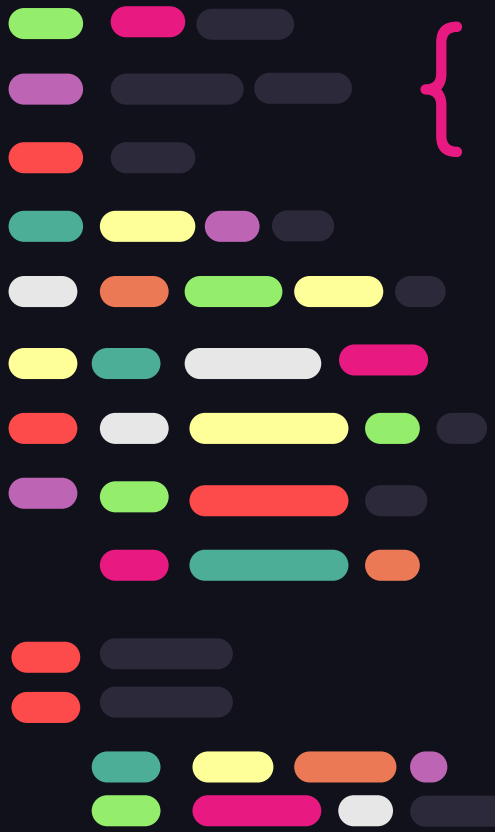


Javascript
1 - Introdução

< Por Isis =) >

01 { ..

Por onde começar

< Uma pequena introdução ao tema >

} ..

Por que Javascript?

Se tratando de uma página web, o HTML é quem define a estrutura, descrevendo cada elemento presente, e o CSS define a aparência. No entanto, o HTML e o CSS são majoritariamente estáticos, eles não mudam e não causam mudanças.

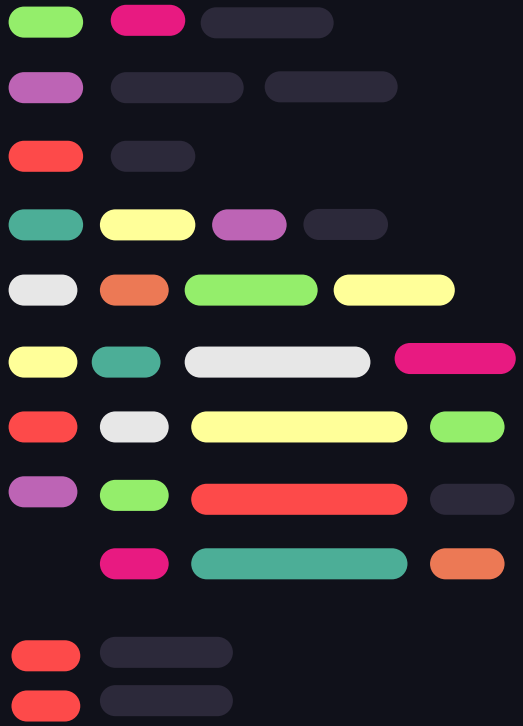
Sem uma linguagem de programação por trás, um botão realiza nenhuma ação ao ser clicado, animações são limitadas, cálculos não são possíveis, formulários não são enviados a lugar algum. É o Javascript que permite ações e eventos.

Como funciona?

O script pode ser declarado diretamente no corpo do arquivo HTML com a tag <script> ou pode ser declarado em um arquivo próprio e ser referenciado no head do HTML, usando a mesma tag.

Apesar de ser conveniente para códigos curtos, declarar o código diretamente no arquivo HTML não é viável em códigos mais extensos. Para melhor organização, é preferível escrever o código em um arquivo “.js”.

<head> ou <body>



Dependendo de onde a tag <script> for declarada, a ordem de carregamento do código varia.
Se for declarado no <head>, o script é carregado antes do corpo da página.
Se for declarado no <body>, ele é carregado apenas junto com o conteúdo da página. Isso pode afetar o funcionamento do código, então é preciso prestar atenção.

