão W3C (World Wide Web Consortium), ele define uma maneira de acesso a documentos. *"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."*

MÉTODOS E PROPRIEDADES DOM

Métodos HTML DOM são ações que podem ser executados em elementos HTML. Propriedades HTML DOM são valores de elementos HTML que pode ser estabelecidos ou alterados. A partir desse conceito é possível verificar essa diferença no seguinte exemplo.

O `getElementById` é um método, já que é uma ação que será executada, é a maneira mais comum de acessar um elemento HTML. A `innerHTML` é uma propriedade que, de maneira comum, é usada para acessar o conteúdo de um elemento HTML e alterá-lo.

```
<p id="demo"></p>
<script>
document.getElementById("demo").innerHTML =
"Hello World!";
</script>                                Hello World!
```

Nesse exemplo foi utilizado o método `getElementById` para “buscar” um elemento pela `id` correspondente, e utilizada a propriedade `innerHTML` para alterar o conteúdo desse elemento.

LOCALIZANDO ELEMENTOS HTML DOM

Muitas vezes, com JavaScript, você deseja manipular elementos HTML. Para fazer isso, você tem que encontrar os elementos primeiro. Existem várias maneiras de fazer isso:

- Localizando elementos HTML por `id`
- Localizando elementos HTML por nome de tag
- Localizando elementos HTML por nome de classe
- Localizando elementos HTML por seletores CSS
- Localizando elementos HTML por coleções de objetos HTML.

A maneira mais fácil de localizar um elemento HTML no DOM é usando o `id` do elemento. O seguinte exemplo localiza o elemento com `id="intro"`

```
const element = document.getElementById("intro");
```

Se o elemento não for encontrado, a variável “`element`” vai receber um valor “`null`”.

O seguinte exemplo localiza todos os elementos com a tag <p>.

```
const element = document.getElementsByTagName("p");
```

O seguinte exemplo encontra todos os elementos HTML com o mesmo nome de classe.

```
const x = document.getElementsByClassName("intro");
```

Para encontrar todos os elementos HTML que correspondem a um seletor CSS específico (id, nomes de classes, tipos, atributos, valores de atributos, etc.), use o método **querySelectorAll()**. Este exemplo retorna uma lista de todos os elementos <p> com class="intro".

```
const x = document.querySelectorAll("p.intro");
```

É possível ainda localizar elementos através da object collections. Este exemplo encontra o elemento de formulário com id="frm1", na coleção de formulários e exibe todos os valores do elemento:

```
const x = document.forms["frm1"];
let text = "";
for (let i = 0; i < x.length; i++) {
    text += x.elements[i].value + "<br>";
}
document.getElementById("demo").innerHTML = text;
```

O primeiro HTML DOM Nível 1 (1998), definiu 11 objetos HTML, coleções de objetos e propriedades. Estes ainda são válidos em HTML5. Mais tarde, no HTML DOM Nível 3, mais objetos, coleções e propriedades foram adicionados.

Property	Description	DOM
document.anchors	Retorna todos os elementos do tipo "ancora" <a>	1
document.applets	Descontinuado	1
document.baseURI	Retorna a base URI absoluta do documento	3
document.body	Retorna o elemento <body>	1
document.cookie	Retorna o cookie do documento	1
document.doctype	Retorna o doctype do documento	3
document.documentElement	Retorna o elemento <html>	3
document.documentMode	Retorna o modo usado pelo navegador	3
document.documentURI	Retorna a URI do documento	3

document.domain	Retorna o nome do domínio do server documento	1
document.domConfig	Obsoleto	3
document.embeds	Retorna todos os elementos <embed>	3
document.forms	Retorna todos os elementos <form>	1
document.head	Retorna o elemento <head>	3
document.images	Retorna todos os elementos 	1
document.implementation	Retorna a implementação DOM	3
document.inputEncoding	Retorna a codificação do documento (o conjunto de caracteres)	3
document.lastModified	Retorna a hora e a data de quando o documento foi atualizado	3
document.links	Retorna todos os elementos <area> e <a> que tenham um atributo href	1
document.readyState	Retorna o status (carregamento) do documento	3
document.referrer	Retorna o URI do referenciador (o documento de vinculação)	1
document.scripts	Retorna todos os elementos <script>	3
document.strictErrorChecking	Retorna se a verificação de erros for aplicada	3
document.title	Retorna o elemento <title>	1
document.URL	Retorna a URL completa do documento	1

ALTERANDO ELEMENTOS HTML DOM

A maneira mais fácil de modificar um conteúdo de um elemento HTML é usando a propriedade **innerHTML**.

```
<p id="p1">Hello World!</p>
<script>
document.getElementById("p1").innerHTML = "New text!";
</script>
```

New text!

Para mudar o atributo de um elemento é utilizada a seguinte sintaxe:

```
document.getElementById(id).attribute = new value
```

O seguinte exemplo altera o valor do atributo src de um elemento .

```


<script>
document.getElementById("myImage").src = "landscape.jpg";
</script>
```

É possível ainda alterar o conteúdo diretamente no fluxo de saída do HTML com o método **document.write()**. Porém se ele for utilizado após a página ser carregada o documento pode ser sobrescrito.

