

DOM - Document Object Model (Modelo de Objetos para Documentos)

Antes de iniciar vamos criar uma estrutura HTML simples no Visual Studio Code, para que a explicação fique mais fácil em modo visual.

- 1- Crie um arquivo chamado ExercicioX.html
- 2- Inicie um HTML simples
- 3- Coloque o Título como DOM - JavaScript
- 4- De os seguintes estilos ao body: fundo - blue, cor – yellowgreen e fonte normal 18 pontos Arial
- 5- Crie um título – Entendendo o DOM
- 6- Crie um paragrafo – Postando aqui os resultados
- 7- Crie outro paragrafo – Aprendendo a usar o DOM em JavaScript e destaque o DOM
- 8- Crie uma **<div>** – Clique aqui!!!
- 9- Abra área para JavaScript

O que é DOM?

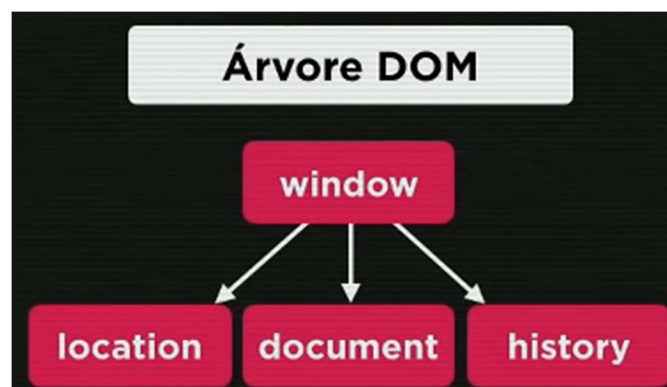
É um acrônimo para **Document Object Model**, que é o modelo de objetos para documentos, é um conjunto de objetos dentro do seu navegador que vai dar acesso aos componentes internos do seu website.

O DOM não funciona dentro do Node.js, ele está presente quando eu estou rodando JavaScript dentro navegador.

É muito importante que você saiba fazer a sua árvore DOM do seu site.

A árvore DOM começa da raiz, que dentro do navegador chamamos de **window**, tudo dentro do JavaScript está dentro de um objeto chamado window, aquela janela do seu navegador é um objeto DOM

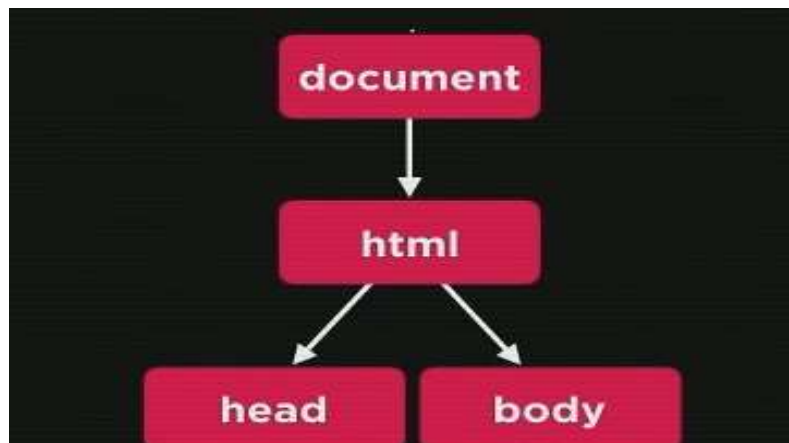
Dentro do **window** existem vários outros objetos, vamos representar apenas três exemplos:



Temos o objeto **location**, que diz qual é a localização do seu site, qual é a URL, qual é a página atual, qual foi a página anterior, o **document**, que é o documento atual e o **history**, que vai guardando de onde você veio, para onde você vai, isso facilita a navegação dentro do seu site.

Mais à frente vou demonstrar tudo o que tem dentro de windows.

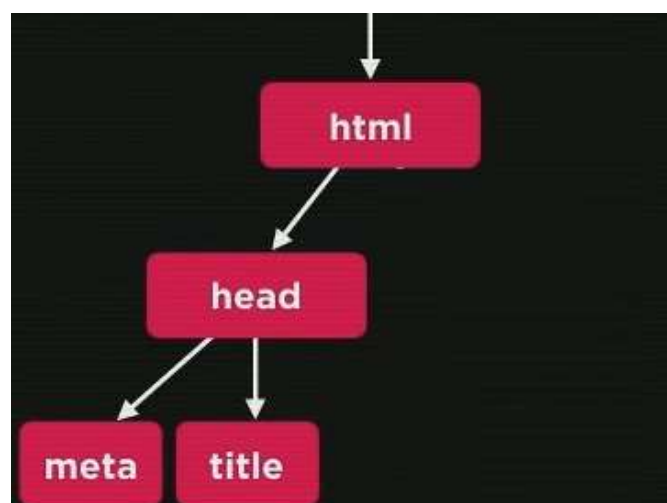
Dentro do **document**, existe um outro objeto muito importante, que é o objeto **HTML**, que é exatamente a parte HTML do site, dentro de HTML eu tenho basicamente dois objetos, o **child**(dois filhos), que são o **head** e o **body**, a parte de cabeçalho e a parte de corpo.