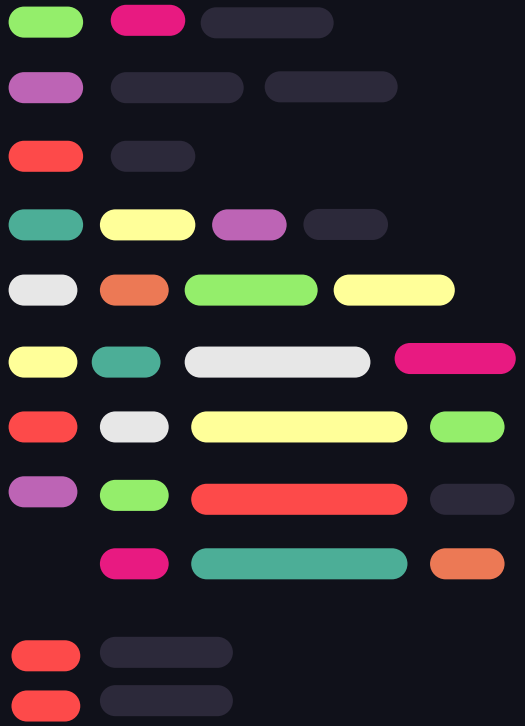
02 { ..

Variáveis e constantes

< Armazenando e utilizando valores >

} ..

Variáveis

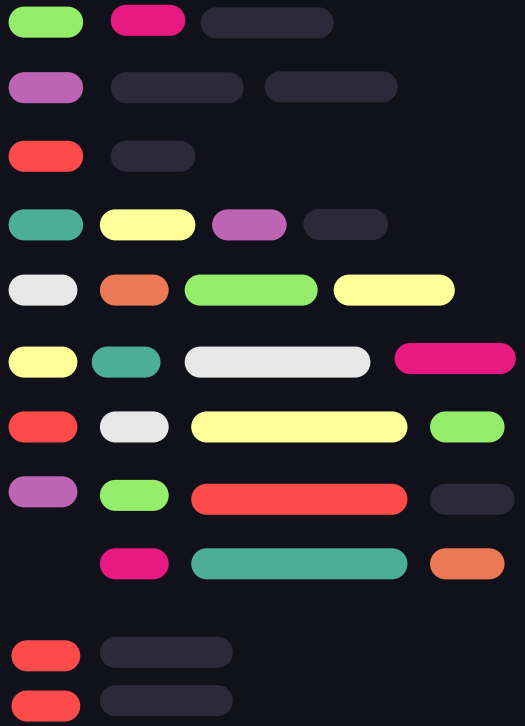


Variáveis são um dos conceitos mais fundamentais da programação.

Elas são como caixas para guardar uma informação que você precise utilizar. Por exemplo, um nome de usuário ou um número para cálculo podem ser armazenados em uma variável.

O propósito delas é que seus conteúdos mudem com a execução do código.

Constantes



Variáveis NÃO são constantes.

Variáveis mudam de valor, constantes não.

Muito cuidado para não confundir esses dois conceitos.

Tipos de declaração

var

Forma mais antiga de declarar variáveis. É possível declarar mais de uma com mesmo nome, em escopo diferente

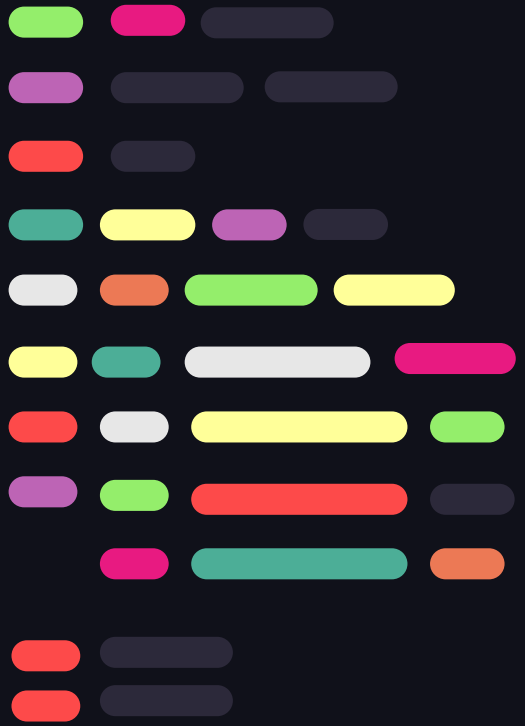
let

Forma mais segura de declarar variáveis. Permite apenas uma declaração de mesmo nome e é presa ao escopo.

const

Utilizada para declaração de constantes. O valor não pode ser mudado.