PROPRIEDADE	DESCRIÇÃO
<i>element.innerHTML = novo conteúdo HTML</i>	Altera a parte interna do HTML de um elemento
<i>element.attribute = novo valor</i>	Altera o valor do atributo de um elemento HTML
<i>element.style.property = novo estilo</i>	Altera o estilo de um elemento HTML
MÉTODO	DESCRIÇÃO
<i>element.setAttribute(atributo, valor)</i>	Altera o valor de um atributo específico

DOM EVENTS

Um código JavaScript também pode ser executado quando um evento ocorrer, como quando um usuário clica em um elemento HTML ou interage com ele. Para executar um código quando um usuário clica em (**on**) um elemento, é adicionado o seguinte atributo ao elemento:

```
onclick=JavaScript
```

O seguinte exemplo altera um elemento HTML quando o usuário clicar sobre ele, uma função é chamada através de um manipulador de eventos.

```
<h2>JavaScript HTML Events</h2>
<h2 onclick="changeText(this)">Click on this text!</h2>
<script>
function changeText(id) {
  id.innerHTML = "Oops!";
}
</script>
```

Click on this text!

Oops!

É possível atribuir eventos usando o HTML DOM, assim como atribuir uma propriedade ou um método, como no seguinte exemplo.

```
<script>
document.getElementById("myBtn").onclick = displayDate;
</script>
```

Nesse exemplo foi atribuído o evento "onclick" a um elemento botão, a função **displayDate** é atribuída a um elemento HTML com o **id="myBtn"**, a função então vai ser executada quando o botão for clicado.

```
<h2>JavaScript HTML Events</h2>
<p>Click "Try it" to execute the displayDate() function.</p>

<button id="myBtn">Try it</button>
<p id="demo"></p>
<script>
document.getElementById("myBtn").onclick = displayDate;

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

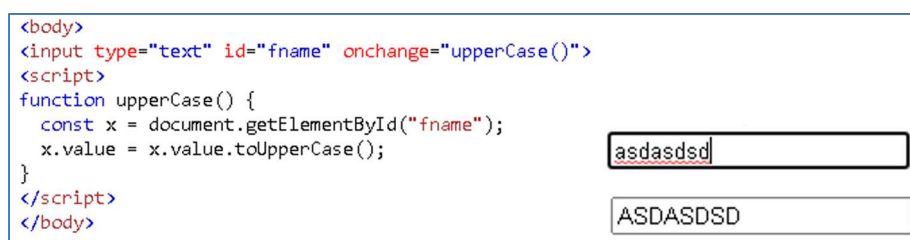
Try it

Sat Jun 11 2022 10:28:41 GMT-0300
(Horário Padrão de Brasília)

Os eventos **onload** e **onunload** são provocados quando um usuário entre ou deixa uma página web, eles podem ser usados para checar se o navegador e a sua versão, e carregar uma página de web adequada com base nessa versão por exemplo. No seguinte exemplo uma caixa de alerta com uma mensagem vai aparecer assim que a página for completamente carregada.



O evento **onchange** é usado muitas vezes para validação de campos de input. Como no seguinte exemplo, o evento vai chamar uma função `toUpperCase()` quando o usuário alterar "changes" o conteúdo do campo de input, essa função vai converter o valor preenchido no campo de input, em letras maiúsculas.



Os eventos **onmousedown**, **onmouseup** e **onclick** utilizam todo os estágios da ação de click do mouse. Primeiro, quando um botão do mouse é clicado, o evento **onmousedown** é provocado, então, quando o botão é liberado, o evento **onmouseup** é provocado, quando essas duas ações forem completadas o evento **onclick** é provocado.

Quando um evento ocorre em HTML, o evento pertence a um determinado objeto de evento, como um evento de clique do mouse pertence ao objeto `MouseEvent`. Todos os objetos de evento são baseados no objeto **Event** e herdam todas as suas propriedades e métodos.

HTML DOM EVENTS:

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

DOM EVENTLISTENER

O método **addEventListener()** anexa um manipulador de eventos a um elemento específico sem substituir manipuladores de eventos existentes. Você pode adicionar muitos manipuladores de eventos a um elemento. Você pode adicionar muitos manipuladores de eventos do mesmo tipo a um elemento, ou seja, dois eventos de "click". Você pode adicionar "event listeners" a qualquer objeto DOM, não apenas a elementos HTML. ou seja, o objeto de janela. É possível remover facilmente um event listeners usando o método **removeEventListener()**.

SINTAXE

O **primeiro** parâmetro é o **tipo do evento** (como "click" ou "mousedown"), o **segundo** parâmetros é a **função** que deve ser chamada quando o evento ocorrer, e o **terceiro** parâmetro é **opcional**, é um valor booleano especificando quando usar **event bubbling** ou **event capturing**.

```
element.addEventListener(event, function, useCapture);
```

O seguinte exemplo exibe uma caixa de alerta na janela do navegador quando o botão com id="myBtn" for clicado (evento "click").

```
<button id="myBtn">Try it</button>
<script>
document.getElementById("myBtn").addEventListener("click",
myFunction);
function myFunction() {
  alert ("Hello World!");
}
</script>
```

Como já foi mencionado, através desse método, se torna possível adicionar muitos manipuladores de eventos, inclusivos os do mesmo tipo em um elemento.

```
<button id="myBtn">Try it</button>
<p id="demo"></p>
<script>
var x = document.getElementById("myBtn");
x.addEventListener("mouseover", myFunction);
x.addEventListener("click", mySecondFunction);
x.addEventListener("mouseout", myThirdFunction);
function myFunction() {
  document.getElementById("demo").innerHTML += "Moused over!<br>";
}

function mySecondFunction() {
  document.getElementById("demo").innerHTML += "Clicked!<br>";
}

function myThirdFunction() {
  document.getElementById("demo").innerHTML += "Moused out!<br>";
}
</script>
```

Try it

Moused over!

Clicked!

Moused out!

THIS KEYWORD

Em JavaScript, a keyword **this** refere-se para a um objeto e pode se referir a diferentes objetos dependendo de como ela for usada.

THIS EM UM MÉTODO

Quando usada em um método de um objeto, this refere-se ao objeto. O seguinte exemplo, “fullName” é um método do objeto com nome “person”, esse método executa uma função que utiliza o this, logo, o this vai referir-se a esse objeto com nome “person”.