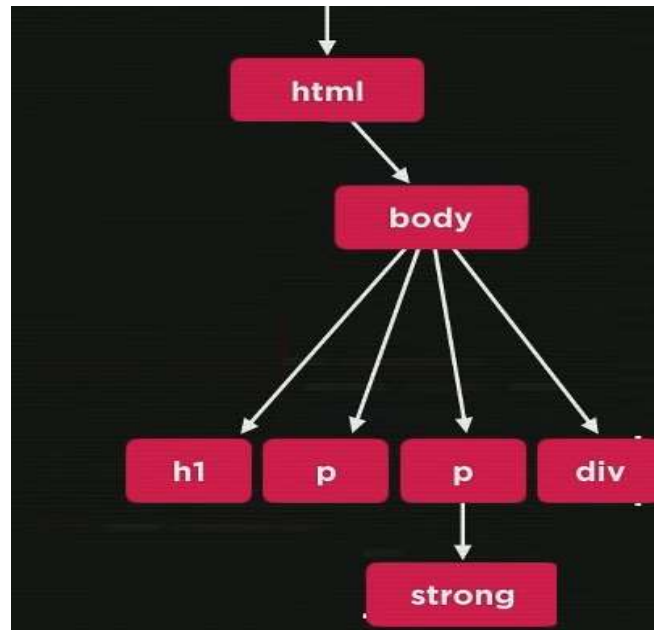
É importante dizer que, **head** e **body**, é **filho de HTML**, são **child**, já por sua vez, o **HTML** é um **parent**, é um **pai/mãe**, de **head** e **body**.

Quem está **embaixo** é **child**, quem está **acima** é **parent**.

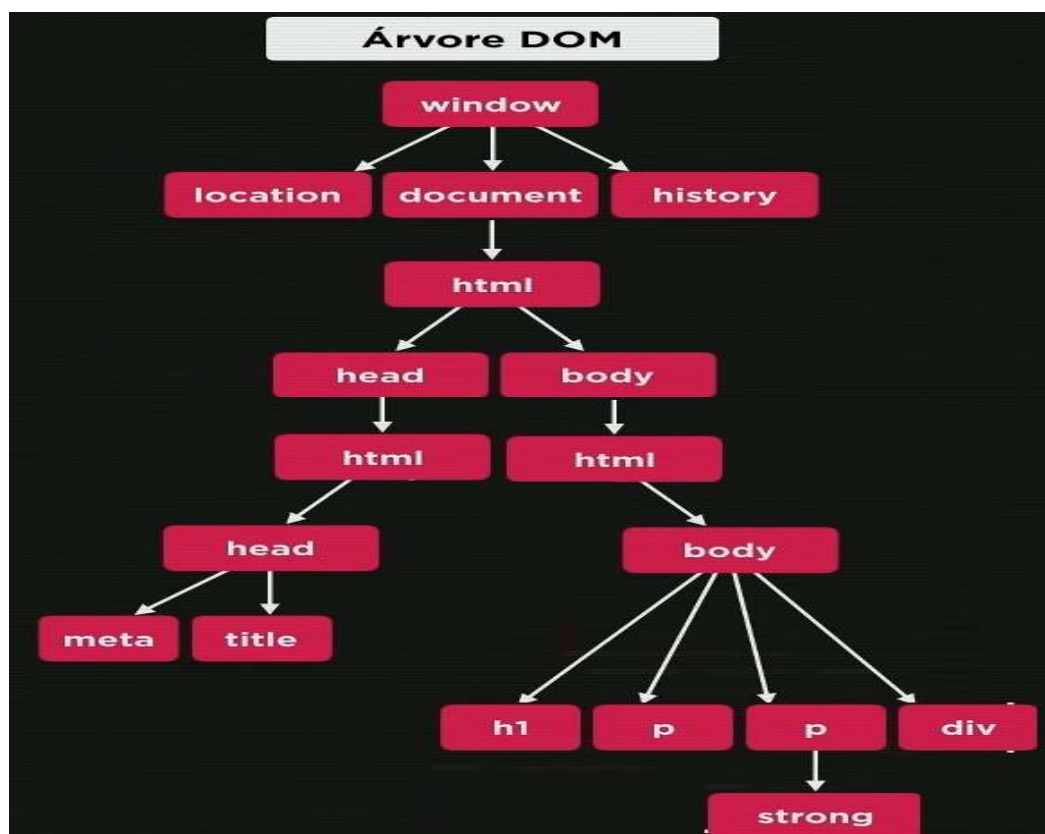
Em nosso documento, dentro de **head**, temos várias **tags**, **meta**, **title** e outras que podemos adicionar em nosso documento.