Escopo



Este é um conceito um pouco abstrato para este início.

Por enquanto, basta entender que `var` é mais aberto a problemas que o `let`.

Para causar menos confusão, preferimos o uso do `let`. Ele é uma versão mais moderna e mais segura de declarar variáveis.

Tipos de variável

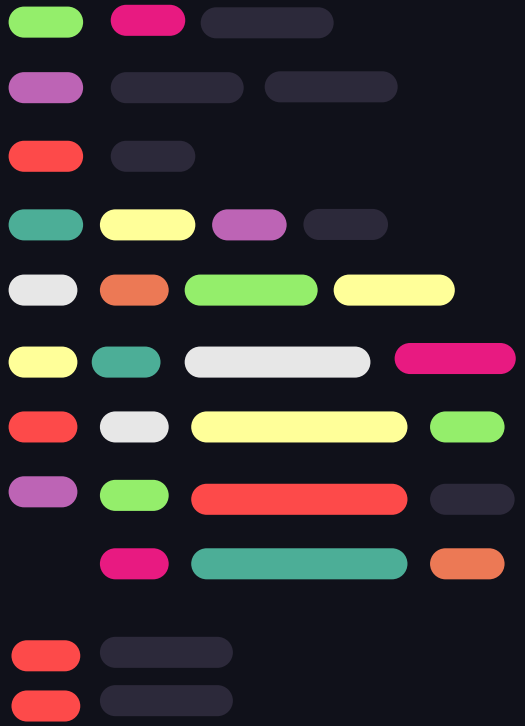
Variáveis podem abrigar diferentes tipos de dados, como números, texto, ou objetos, como imagens ou datas.

Certas linguagens exigem que o tipo seja definido na declaração da variável.

Javascript não é uma linguagem fortemente tipada, ou seja, não exige o tipo de dado na declaração.

Mesmo assim, ainda é muito útil entender esses diferentes tipos de dados e as suas propriedades.

String



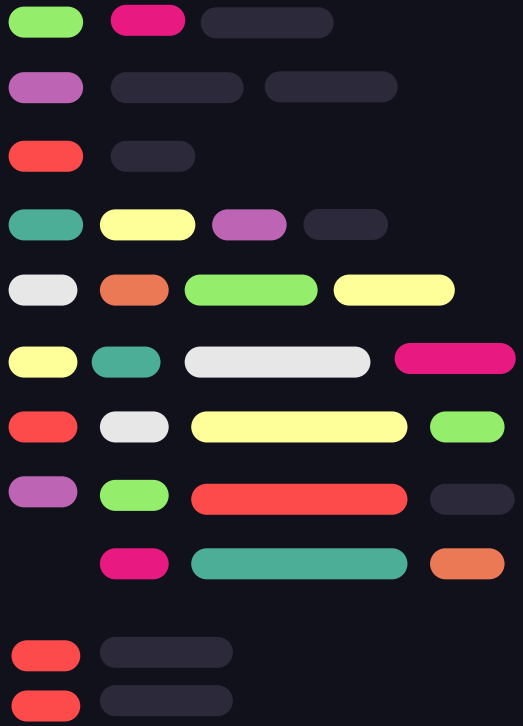
String é um tipo de dado dedicado para texto.

Por exemplo:

```
let nome = ‘Letícia’;
```

Strings recebem este nome porque elas são uma sequência (linha) de caracteres.