```
const person = {
  firstName: "John",
  lastName: "Doe",
  id: 5566,
  fullName : function() {
    return this.firstName + " " + this.lastName;
  }
};
document.getElementById("demo").innerHTML = person.fullName(); John Doe
```

THIS SOZINHO

Quando usado sozinho, this refere-se ao objeto global, ou seja, em um navegador, this refere-se ao objeto window.

```
let x = this;
document.getElementById("demo").innerHTML = x; [object Window]
```

THIS EM UMA FUNÇÃO (DEFAULT)

Em uma função, por padrão o objeto global é vinculado ao this, ou seja, em um navegador, [object Window].

THIS EM UMA FUNÇÃO (STRICT)

O Strict Mode (que será abordado em um tópico mais avançado) não permite “vinculações padrões”, logo, this em uma função em strict mode será undefined.

THIS EM EVENT HANDLERS.

Em HTML event handlers, this vai se referir ao elemento HTML que vai receber aquele evento.

```
<button
onclick="this.style.display=
'none'">Click to Remove Me!
</button>
```



EMPRÉSTIMO DE FUNÇÃO BIND()

No seguinte exemplo, o objeto “member” utiliza o método “fullName”, mas esse é um método do objeto “person”, logo, ele vai se referir ao objeto “person”, com o uso do método bind() um objeto pode pegar emprestado o método de outro objeto.

```
<script>
const person = {
  firstName: "John",
  lastName: "Doe",
  fullName: function() {
    return this.firstName + " " + this.lastName;
  }
}
const member = {
  firstName: "Hege",
  lastName: "Nilsen",
}
let fullName = person.fullName.bind(member);
document.getElementById("demo").innerHTML = fullName(); Hege Nilsen
```

ORDEM DE PRECEDENCIA THIS

Para identificar a qual objeto this está se referindo, é possível utilizar a seguinte ordem de precedência.

PRECEDENCIA	OBJETO
1	bind()
2	apply() and call()
3	Object method
4	Global scope

- This está em uma função sendo chamada usando os métodos bind(), apply() ou call()?
- This está em uma função de um objeto (método)?
- This está em uma função de escopo global?

BIBLIOTECA MATH

O objeto Math em JavaScript permite executar cálculos matemáticos com números. Diferente de outros objetos, Math não tem um “constructor”, é um objeto estático.

MATH PROPERTIES (CONSTANTS)

O JavaScript fornece 8 “constantes matemáticas” que podem ser acessadas com as propriedades Math.

- Math.E // retorna o número de Euler
- Math.PI // retorna PI
- Math.SQRT2 // retorna a raiz quadrada de 2
- Math.SQRT1_2 // retorna a raiz quadrada de 1/2
- Math.LN2 // retorna o logaritmo natural de 2
- Math.LN10 // retorna o logaritmo natural de 10
- Math.LOG2E // retorna o logaritmo de base 2 de E
- Math.LOG10E // retorna o logaritmo de base 10 de E

MATH METHODS

A sintaxe para qualquer método Math é: Math.method(number)

MÉTODO	DESCRIÇÃO
<u>abs(x)</u>	Retorna o valor absoluto de x
<u>acos(x)</u>	Retorna o arco-cosseno de x, em radianos
<u>acosh(x)</u>	Retorna o arco-cosseno hiperbólico de x
<u>asin(x)</u>	Retorna o arco-seno de x, em radianos
<u>asinh(x)</u>	Retorna o arco-seno hiperbólico de x
<u>atan(x)</u>	Retorna o arco-tangente de x como um valor numérico entre -PI/2 e PI/2 radianos
<u>atan2(y, x)</u>	Retorna o arco tangente do quociente de seus argumentos
<u>atanh(x)</u>	Retorna o arcotangente hiperbólico de x
<u>cbrt(x)</u>	Retorna a raiz cúbica de x
<u>ceil(x)</u>	Retorna x, arredondado para cima para o valor inteiro mais próximo
<u>cos(x)</u>	Retorna o cosseno de x (x deve estar em radianos)
<u>cosh(x)</u>	Retorna o cosseno hiperbólico de x
<u>exp(x)</u>	Retorna o valor de E^x

<u>floor(x)</u>	Retorna x, arredondado para baixo para o valor inteiro mais próximo
<u>log(x)</u>	Retorna o logarimo natural (base E) de X
<u>max(x, y, z, ..., n)</u>	Retorna o número de maior valor
<u>min(x, y, z, ..., n)</u>	Retorna o número de menor valor
<u>pow(x, y)</u>	Retorna o valor de x elevado a y
<u>random()</u>	Retorna um número aleatório entre 0 e 1
<u>round(x)</u>	Arredonda x para o valor inteiro mais próximo
<u>sign(x)</u>	Retorna se x é um valor positivo, negativo ou nulo (-1, 0, 1)
<u>sin(x)</u>	Retorna o seno de x (x deve estar em radianos)
<u>sinh(x)</u>	Retorna o seno hiperbólico de x
<u>sqrt(x)</u>	Retorna a raiz quadrada de x
<u>tan(x)</u>	Retorna a tangente de um ângulo
<u>tanh(x)</u>	Retorn a tangente hiperbólica de um número
<u>trunc(x)</u>	Retorna a parte inteira de x

NUMEROS INTEIROS

Existem 4 métodos que são utilizados para arredondar números para um número inteiro:

Math.round(x)	Retorna x arredondado para o inteiro mais próximo
Math.ceil(x)	Retorna x arredondado para cima para o inteiro mais próximo
Math.floor(x)	Retorna x arredondado para baixo para o inteiro mais próximo
Math.trunc(x)	Retorna a parte inteira de x

JSON

JSON significa **JavaScript Object Notation**, é um formato de texto para armazenar e transportar dados, é "auto descritivo" e fácil de entender. A sintaxe JSON é derivada da notação de objeto JavaScript, mas o formato JSON é somente texto. O formato JSON é sintaticamente semelhante ao código para criar objetos JavaScript. Por causa disso, um programa JavaScript pode facilmente converter dados JSON em objetos JavaScript. Como o formato é apenas texto, os dados JSON podem ser facilmente enviados entre computadores e usados por qualquer linguagem de programação.

JSON PARSE

O JavaScript tem uma função integrada para converter strings JSON em objetos JavaScript. Um uso comum do JSON é trocar dados de/para um servidor web. Ao receber dados de um servidor web, os dados são sempre uma string. Analise os dados com `JSON.parse()` e os dados se tornam um objeto JavaScript.

Imagine que recebemos este texto de um servidor web:

```
'{"name":"John", "age":30, "city":"New York"}'
```

Use a função JavaScript **JSON.parse()** para converter texto em um objeto JavaScript:

```
const obj = JSON.parse('{"name":"John", "age":30, "city":"New York"}');
```

JSON STRINGIFY

Ao enviar dados para um servidor web, os dados devem ser uma string. Converta um objeto JavaScript em uma string com `JSON.stringify()`.

Imagine que temos este objeto em JavaScript:

```
const obj = {name: "John", age: 30, city: "New York"};
```

Use a função JavaScript **JSON.stringify()** para convertê-lo em uma string.

```
const myJSON = JSON.stringify(obj);
```

LOCAL STORAGE

A propriedade `localStorage` permite acessar um objeto `Storage` local. A `localStorage` é similar ao `sessionStorage`. A única diferença é que enquanto os dados armazenados no `localStorage` não expiram, os dados no `sessionStorage` tem os seus dados limpos ao expirar a sessão da página — ou seja, quando a página (aba ou janela) é fechada.

SINTAXE

Salvar dados no armazenamento local:

```
localStorage.setItem(key, value);
```

Ler dados do armazenamento local:

```
let lastname = localStorage.getItem(key);
```

Remover dados do armazenamento local:

```
localStorage.removeItem(key);
```

Remover tudo (Limpar armazenamento local):

```
localStorage.clear();
```

O seguinte exemplo conta o número de vezes que um usuário clicou em um botão e mantém o valor mesmo se a página for recarregada ou fechada:

```
<p><button onclick="clickCounter()" type="button">Click me!</button></p>
<p>Number of clicks:</p>
<p id="demo"></p>
<script>
clickCounter();

function clickCounter() {
  if (localStorage.clickcount) {
    localStorage.clickcount = Number(localStorage.clickcount)+1;
  } else {
    localStorage.clickcount = 1;
  }
  document.getElementById("demo").innerHTML = localStorage.clickcount;
}
</script>
```

Click me!

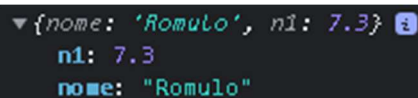
 Number of clicks:
 9

O local storage converte sempre os dados em strings. O seguinte exemplo armazena um objeto no local storage e, através do uso do JSON, converte esse objeto para strings:

```
let a = {nome: "Romulo", n1: 7.3};  
localStorage.setItem("aluno", JSON.stringify(a));
```

Após isso, esse objeto é acessado com o uso do método getItem() da propriedade localStorage, porém, os dados estão em formato de string, então é utilizado o JSON para converter em um objeto novamente (o valor é exibido no console para teste).

```
let b = localStorage.getItem("aluno");  
console.log(JSON.parse(b))
```



```
▼ {nome: 'Romulo', n1: 7.3} ⓘ  
  n1: 7.3  
  nome: "Romulo"
```


JAVASCRIPT TIMING EVENTS

O objeto window permite a execução de códigos de acordo com intervalos de tempo, esses intervalos de tempo são chamados de timing events.

SETTIMEOUT

O método **setTimeout()** executa uma função, após aguardar um número especificado de milissegundos.

```
window.setTimeout(function, milliseconds);
```

Pode ser escrito sem o prefixo window, o primeiro parâmetro é uma função a ser executada, o segundo parâmetro indica o número de milissegundos antes da execução.

O método **clearTimeout()** interrompe a execução da função especificada em setTimeout(), porém, uma variável deve ser declarada e atribuída ao método setTimeout().

```
window.clearTimeout(timeoutVariable)
```

O seguinte exemplo vai exibir uma mensagem na tela após 3 segundos que o usuário interagir com o elemento botão "Try it", porém se o usuário clicar no botão "Stop it" antes dos 3 segundos essa mensagem não será exibida.

```
<button onclick="myVar = setTimeout(myFunction,
3000)">Try it</button>
<button onclick="clearTimeout(myVar)">Stop it</button>
<script>
function myFunction() {
  alert("Hello");
}
</script>
```

SETINTERVAL

O método **setInterval()** faz o mesmo que setTimeout(), mas repete a execução da função continuamente.