Já dentro do **body**, para esse documento que estamos em desenvolvimento, temos um **h1**, dois **parágrafos** e uma **<div>** e dentro do segundo parágrafo, um filho que é o **strong**.



Então com isso acabamos de criar a árvore hierárquica do site que acabamos de criar.



É importante que você consiga montar em sua cabeça essa árvore, porque vamos precisar acessar os seus componentes.

Ver exemplos no Visual Studio Code.

Assim conseguimos ter acesso a vários componentes utilizando diretamente DOM, dentro do JavaScript, assim podemos navegar dentro da árvore, da maneira que achar melhor.

E para isso existem várias maneiras de navegar entre os elementos, posso selecionar esses elementos para poder navegar dentro da minha árvore DOM. E existem vários métodos para isso.

Existem 5 métodos de acesso, podemos acessar por:

1. Por Marca (tag);
2. por ID;
3. por Nome;
4. por Classe;
5. por seletor (por CSS, que é um recurso mais recente. Ele não está presente em todas as versões do ECMAScript, mas nas mais recentes, todos os navegadores atualizados já têm acesso a esse outro método aqui que é por seletor).

Em primeiro lugar, vamos aprender a fazer a primeira seleção, como selecionar por marca ou por tag name.

O comando está dentro de document e é o método:

getElementsByTagName()

É importante que você saiba que quando você usa o **getElementsByTagName()**, você consegue selecionar mais de um objeto, porque existem vários objetos do mesmo tipo, da mesma tag.

Por exemplo, o nosso código nós temos dois parágrafos, tenho duas tags <p>, mas tenho só uma tag <body>, só uma tag <div>, só uma tag <h1>.

E como vou selecionar os meus objetos aqui? Veja no Visual Studio Code:

Vamos criar uma variável e essa variável vai se chamar p1, que vai ser meu primeiro parágrafo, e para selecionar os parágrafos, vamos utilizar window.document ou simplesmente só document, ai digitar getE e ele já mostra as opções, seleciono o **getElementsByTagName()**, dentro dos parênteses, entre aspas simples ou duplas, colocar a tag que eu quero selecionar.

```
var p1 = window.document.getElementsByTagName('p')
```

Mas ele não vai pegar um elemento só, note que o comando está no plural **Elements**, e tenho dois parágrafos, para selecionar o primeiro é preciso adicionar depois desse parênteses sem dar espaço, abre e fecha colchete dentro colocar um número, que neste caso será zero, que representa o primeiro parágrafo.

```
var p1 = window.document.getElementsByTagName('p')[0]
```

Se quiser selecionar o segundo, eu vou usar no lugar de zero, vou colocar 1 e assim sucessivamente.

Vamos escrever na tela o utilizando o innerText, ele pega texto que está dentro do primeiro parágrafo.

```
window.document.write('Esse é o texto: ' + p1.innerText)
```

Assim ele replica o mesmo parágrafo original

Outro por exemplo, vamos modificar o estilo do meu p1, vamos modificar para color black.

```
p1.style.color = 'black'
```

Podemos criar uma variável para o corpo do site e acessamos com window.document.body para modificar a cor de fundo a qualquer momento.

```
var corpo = window.document.body  
corpo.style.background = 'green'
```

Assim podemos modificar partes diferentes.

Veja neste outro exemplo:

```
document.write(p1.innerText)
```

Veja, quando digito inner temos innerHTML e innerText. Se escolho innerText ele vai mostrar o resultado, Aprendendo a usar o DOM em JavaScript.

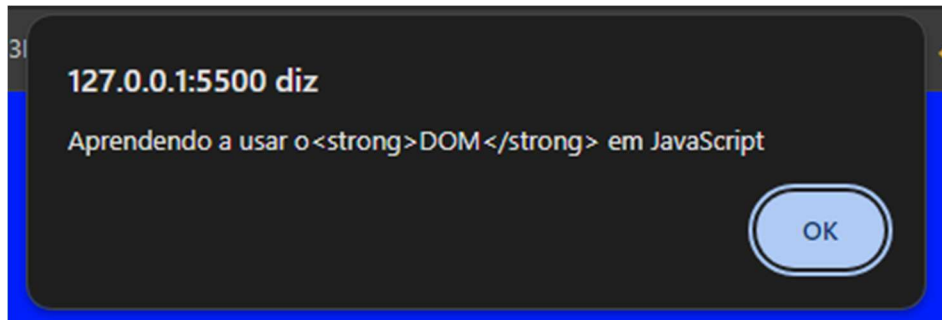
Clique aqui!!!
Aprendendo a usar oDOM em JavaScript

Só que dentro do parágrafo não é só aprendendo a usar o DOM em JavaScript, o **DOM** está em negrito e ele não puxou o negrito, mas se quiser eu posso ver qual é o comando inteiro, em vez de innerText, digitar innerHTML assim ele vem já formatado.