Number e Bigint



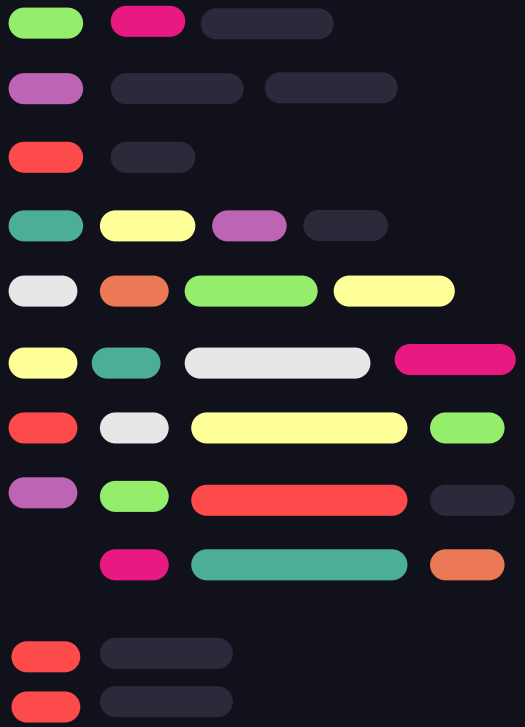
Number, como o nome já diz, é um tipo de variável para abrigar números.

Exemplo:

```
let a = 3.5;
let b = 25;
```

Bigint também é um tipo numérico, mas com duas propriedades específicas: ele é inteiro e permite números muito grandes.

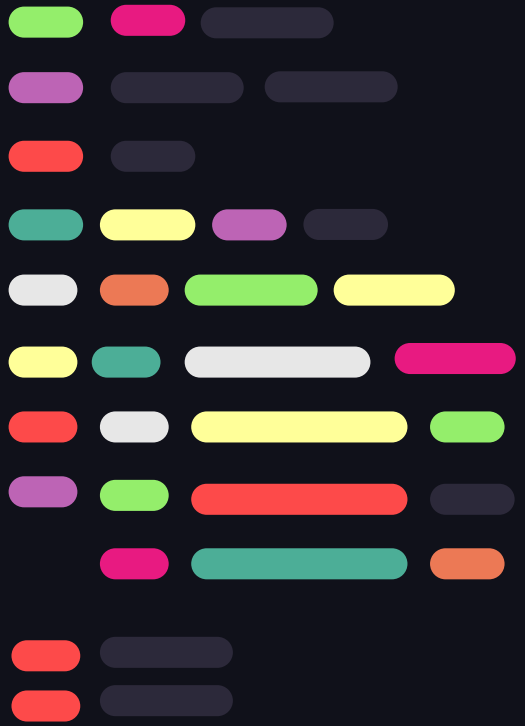
Boolean



Boolean (booleano, vindo do nome do matemático George Boole) é uma variável binária, ou seja, ela assume apenas dois valores: true (verdadeiro) ou false (falso).

Este tipo de dado é muito importante para a lógica computacional, e é utilizado extensivamente em toda programação. Mais pra frente veremos o porquê disso.

Undefined e Null



O Undefined (indefinido) é o tipo padrão para uma variável a qual ainda não foi atribuído um valor.

O Null (nulo) é o tipo de dado que utilizamos caso queiramos que o conteúdo seja “vazio”.

Ambos indicam ausência de dados.

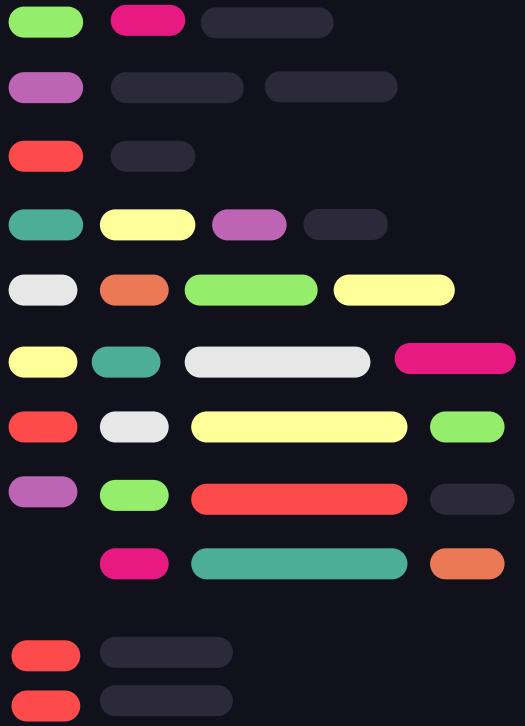
03 { ..