```
window.setInterval(function, milliseconds);
```

O primeiro parâmetro é a função a ser executada. O segundo parâmetro indica a duração do intervalo de tempo entre cada execução.

O método **clearInterval()** interrompe as execuções da função especificada no método setInterval(). Esse método usa a variável retornada de setInterval().

O seguinte exemplo exibe as horas atuais, atualizando a contagem dos segundos, porém o botão “stop time” interrompe essa contagem quando for pressionado.

```
<p id="demo"></p>
<button onclick="clearInterval(myVar)">Stop time</button>
<script>
let myVar = setInterval(myTimer ,1000);
function myTimer() {
  const d = new Date();
  document.getElementById("demo").innerHTML =
d.toLocaleTimeString();
}
</script>
```

18:29:55

Stop time

HTML DOM EVENTS REFERENCE

Os eventos HTML DOM permitem que o JavaScript registre diferentes event handlers (manipuladores eventos) em elementos de um documento HTML. Eventos são normalmente usados com uma combinação de funções e essas funções não vão ser executadas antes desses eventos ocorrerem.

EVENTOS	DESCRIÇÃO	PERTENCE A
<u>abort</u>	O evento ocorre quando o carregamento de uma mídia é interrompido	UiEvent, Event
<u>afterprint</u>	O evento ocorre quando uma página está sendo impressa, ou se a caixa de diálogo de impressão for fechada	<u>Event</u>
<u>animationend</u>	O evento ocorre quando uma animação CSS for completada	<u>AnimationEvent</u>
<u>animationiteration</u>	O evento ocorre quando uma animação CSS é repetida	<u>AnimationEvent</u>
<u>animationstart</u>	O evento ocorre quando uma animação CSS é começada	<u>AnimationEvent</u>
<u>beforeprint</u>	O evento ocorre uma página está prestes a ser impressa	<u>Event</u>
<u>beforeunload</u>	O evento ocorre antes do documento ser carregado	UiEvent, Event
<u>blur</u>	O evento ocorre quando um elemento perde o foco	<u>FocusEvent</u>
<u>canplay</u>	O evento ocorre quando o navegador pode começar a reproduzir a mídia (quando tiver buffer suficiente para começar)	<u>Event</u>
<u>canplaythrough</u>	O evento ocorre quando o navegador pode reproduzir a mídia sem parar para armazenamento em buffer	<u>Event</u>
<u>change</u>	O evento ocorre quando o conteúdo de um elemento de formulário, seleção ou verificação de estado for alterado (para <input>, <select> e <textarea>)	<u>Event</u>
<u>click</u>	O evento ocorre quando o usuário clicar em um elemento	<u>MouseEvent</u>
<u>contextmenu</u>	O evento ocorre quando o usuário clica com o botão direito em um elemento para abrir um context menu	<u>MouseEvent</u>
<u>copy</u>	O evento ocorre quando o usuário copia o conteúdo de um elemento	<u>ClipboardEvent</u>
<u>cut</u>	O evento ocorre quando o usuário recorta o conteúdo de um elemento	<u>ClipboardEvent</u>

<u>dblclick</u>	O evento ocorre quando o usuário clica duas vezes em um elemento	<u>MouseEvent</u>
<u>drag</u>	O evento ocorre quando um elemento está sendo arrastado	<u>DragEvent</u>
<u>dragend</u>	O evento ocorre quando o usuário para de arrastar um elemento	<u>DragEvent</u>
<u>dragenter</u>	O evento ocorre quando o elemento arrastado entra em um lugar onde ele pode ser "solto"	<u>DragEvent</u>
<u>dragleave</u>	O evento ocorre quando o elemento deixa o lugar onde ele pode ser solto	<u>DragEvent</u>
<u>dragover</u>	O evento ocorre quando o elemento arrastado está sobre o lugar onde ele pode ser solto	<u>DragEvent</u>
<u>dragstart</u>	O evento ocorre quando o usuário começa a arrastar um elemento	<u>DragEvent</u>
<u>drop</u>	O evento ocorre quando o usuário solta um elemento que estava sendo arrastado	<u>DragEvent</u>
<u>durationchange</u>	O evento ocorre quando a duração da mídia é alterada	<u>Event</u>
<u>ended</u>	O evento ocorre quando a mídia está chegando ao fim (útil para "obrigado por assistir")	<u>Event</u>
<u>error</u>	O evento ocorre quando há um erro durante o carregamento de arquivos externos	ProgressEvent, UiEvent, Event
<u>focus</u>	O evento ocorre quando um elemento está sendo focado	<u>FocusEvent</u>
<u>focusin</u>	O evento ocorre quando um elemento está prestes a ser focado	<u>FocusEvent</u>
<u>focusout</u>	O evento ocorre quando o elemento está prestes a perder o foco	<u>FocusEvent</u>
<u>fullscreenchange</u>	O evento ocorre quando um elemento é exibido em modo de tela-cheia	<u>Event</u>
<u>fullscreenerror</u>	O evento ocorre quando um elemento não pode ser exibido em modo de tela-cheia	<u>Event</u>
<u>hashchange</u>	O evento ocorre quando houver alterações na URL na "âncora"	<u>HashChangeEvent</u>
<u>input</u>	O evento ocorre quando um elemento recebe um input do usuário	InputEvent, Event
<u>invalid</u>	O evento ocorre quando um elemento é inválido	<u>Event</u>
<u>keydown</u>	O evento ocorre quando o usuário aperta uma tecla	<u>KeyboardEvent</u>