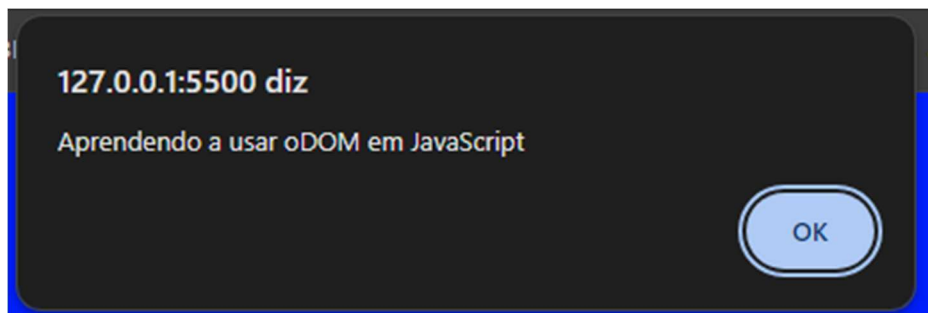
```
document.write(p1.innerHTML)
```

Clique aqui!!!
Aprendendo a usar o**DOM** em JavaScript

Se transformar em um alert, por exemplo, `window.alert`, ele já mostra, só que o DOM veio com as tags.



Em vez de utilizar a `innerHTML`, basta usar o `innerText`.



O `innerText` pega o texto sem as filhas, sem a formatação, ele pega simplesmente o texto o `innerHTML` pega o HTML inteiro, inclusive com as tags filhas.

E assim conseguimos fazer o acesso a todos os componentes utilizando **`getElementsByTagName`** mas essa não é única técnica.

Também podemos selecionar objetos quando o site é um pouco mais evoluído, **por ID**, e não precisamos ficar selecionando elemento por elemento.

Podemos identificar um parágrafo ou a **`<div>`** por um ID e usar o **`getElementById()`**.

Podemos utilizar o **nome** do objeto se tenho uma propriedade **name**, então vamos utilizar o **`getElementsByName()`**, vejam que o **elements** está no plural, então é preciso usar colchete quando temos mais de um objeto.

Podemos usar como conjunto por **classe**, vamos utilizar o **`getElementsByClassName()`**.

Vamos à alguns exemplos:

Vamos pegar a **<div>**, em vez de utilizar o `getElementsByTag`, que vai ser muito genérico, às vezes teremos muitas tags **<div>**, isso vai nos prejudicar, criaremos um id para essa **<div>**, vou chamar de msg. Assim temos um id mensagem.


```
<div id="msg">Clique aqui!!!</div>
```

Vamos selecionar por id, `getElementById`.

```
var d = window.document.getElementById('msg')
```

Vamos fazer o background, fica verde.


```
d.style.background = 'green'
```



Clique aqui!!!

Vamos modificar o conteúdo como `innerText` ou `innerHTML`

```
d.innerText = 'Estou aguardando...'
```




Estou aguardando...

Originalmente a **<div>** está clique aqui!!!, mas madamos modificar por JavaScript usando DOM com o `innerText`, e ele escreveu estou aguardando.

Podemos fazer também, em vez de selecionar por id, selecionar por name. Fazemos a mesma coisa, na **<div>** o nome vai ser msg e como temos no plural elements temos que colocar o primeiro elemento no colchete.

```
<div name="msg">Clique aqui!!!</div>  
var d = window.document.getElementsByName('msg')[0]  
d.innerText = 'Estou aguardando...'
```



Estou aguardando...

Vimos que conseguimos fazer a seleção por **nome**, por **id**, podemos fazer também por **classe**, que basta trocar nome por class, é tudo o mesmo princípio, a forma de acessar é a que você prefere, dependendo da situação pode ser uma, dependendo da situação pode ser outra.

```
<div class="msg">Clique aqui!!!</div>
var d = window.document.getElementsByClassName('msg')[0]
d.innerText = 'Estou aguardando...'
```

Estou aguardando...

E temos uma forma nova de se fazer, que é utilizando por seletor, essa forma nova é até recomendável pela maioria dos manuais, que é utilizando o **querySelector()**, e o **querySelectorAll()** sendo o plural.