Funções e eventos

< Fazendo algo acontecer >

} ..

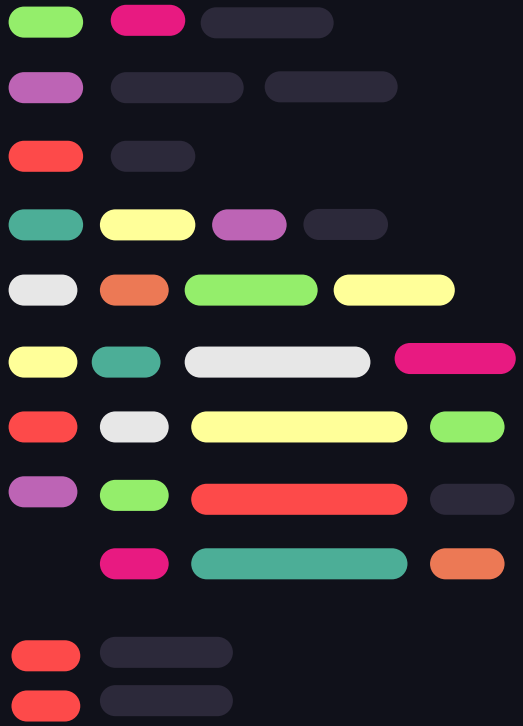
Funções



Funções (functions) são blocos de código que podem ser executados ao chamá-los.

Funções são muito boas em organizar o código e permitem a reutilização de sequências específicas de passos em vários locais diferentes.

Entrada e saída

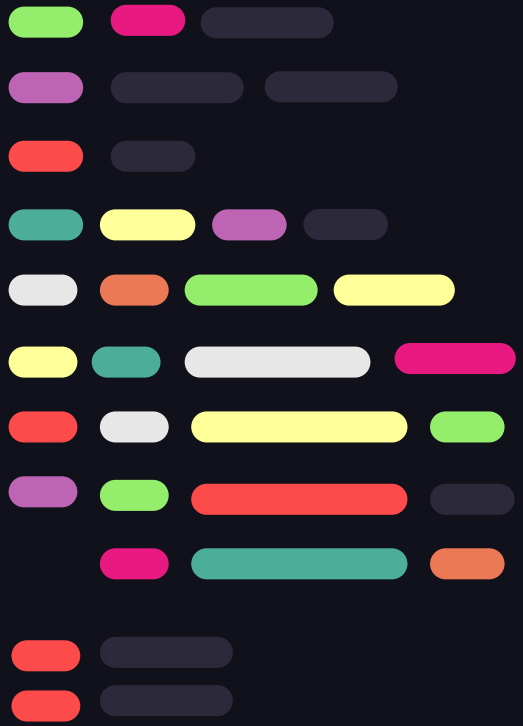


Em algoritmos, sempre pensamos em termos de entrada e saída de dados.

Por exemplo, na soma entre dois números, nossa entrada são um número a e um número b, e a saída é a soma de a e b.

Em um login, a entrada é o usuário e a senha, e a saída é um “sim” ou “não”, dependendo de se as credenciais são válidas.