<u>keypress</u>	O evento ocorre quando o usuário pressiona uma tecla	<u>KeyboardEvent</u>
<u>keyup</u>	O evento ocorre quando o usuário solta uma tecla	<u>KeyboardEvent</u>
<u>load</u>	O evento ocorre quando um objeto é carregado	UiEvent, Event
<u>loadeddata</u>	O evento ocorre quando os dados de uma mídia forem carregados	<u>Event</u>
<u>loadedmetadata</u>	O evento ocorre quando os metadados são carregados	<u>Event</u>
<u>loadstart</u>	O evento ocorre quando o navegador começar a procurar por uma mídia específica	<u>ProgressEvent</u>
<u>message</u>	O evento ocorre quando uma mensagem for recebida através da fonte do evento	<u>Event</u>
<u>mousedown</u>	O evento ocorre quando o usuário pressiona um botão do mouse sobre um elemento	<u>MouseEvent</u>
<u>mouseenter</u>	O evento ocorre quando o ponteiro do mouse está sobre um elemento	<u>MouseEvent</u>
<u>mouseleave</u>	O evento ocorre quando o ponteiro do mouse deixa um elemento	<u>MouseEvent</u>
<u>mousemove</u>	O evento ocorre quando o ponteiro do mouse está se movendo sobre um elemento	<u>MouseEvent</u>
<u>mouseover</u>	O evento ocorre quando o ponteiro do mouse está sobre um elemento ou sobre seus "filhos"	<u>MouseEvent</u>
<u>mouseout</u>	O evento ocorre quando o usuário move o ponteiro do mouse para fora de um elemento ou de seus "filhos"	<u>MouseEvent</u>
<u>mouseup</u>	O evento ocorre quando o usuário solta o botão do mouse	<u>MouseEvent</u>
<u>mousewheel</u>	<u>Deprecated. Use the wheel event instead</u>	<u>WheelEvent</u>
<u>offline</u>	O evento ocorre quando o navegador começa a trabalhar de maneira offline.	<u>Event</u>
<u>online</u>	O evento ocorre quando o navegador começa a trabalhar de maneira online.	<u>Event</u>
<u>open</u>	O evento ocorre quando uma conexão com um evento de servidor é aberta.	<u>Event</u>
<u>pagehide</u>	O evento ocorre quando o usuário navega para fora de uma página web	<u>PageTransitionEvent</u>
<u>pageshow</u>	O evento ocorre quando o usuário navega para uma página web.	<u>PageTransitionEvent</u>

<u>paste</u>	O evento ocorre quando o usuário cola algum conteúdo em um elemento	<u>ClipboardEvent</u>
<u>pause</u>	O evento ocorre quando uma mídia é pausada	<u>Event</u>
<u>play</u>	O evento ocorre quando uma mídia começa a ser reproduzida	<u>Event</u>
<u>playing</u>	O evento ocorre quando a mídia está sendo reproduzida após ter sido pausada ou interrompida para armazenamento em buffer	<u>Event</u>
<u>popstate</u>	O evento ocorre quando o histórico é alterado	<u>PopStateEvent</u>
<u>progress</u>	O evento ocorre quando o navegador está em processo de obtenção dos dados de mídia (baixando a mídia)	<u>Event</u>
<u>ratechange</u>	O evento ocorre quando a velocidade de reprodução de uma mídia é alterada	<u>Event</u>
<u>resize</u>	O evento ocorre quando a visualização de um documento for redimensionada	UiEvent, Event
<u>reset</u>	O evento ocorre quando um formulário é resetado	<u>Event</u>
<u>scroll</u>	O evento ocorre quando a barra de rolagem de um elemento está sendo rolada	UiEvent, Event
<u>search</u>	O evento ocorre quando o usuário escreve algo em um campo de pesquisa (para <code><input="search"></code>)	<u>Event</u>
<u>seeked</u>	O evento ocorre quando o usuário para de mover ou pular para uma nova posição da mídia.	<u>Event</u>
<u>seeking</u>	O evento ocorre quando o usuário começa a mover ou pular para uma nova posição da mídia.	<u>Event</u>
<u>select</u>	O evento ocorre após o usuário selecionar algum texto (para <code><input></code> e <code><textarea></code>)	UiEvent, Event
<u>show</u>	O evento ocorre quando um elemento <code><menu></code> é exibido como um context menu	<u>Event</u>
<u>stalled</u>	O evento ocorre quando o navegador está tentando receber dados de mídia, mas os dados não são disponíveis	<u>Event</u>
<u>storage</u>	O evento ocorre quando uma área de Web Storage é atualizada	<u>StorageEvent</u>
<u>submit</u>	O evento ocorre quando um formulário é enviado	<u>Event</u>

<u>suspend</u>	O evento ocorre quando o navegador não está recebendo dados de mídia de maneira intencional	<u>Event</u>
<u>timeupdate</u>	O evento ocorre quando a posição de reprodução é alterada.	<u>Event</u>
<u>toggle</u>	O evento ocorre quando o usuário fecha ou abre o elemento <details>	<u>Event</u>
<u>touchcancel</u>	O evento ocorre quando o usuário interrompe o touch	<u>TouchEvent</u>
<u>touchend</u>	O evento ocorre quando o dedo é removido de uma tela touch screen	<u>TouchEvent</u>
<u>touchmove</u>	O evento ocorre quando o dedo é arrastado na tela	<u>TouchEvent</u>
<u>touchstart</u>	O evento ocorre quando um dedo é pressionado em uma tela touch screen	<u>TouchEvent</u>
<u>transitionend</u>	O evento ocorre quando uma transição CSS é completada	<u>TransitionEvent</u>