Vamos aprender a utilizar o **querySelector** no lugar de utilizar qualquer uma dessas outras formas.

Vamos voltar a **<div>** para id.

```
<div id="msg">Clique aqui!!!</div>
```

E utilizar o **querySelector**.

```
var d = window.document.querySelector('div#msg')
d.style.backgroundColor = "black"
```

Clique aqui!!!

Dentro dos parenteses, vamos usar a síntese do CSS, a **<div>** que tenho id msg.

Toda **<div>** é representada por uma hashtag (#), toda classe é representada por um ponto(.).

```
<div class="msg">Clique aqui!!!</div>
var d = window.document.querySelector('div.msg')
d.style.backgroundColor = "red"
```

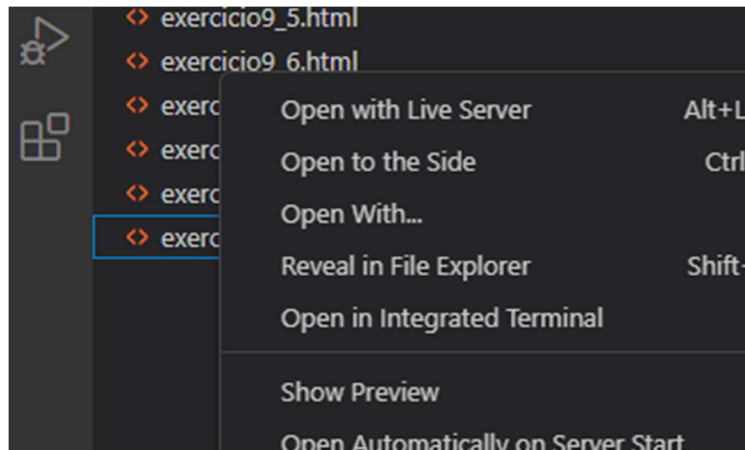
Clique aqui!!!

Lembrando, o **querySelector** é um método mais recente. Então, navegadores mais antigos, que rodam ECMAScript, versões mais antigas, não vão ter suporte a ele.

Vamos aprender agora os **eventos DOM**.

Vamos nas extensões e instalar uma chamada de Live Preview, para que vejamos automaticamente o código assim que digitado sem a necessidade de ficar salvando.

Depois de instalado, basta clicar com o direito do mouse sobre o arquivo que deseja editar e clicar em Show Preview



Agora temos o HTML sendo editado e exibido dentro do Visual Studio Code sem a necessidade de salvar para visualizar.

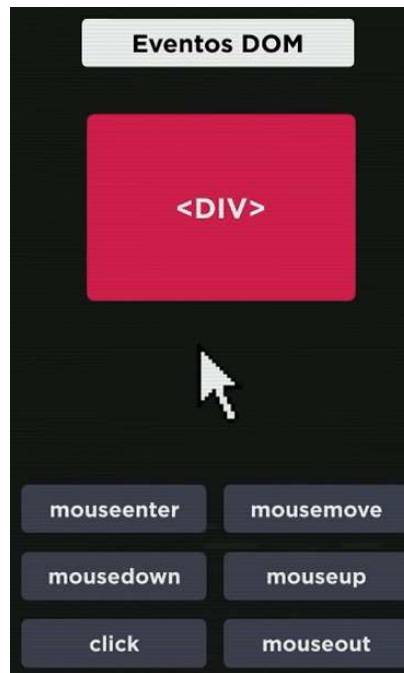
Antes de iniciar vamos criar uma estrutura HTML simples no Visual Studio Code, para que a explicação fique mais fácil em modo visual.

- 1- Crie um arquivo chamado ExercicioX.html
- 2- Inicie um HTML simples
- 3- Coloque o Título como Eventos DOM – JavaScript
- 4- Crie uma **<div>**, de um id área e dentro coloque Interaja...
- 5- De os seguintes estilos a div#area: background verde, a cor da letra branca, largura e a altura em 200 pixels, texto no centro, a linha em 200px com line height e fonte normal 20 pontos Arial
- 6- Abra área para JavaScript

Aproveitando essa **<div>** que acabamos de criar, vamos entender o que são os eventos.

Evento é tudo aquilo que possa acontecer com a **<div>**, ou com qualquer elemento. E várias coisas podem acontecer com ela.

Os eventos de mouse, neste caso, são os mais comuns.



Por exemplo, chegar com o cursor do mouse e mover o mouse até encostar, até chegar dentro da **<div>**.

Quando chega o mouse dentro da **<div>**, dispara um evento chamado **mouseenter**, que o mouse entrou na **<div>**.

Outro evento é continuar movendo o mouse por dentro da **<div>**, nesse momento será disparado várias vezes o método **mousemove**.