Parâmetros e retorno

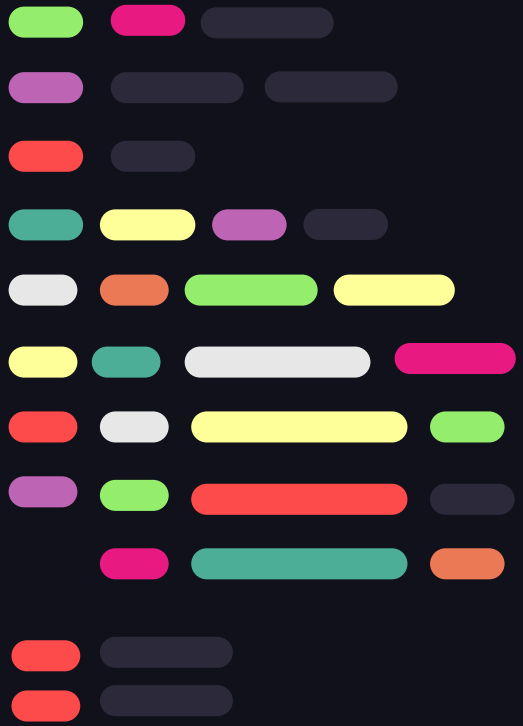


Nós chamamos os valores de entrada de uma função de parâmetros. A saída é o retorno da função.

Exemplo:

```
function soma(a, b) {
    return a + b;
}
```

Utilizando uma função



Para invocar uma função é muito simples.
Utilizando o exemplo anterior:

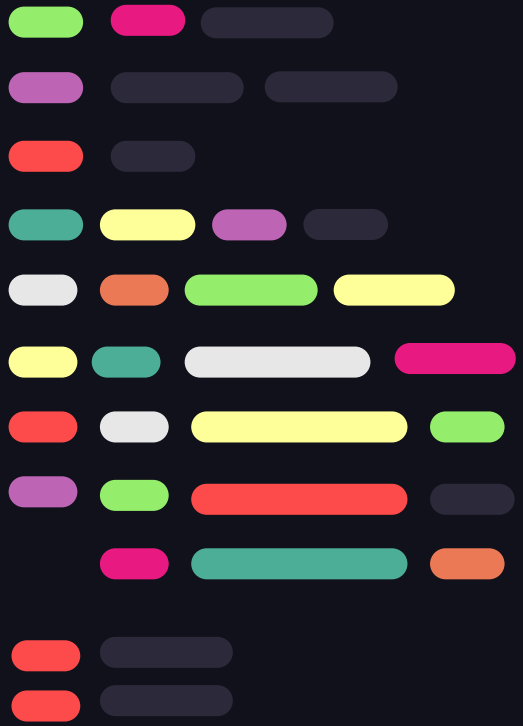
```
let s = soma(2,3);
```

Neste caso, a = 2 e b = 3.
A variável s irá receber o retorno da função soma(), então s = 5.

```
let r = soma(2,2);
```

Neste caso, r = 4.

Funções sem retorno



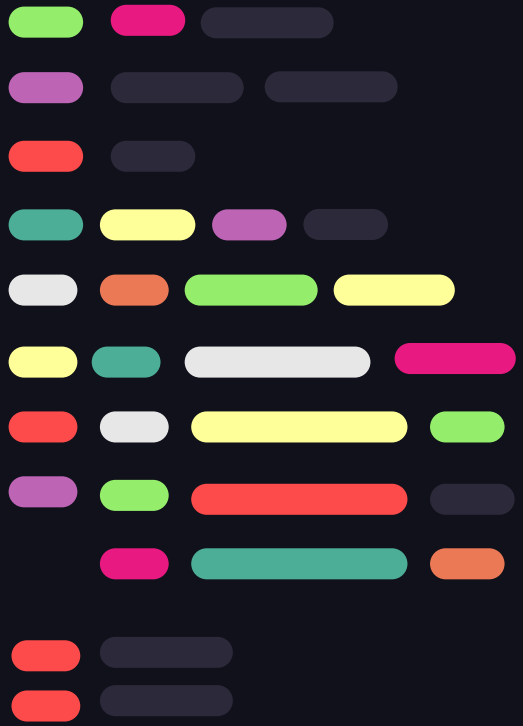
Uma função não precisa retornar um valor.
Por exemplo:

```
function hello(){
    alert(‘Hello world’);
}
```