<u>unload</u>	O evento ocorre quando uma página estiver completamente carregada (para <body>)	UiEvent, Event
<u>volumechange</u>	O evento ocorre quando o volume de uma mídia é alterado	<u>Event</u>
<u>waiting</u>	O evento ocorre quando uma mídia é pausada, mas está esperando para ser reproduzida	<u>Event</u>
<u>wheel</u>	O evento ocorre quando a roda do mouse rola para cima ou para baixo sobre um elemento	<u>WheelEvent</u>

OBJETOS DE EVENTO HTML DOM

Quando um evento ocorre em HTML, o evento pertence a um determinado objeto de evento, como um evento de clique do mouse pertence ao objeto MouseEvent. Todos os objetos de evento são baseados no objeto Event e herdam todas as suas propriedades e métodos

Event Object	Descrição
Event	Parent de todos os objetos de evento

Objeto de Evento	Descrição
AnimationEvent	Para animações CSS
ClipboardEvent	Para modificação da área de transferência
DragEvent	Para interações de arrastar e soltar
FocusEvent	Para eventos de foco
HashChangeEvent	Para alterações na parte âncora do URL
InputEvent	Para input do usuário
KeyboardEvent	Para interação com o teclado
MouseEvent	Para interação com o mouse
PageTransitionEvent	Para navegar e sair de páginas web
PopStateEvent	Para alterações no histórico de entrada
ProgressEvent	Para progresso de carregamento de recursos extenos
StorageEvent	Para alterações na área de armazenamento da janela.
TouchEvent	Para interações touch
TransitionEvent	Para transições CSS
UiEvent	Para interações de interface de usuário UI
WheelEvent	Para interações com a roda do mouse

PROPRIEDADES E MÉTODOS DO EVENTO HTML DOM

Propriedades/Métodos	Descrição	Pertence a
<u>altKey</u>	Retorna se a tecla "ALT" for pressionada quando um evento de mouse for acionado	<u>MouseEvent</u>
<u>altKey</u>	Retorna se a tecla "ALT" for pressionada quando um evento de tecla for acionado	KeyboardEvent, TouchEvent
<u>animationName</u>	Retorna o nome da animação	<u>AnimationEvent</u>
<u>bubbles</u>	Retorna se um evento específico é um "bubbling event"	<u>Event</u>
<u>button</u>	Retorna qual botão do mouse foi pressionado quando o "mouse event" foi acionado	<u>MouseEvent</u>
<u>buttons</u>	Retorna quais botões do mouse foram pressionados quando o "mouse event" foi acionado	<u>MouseEvent</u>
<u>cancelable</u>	Retorna se um evento pode ou não ter sua ação padrão impedida	<u>Event</u>
<u>charCode</u>	Retorna o código de caractere UNICODE da tecla que acionou o evento onkeypress	<u>KeyboardEvent</u>
changeTouches	Retorna uma lista de todos os objetos touch que tiveram seus estados alterados	<u>TouchEvent</u>
<u>clientX</u>	Retorna a coordenada horizontal do ponteiro do mouse relativa a janela atual quando um "mouse event" for acionado	MouseEvent, TouchEvent
<u>clientY</u>	Retorna a coordenada vertical do ponteiro do mouse relativa a janela atual quando um "mouse event" for acionado	MouseEvent, TouchEvent
clipboardData	Retorna um objeto contendo os dados afetados pela execução da área de transferência	<u>ClipboardData</u>
<u>code</u>	Retorna o código da tecla que acionou o evento	<u>KeyboardEvent</u>
composed	Retorna se um evento é composto	<u>Event</u>
<u>ctrlKey</u>	Retorna se a tecla "CTRL" for pressionada quando um evento de mouse for acionado	<u>MouseEvent</u>
<u>ctrlKey</u>	Retorna se a tecla "CTRL" for pressionada quando um evento de tecla for acionado	KeyboardEvent, TouchEvent
<u>currentTarget</u>	Retorna o elemento em que foi acionado um evento pelos "event listeners"	<u>Event</u>
<u>data</u>	Retorna os caracteres inseridos	<u>InputEvent</u>
dataTransfer	Retorna o elemento em que foi acionado um evento pelos "event listeners"	DragEvent, InputEvent

<u>defaultPrevented</u>	Retorna se o método preventDefault() foi ou não chamado para o evento	Event
<u>deltaX</u>	Retorna a quantidade de rolagem horizontal da roda do mouse	WheelEvent
<u>deltaY</u>	Retorna a quantidade de rolagem vertical da roda do mouse	WheelEvent
<u>deltaZ</u>	Retorna a quantidade de rolagem de uma roda do mouse para o eixo z	WheelEvent
<u>deltaMode</u>	Retorna a unidade de medida dos valores delta (pixels, linhas ou páginas)	WheelEvent
<u>detail</u>	Retorna o número que indica quantas vezes o mouse foi clicado	UiEvent
<u>elapsedTime</u>	Retorna o número de segundos de duração de uma animação	AnimationEvent
<u>elapsedTime</u>	Retorna o número de segundos de duração de uma transição	
<u>eventPhase</u>	Retorna qual fase de fluxo de eventos está sendo avaliada no momento	Event
<u>getTargetRanges()</u>	Retorna um array contendo intervalos de destino que podem ser inseridos ou deletados.	InputEvent
<u>getModifierState()</u>	Retorna um array contendo intervalos de destino que podem ser inseridos ou deletados.	MouseEvent
<u>inputType</u>	Retorna o tipo de alteração ("inserir" ou "excluir")	InputEvent