Se clicar e segurar, teremos o disparo do evento **mousedown**.

No momento em que eu soltar o botão do mouse eu disparo outro evento, o **mouseup**.

E temos o movimento do clique inteiro, soltar, apertar e soltar rapidamente, temos o evento **click**.

E da mesma maneira que com o **mouseenter**, quando mover o mouse para fora da **<div>**, vai disparar um evento **mouseout**.

Com isso, vimos 6 métodos, 6 eventos que podem ser disparados só com o movimento do mouse.

Existem outros eventos que podem acontecer dentro de elementos em JavaScript.

Acessando em:

<https://html.spec.whatwg.org/multipage/indices.html#events-2>,

teremos uma vasta lista de **Event Reference**, não tem como fazer uma aula falando de todos os eventos, até porque nem todos os elementos são compatíveis com todos os eventos.

Existem eventos específicos para cada situação, agora com o advento dos celulares, temos os **Touch Events**, que são diferentes dos **Mouse Events**, temos uma série de funcionalidades, para quem quiser se especializar, basta acessar esta página.

Essa página do MDN é parte de nossa bibliografia.

Agora sabemos que eventos, são coisas que podemos fazer com esse elemento que criamos, no caso a **<div>**, exemplifiquei somente com os de mouse, mas como demonstrei, existem muitos outros.

Mas para disparar um evento, para tratar um evento, é preciso aprender o que é uma **função** ou uma **funcionalidade**.

Funções

Uma **função** em JavaScript, é um conjunto de códigos, um conjunto de linhas que vão ser executadas só quando o **evento ocorrer**.

Por exemplo, vamos imaginar que vou programar 10 linhas. Essas 10 linhas chamamos de **bloco**. Imagine que esse bloco tivesse 10 linhas, elas não vão ser executadas automaticamente, todos os nossos códigos até o momento foram executados automaticamente, carregava a página e executava o código.

bloco

Agora essas 10 linhas, esse **bloco** só vai ocorrer, no nosso exemplo, quando clicar dentro da **<div>**.

Assim essas 10 linhas vão ser disparadas somente quando o **evento ocorrer**.

Para executar essas 10 linhas somente quando o evento ocorrer, o primeiro passo é colocar elas **dentro de um bloco**.

Um bloco em JavaScript é delimitado entre os sinais de chaves { }.

bloco

Esse bloco é preciso nomear ele como uma **function**, que quer dizer **função**, antes do bloco.

No JavaScript existe essa maneira, que é função anônima, é uma função que simplesmente não tem nome.

```
function {  
    
}
```

Mas para que o método possa funcionar eu tenho que dar um nome para essa função e geralmente os nomes das funções são ações que podemos fazer, geralmente funções de evento, eles são nomes de ação.

Vamos dar o nome da ação que vai acontecer e abre e fecha parênteses.

```
function ação(){  
    
}
```

E opcionalmente, você pode também colocar dentro desses parênteses alguns parâmetros, pode ser um, podem ser vários parâmetros.

```
function ação(param) {  
    
}
```

Vamos ao nosso código, e vamos disparar um código simples, que é um código de clique. Vamos aprender como é feito na parte HTML e também a parte JavaScript.

Os eventos podem ser configurados na parte HTML ou diretamente do JavaScript, serão demonstradas as duas técnicas, você quem escolherá qual você prefere.

O meu método que vamos disparar é o meu evento de click, e para disparar o clique dentro do HTML, vamos colocar dentro da `<div>`, ao lado do `id`, o `onclick` e o nome do método que queremos disparar, o nome do evento `onclick`.

```
<div id="area" onclick="clicar()">
```

Assim o **onclick** vai chamar o nome da ação que queremos adicionar, vamos digitar clicar(), quando clicar em cima da **<div>**, ele vai disparar o clicar

Vamos na parte do JavaScript, e criamos um bloco. Iniciaremos com function {}.

```
function{}
```

Isso é uma função anônima, funciona sem problema no JavaScript, só que temos que colocar um nome para poder encontrar essa função.

```
function clicar(){} 
```

Com isso, se chamar o clicar, ele vai executar tudo que está dentro das chaves{ }.

Essa é a configuração básica, tenho que disparar, configurar o método e esse ele só vai ser disparado quando clicar com o mouse. assim, vamos chamar o método clicar, que chama a função clicar e dentro dela vai ser configurado o que vai acontecer quando for clicado.

Agora vamos fazer o seguinte, quando clicar quero que esse interaja mude o seu conteúdo. Para mudar o conteúdo dele, tenho que ver dentro da minha árvore DOM, e lá temos uma **<div>** de id área.