Neste caso, ela é invocada por si só:

```
hello();
```

Funções pré-definidas

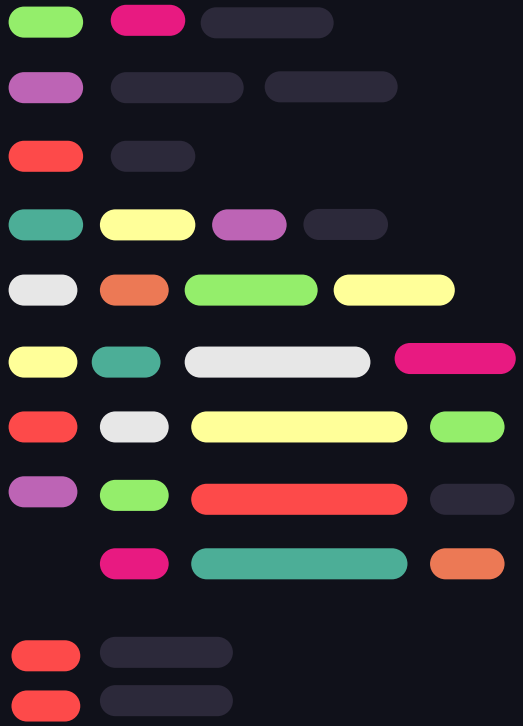


Algumas funções vêm pré-definidas na própria linguagem do Javascript!

O alert() é uma delas. Esta exibe uma mensagem graciosa na tela do navegador.

Antes de sair programando alguma coisa, é interessante procurar se já não existe uma função pré-definida ;)

Eventos



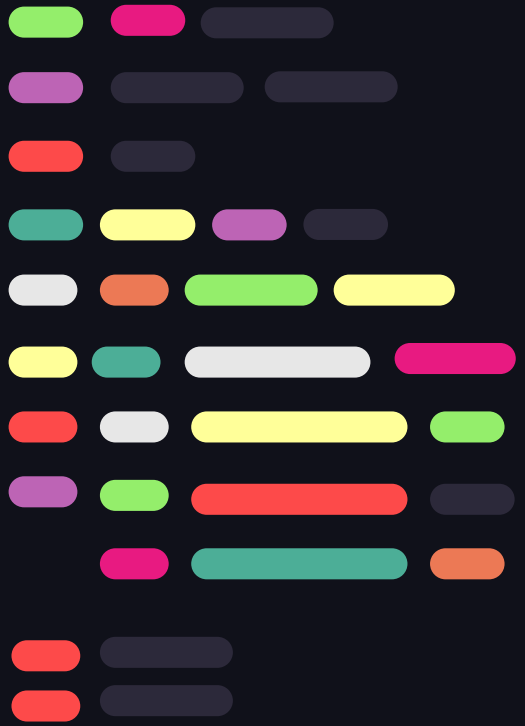
Eventos são gatilhos de execução de código.

Por exemplo:

- Um botão ser clicado
- Digitar em um campo de texto
- Selecionar uma opção
- Carregamento da página ou de um elemento

Existem muitos e muitos tipos de eventos!!

Como os dois interagem

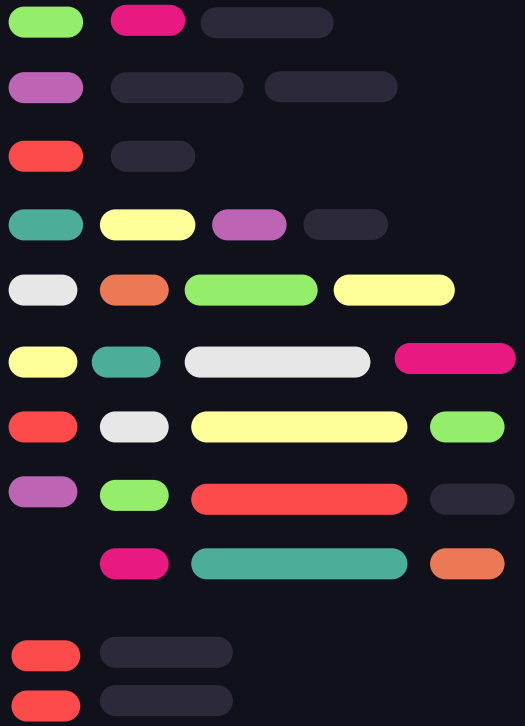


Um evento é normalmente vinculado a uma sequência de passos.