<u>isComposing</u>	Retorna se o estado de um evento é composto ou não	InputEvent, KeyboardEvent
<u>isTrusted</u>	Retorna se um evento é verdadeiro	Event
<u>key</u>	Retorna o valor da tecla representada por um evento	KeyboardEvent
<u>key</u>	Retorna a chave do item de armazenamento alterado	StorageEvent
<u>keyCode</u>	Retorna o código de caractere Unicode da tecla que acionou o evento onkeypress, onkeydown ou onkeyup.	KeyboardEvent
<u>location</u>	Retorna a localização da tecla em um teclado ou dispositivo	KeyboardEvent
<u>lengthComputable</u>	Retorna se a comprimento de uma progressão pode ser computável	ProgressEvent
<u>loaded</u>	Retorna quanto foi carregado	ProgressEvent
<u>metaKey</u>	Retorna se a tecla "META" foi pressionada quando o evento de tecla foi acionado	MouseEvent
<u>metaKey</u>	Retorna se a tecla "meta" foi pressionada quando o evento de tecla foi acionado	KeyboardEvent, TouchEvent

MovementX	Retorna a coordenada horizontal do ponteiro do mouse relativa à posição do último mouse event	MouseEvent
MovementY	Retorna a coordenada vertical do ponteiro do mouse relativa à posição do último mouse event	MouseEvent
newValue	Retorna o novo valor do item de armazenamento alterado	StorageEvent
newURL	Retorna a URL do documento, após o hash ter sido alterado	HasChangeEvent
offsetX	Retorna a coordenada horizontal do ponteiro do mouse em relação à posição da borda do elemento alvo	MouseEvent
offsetY	Retorna a coordenada vertical do ponteiro do mouse em relação à posição da borda do elemento alvo	MouseEvent
oldValue	Retorna o valor antigo do item de armazenamento alterado	StorageEvent
oldURL	Retorna a URL do documento, antes da alteração do hash	HasChangeEvent
onemptied	O evento ocorre quando algo ruim acontece e o arquivo de mídia fica repentinamente indisponível (como desconexões inesperadas)	
pageX	Retorna a coordenada horizontal do ponteiro do mouse, relativa ao documento, quando o evento do mouse foi acionado	MouseEvent
pageY	Retorna a coordenada vertical do ponteiro do mouse, relativa ao documento, quando o evento do mouse foi acionado	MouseEvent
persisted	Retorna se a página da web foi armazenada em cache pelo navegador	PageTransitionEvent
preventDefault()	Cancela o evento se for cancelável, o que significa que a ação padrão que pertence ao evento não ocorrerá	Event
propertyName	Retorna o nome da propriedade CSS associada à animação ou transição	AnimationEvent, TransitionEvent
pseudoElement	Retorna o nome do pseudoelemento da animação ou transição	AnimationEvent, TransitionEvent
region		MouseEvent
relatedTarget	Retorna o elemento relacionado ao elemento que acionou o evento do mouse	MouseEvent
relatedTarget	Retorna o elemento relacionado ao elemento que acionou o evento	FocusEvent
repeat	Retorna se uma tecla está sendo pressionada repetidamente ou não	KeyboardEvent

<u>screenX</u>	Retorna a coordenada horizontal do ponteiro do mouse, relativa à tela, quando um evento foi acionado	MouseEvent
<u>screenY</u>	Retorna a coordenada horizontal do ponteiro do mouse, relativa à tela, quando um evento foi acionado	MouseEvent
<u>shiftKey</u>	Retorna se a tecla "SHIFT" foi pressionada quando um evento foi acionado	MouseEvent
<u>shiftKey</u>	Retorna se a tecla "SHIFT" foi pressionada quando um evento de tecla foi acionado	KeyboardEvent , TouchEvent
<u>state</u>	Retorna um objeto contendo uma cópia das entradas do histórico	PopStateEvent
<u>stopImmediatePropagation()</u>	Impede que outros "event listeners" do mesmo evento sejam chamados	Event
<u>stopPropagation()</u>	Impede a propagação de um evento durante o fluxo de eventos	Event
<u>storageArea</u>	Retorna um objeto que representa o objeto de armazenamento afetado	StorageEvent
<u>target</u>	Retorna o elemento que acionou o evento	Event
<u>targetTouches</u>	Retorna uma lista de todos os objetos de toque que estão em contato com a superfície e onde o evento touchstart ocorreu no mesmo elemento de destino que o elemento de destino atual	TouchEvent
<u>timeStamp</u>	Retorna a hora (em milissegundos em relação à época) em que o evento foi criado	Event
<u>total</u>	Retorna a quantidade total de trabalho que será carregado	ProgressEvent
<u>touches</u>	Retorna uma lista de todos os objetos de toque que estão atualmente em contato com a superfície	TouchEvent
<u>transitionend</u>	O evento ocorre quando uma transição CSS é concluída	TransitionEvent
<u>type</u>	Retorna o nome do evento	Event
<u>url</u>	Retorna a URL do documento do item alterado	StorageEvent
<u>which</u>	Retorna qual botão do mouse foi pressionado quando o evento do mouse foi acionado	MouseEvent
<u>which</u>	Retorna o código de caractere Unicode da tecla que acionou o evento onkeypress, onkeydown ou onkeyup	KeyboardEvent
<u>view</u>	Retorna uma referência ao objeto Window onde ocorreu o evento	UIEvent