Podemos fazer assim:

```
var texto = window.document.getElementById('area')
texto.innerText = 'Clicou!'
```

Podemos utilizar também o queryselector, mas vamos iniciar por getElementById, não está no plural então não tem que usar colchetes, utilizamos o innerText, mas podemos fazer com innerHTML, onde podemos colocar tags HTML também.

Analisando antes de colocar para funcionar, quando clicar na minha **<div>**, vou criar um **objeto** chamado **texto** e ele será uma ligação entre o objeto **texto** dentro do JavaScript com esse elemento **area** de **id área** que está dentro do HTML e esse **objeto** vai modificar o conteúdo para **clicou!**.



Se mexer o mouse nada acontece, se clicar ele mostra clicou!.

Vamos disparar outro elemento aqui que é o onmouseenter, quando o mouse entrar, vamos chamar o entrar, vamos ter que criar outra function.

A function entrar.

```
function entrar(){
    texto.innerText = Entrou!
}
```

Vamos colocar esse **var texto** para fora da função, pois assim ele deixara de ser variável local e passara a ser uma variável global, e ai esse **texto** vai funcionar dentro de qualquer outra função.

```
<script>

    var texto = window.document.getElementById('area')

    function clicar(){
        texto.innerText = 'Clicou!'
    }
    function entrar(){
        texto.innerText = 'Entrou!'
    }

</script>
```

Vamos aproveitar e criar o onmouseout e colocar Saiu!.

```
function sair(){
    texto.innerText = 'Saiu!'
}
```

Com isso, disparamos três eventos, e aqui no function, sair, aponto, innerText, saiu.

Então fizemos três métodos, criamos um objeto do lado de fora, para ficar com um escopo global, quando criamos isso do lado de fora, ele serve para todas as três functions.

Agora se passar o mouse ele vai mudar para Entrou!, Saiu!, Entrou!, Saiu!, não estou clicando, cliquei, o clicou! vai aparecer.



Outra coisa que podemos fazer é quando clicar também mudar o style dele, mudar o background para red.

```

<script>

    var texto = window.document.getElementById('area')

    function clicar(){
        texto.innerText = 'Clicou!'
        texto.style.background = 'red'
    }
    function entrar(){
        texto.innerText = 'Entrou!'
    }
    function sair(){
        texto.innerText = 'Saiu!'
        texto.style.background = 'green'
    }

</script>

```

Quando clicar ele vai ficar vermelho, quando ele sair ele vai voltar a ficar verde.



Então a interatividade que temos com a **<div>**, neste caso com a **<div>**, podemos fazer com qualquer elemento, é muito interessante.

E aí vem uma outra dica, além de disparar diretamente pelo HTML, dá para eu disparar os eventos utilizando **listener**, que são ouvidores, eles ficam prestando atenção, dentro do próprio JavaScript.

No lugar de fazer assim:

```

onclick="clicar()" onmouseenter="entrar()" onmouseout="sair()"

```

O que deixar o HTML muito poluído, podemos apagar essa parte do HTML e criar dentro do JavaScript o **addEventListener**, que vai ficar prestando atenção em um determinado evento, e aí vamos criar três listener.

Um para click, que vai ser a execução do clicar, um para o entrar que será **mouseenter**, e por último o **mouseout**, para sair.

Assim, o HTML fica limo e o JavaScript é que vai tratar de fazer a ligação. entre o HTML e o JavaScript, vai ser feito por essas três linhas que é o event listener.

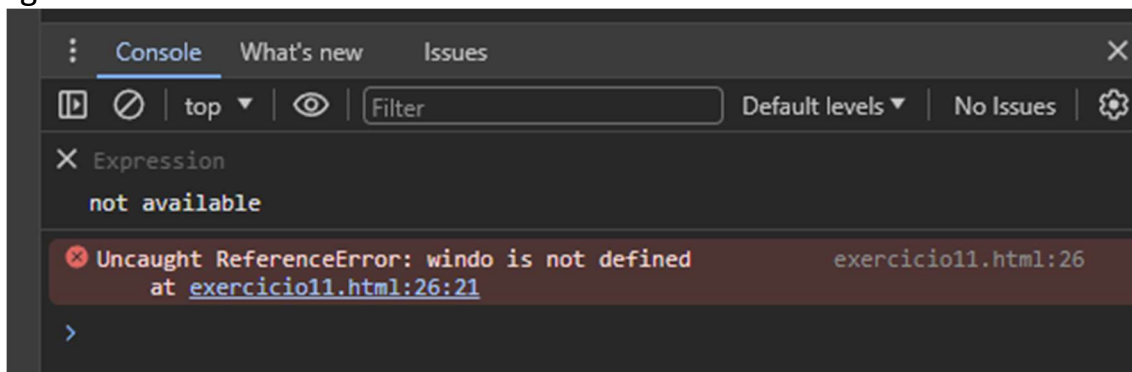
```
texto.addEventListener('click', clicar)
texto.addEventListener('mouseenter', entrar)
texto.addEventListener('mouseout', sair)
```

E agora temos uma coisa que é importante, a detecção de erros.