Podemos assim vincular um evento a uma função.

Por exemplo, se eu tiver uma função que realiza um cálculo e exibe na tela, eu posso atribuir o evento de clique a um botão, que utilizaria essa função. Assim, o cálculo seria realizado após o clique.

No HTML



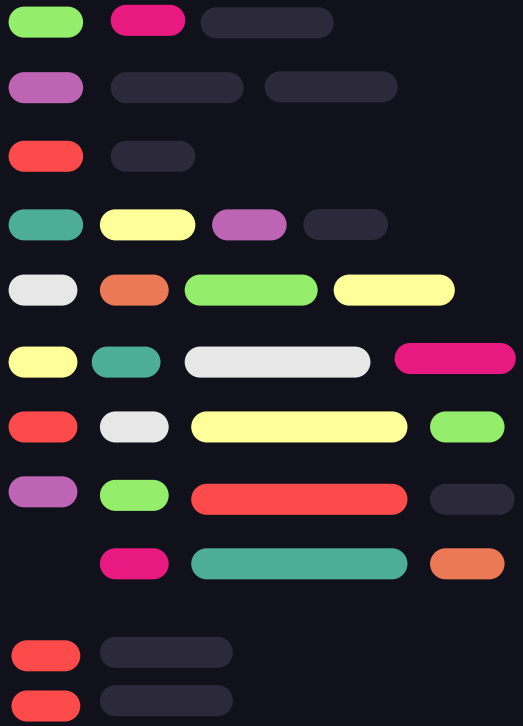
O evento é vinculado a uma tag, através de um atributo do nome do evento.

Exemplo:

```
<button onclick=”hello()”>Clique aqui!</button>
```

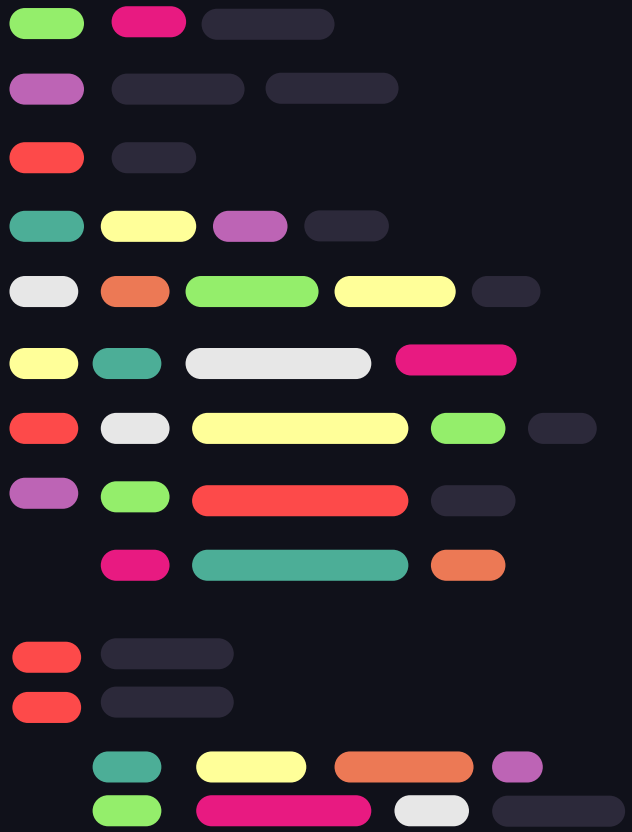
Neste caso, ao clicar no botão, a função hello() é invocada.

Alguns eventos



- onclick: ao clicar no elemento
- onchange: ao sofrer uma alteração
- onmouseover: ao passar o mouse por cima
- onmouseout: ao tirar o mouse de cima
- onkeydown: ao pressionar tecla
- onload: ao carregar o elemento
- onsubmit: ao enviar um formulário

Tem muitos outros:
https://www.w3schools.com/jsref/dom_obj_event.asp



Yey! }

< Sabendo de tudo isso, já podemos começar a olhar nossos exemplos >