Só que ele não dá o erro, ele não fala assim, olha teve um erro, o erro é esse. Como descobrimos que está dando erro, o que aconteceu, como você vai descobrir que problema aconteceu?

Na maioria dos casos é só você clicar com o botão direito, vira em inspecionar, aí ele vai abrir um inspetor aqui, ele vai abrir o DevTools. Ao olhar embaixo no console veremos, a mensagem eu não consegui ler a propriedade GetElementaryD na linha X, o erro está da linha X para cima, basta corrigir que já volta a funcionar.

Por exemplo, vou esquecer o **w**, vai ficar o **Windo**, ele não está funcionando no interaja, ele deu o erro aqui, Windo is not defined, ele não reconheceu a palavra Windo, você consegue detectar erros diretamente usando o DevTools do próprio Google Chrome.



Basta corrigir, **window**, salvar e agora já voltou a funcionar, é comum e normal termos pequenos erros, não tem problema nenhum com isso, com essa dica, conseguimos resolver.

Vamos criar outro exemplo utilizando eventos DOM, só que agora vamos começar a interagir mais com o usuário, buscar valores do usuário, começar a deixar ele digitar os dados.

No Visual Studio Code e vamos criar mais um arquivo, exercicio12.html.

Vamos criar um HTML básico:

1. Colocar no título do site, somando números;
2. No body criar um Título Somando Valores;
3. Criar input para do tipo number com atributo name e o id = txtn1, colocar um espaço e um sinal de + no final;
4. Criar outro input do tipo number com atributo name e id = txtn2 e o id txtn2.

5. criar outro input do tipo button com valor somar e nesse botão colocar o onclick = somar(), que será o método, a função.

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Somando Numeros</title>
</head>
<body>
  <h1>Somando Valores</h1>
  <input type="number" name="txtn1" id="txtn1"> +
  <input type="number" name="txtn2" id="txtn2">
  <input type="button" value="Somar" onclick="somar()">
</body>
</html>
```

Vamos ao script que terá uma função somar. Nesta função somar, primeiro vamos ter que saber quem é N1 e N2.

Mas antes vamos criar dois objetos, o tn1, que é a caixa de texto n1 e o tn2, que é a caixa de texto n2.

```
function somar(){
  var tn1
  var tn2
}
```

Vamos fazer a ligação com a caixa, o input:

```
var tn1 = window.document.getElementById('txtn1')
var tn2 = window.document.querySelector('input#txtn2')
```

Podemos utilizar tanto o getElementById, como também o querySelector, vamos deixar um de cada para demonstrar que os dois funcionam.

Agora precisamos de uma variável para o valor do n1 e uma variável para o valor do n2, isso porque quando eles vem de uma caixa de texto, ele é texto.

Para isso, temos que fazer o seguinte, usar o number e pegar o tn1.value, que é o valor que está dentro dessa caixa de texto n1, a mesma coisa para o outro.

```
var n1 = Number(tn1.value)
var n2 = Number(tn2.value)
```

Agora que temos o valor de n1 e o valor de n2 em forma de número, porque se somar sem essa conversão de número, ele ia concatenar, vamos criar uma variável s que vai ser n1 + n2.

```
var s = n1 + n2
```

Vamos criar o resultado dentro de uma **<div>** com um id res.

```
<div id="res">Resultado</div>
```

Vamos criar var res para trabalhar o resultado.

```
var res = window.document.getElementById('res')
```

Agora eu vou fazer com que o resultado saia na tela.

```
res.innerHTML = `A soma entre ${n1} e ${n2} é igual a ${s}`
```

Vamos criar um style para o body normal com font 18pt Arial, aumentar a fonte do input, fonte, normal, 18pt Arial, a largura, Width 100 pixels, quebrando as regras vamos colocar um margin top de 20 pixels, só para dar uma espaçada.

```
<style>
  Body {
    font: normal 18pt Arial;
  }
  input {
    font: normal 18pt Arial;
    width: 100px;
  }
  div#res {
    margin-top: 20px;
  }
</style>
```

E a **<div>** como usamos o innerHTML, podemos colocar o resultado em destaque, com o ****.

```
res.innerHTML = `A soma entre ${n1} e ${n2} é igual a <strong>${s}</strong>`
```

Agora estamos com o código funcionando, bem bonito. Temos a parte HTML, que é a parte do conteúdo, temos a parte CSS, que é a parte de estilo, e temos a parte de interatividade, que é a parte de JavaScript.

As três partes sendo criadas no mesmo exercício.