

Aula 3

Criando objetos

► **Unidade**

**Funções: criando uma missão
sobre Inteligência Artificial**

O que vamos aprender?



Criar um código de referência aos objetos HTML.



Compreender listas e objetos.



Criar uma lista e alguns objetos de perguntas.



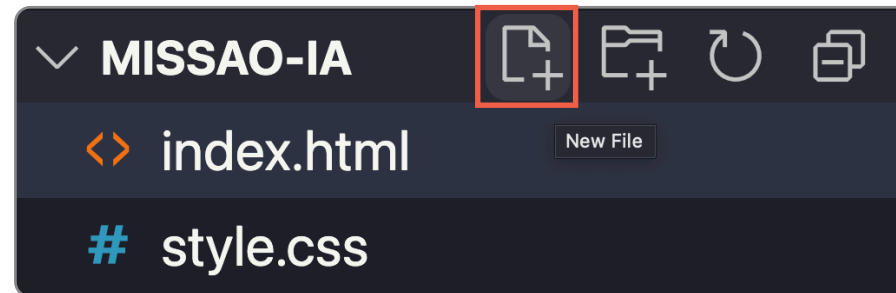
ACOMPANHE O VÍDEO DA AULA

Estruturando perguntas

Na aula anterior, analisamos um protótipo e construímos o HTML de base da nossa aplicação. Nesta aula, começaremos a utilizar JavaScript para construir as perguntas do nosso projeto na forma de objetos.


```
const perguntas = [  
  {  
    enunciado: "Pergunta 1",  
    alternativas: ["Alternativa 1", "Alternativa 2"],  
  },  
  
  {  
    enunciado: "Pergunta 2",  
    alternativas: ["Alternativa 1", "Alternativa 2"],  
  },  
]
```

De volta ao VSCode, a primeira coisa que precisamos fazer para trabalhar com JavaScript é criar um arquivo com extensão .js e estabelecer o link desse novo arquivo com o arquivo HTML. Você pode criar o arquivo JS clicando no botão *New File* (Novo Documento), ao lado do nome da pasta, localizado no canto superior esquerdo da tela. Nomeie seu novo arquivo como *script.js*.



Agora, abriremos o arquivo HTML para criar o link com o *script.js* que acabamos de criar. Ao fim do HTML, antes de fechar a tag **</body>**, você pode adicionar o atalho **script:src** e apertar *Enter*. O seguinte texto deve aparecer:

```
<script src=""></script>
```

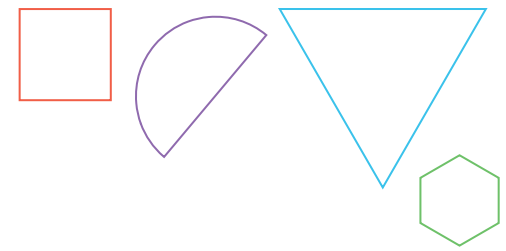


Agora, basta adicionar o nome do arquivo de referência no campo src e o link entre os dois arquivos estará feito.

```
<script src="script.js"></script>
```

Feito isso, abriremos o arquivo *script.js* e começaremos a chamar os elementos que criamos no HTML na aula anterior para dentro do código. Para isso, usaremos uma constante, pois, uma vez que chamarmos esse elemento, a variável não deve ficar mudando de valor.

Para acessar a classe **.caixa-principal**, que criamos na aula anterior, vamos usar o método 'querySelector' do objeto document. Você deve se lembrar que o objeto document refere-se ao DOM, objeto que contém os conteúdos da nossa página.



No arquivo *script.js*, seu código ficará da seguinte forma:

```
const caixaPrincipal = document.querySelector('.caixa-principal');
```

Não se esqueça do ponto em **.caixa-principal**, pois estamos referenciando uma classe.

Agora, podemos replicar o código anterior, referenciando todas as classes que criamos na aula anterior. Você pode copiar e colar a linha que escrevemos, e alterar apenas o nome da variável e o nome da classe referenciada. Observe o resultado em seu código:


```
const caixaPrincipal = document.querySelector('.caixa-principal');
const caixaPerguntas = document.querySelector('.caixa-perguntas');
const caixaAlternativas = document.querySelector('.caixa-alternativas');
const caixaResultado = document.querySelector('.caixa-resultado');
const textoResultado = document.querySelector('.texto-resultado');
```



Agora que já fizemos referência a todas as nossas classes, vamos começar a criar as perguntas.

Você deve se lembrar de que existe um tipo de dado chamado lista. Ele nada mais é do que uma forma de salvar uma lista de vários itens dentro de uma variável. Por exemplo:

```
const lista = [item1, item2]
```




Desse modo, em vez de criarmos uma variável para cada pergunta, podemos criar uma única variável para todas elas, que será uma lista de perguntas. Observe:

```
const perguntas = []
```

Mas qual seria a melhor forma de criar nossas perguntas? Imagine um objeto qualquer, como um lápis. Ele tem uma série de atributos como tamanho, tipo, cor, tem ou não tem borracha na ponta, entre outros. Em programação, poderíamos criar um objeto lápis da seguinte forma:

```
const lapis = {  
  tamanho: 20,  
  tipo: 'HB',  
  cor: 'Grafite',  
  temBorrachaAtras: false  
}
```

O uso do objeto facilita a programação, pois teremos um único objeto lápis e toda a lista de atributos dentro dele. Além disso, se quisermos criar outro lápis, ele já terá os mesmos atributos do primeiro.




No caso das perguntas para o nosso projeto, temos a mesma situação. Embora tenhamos diversas perguntas diferentes, todas elas terão os mesmos atributos: um enunciado e alternativas.

Nesse sentido, podemos colocar dentro da lista **perguntas** um objeto de pergunta, da seguinte forma:

```
const perguntas = [  
  {  
    enunciado: "Pergunta 1",  
    alternativas: [  
      "Alternativa 1",  
      "Alternativa 2"  
    ]  
  }  
]
```

Depois, podemos alterar o conteúdo, mas perceba que o atributo **enunciado** já pode receber o enunciado da pergunta, enquanto o atributo **alternativas** também já pode receber uma lista de alternativas.



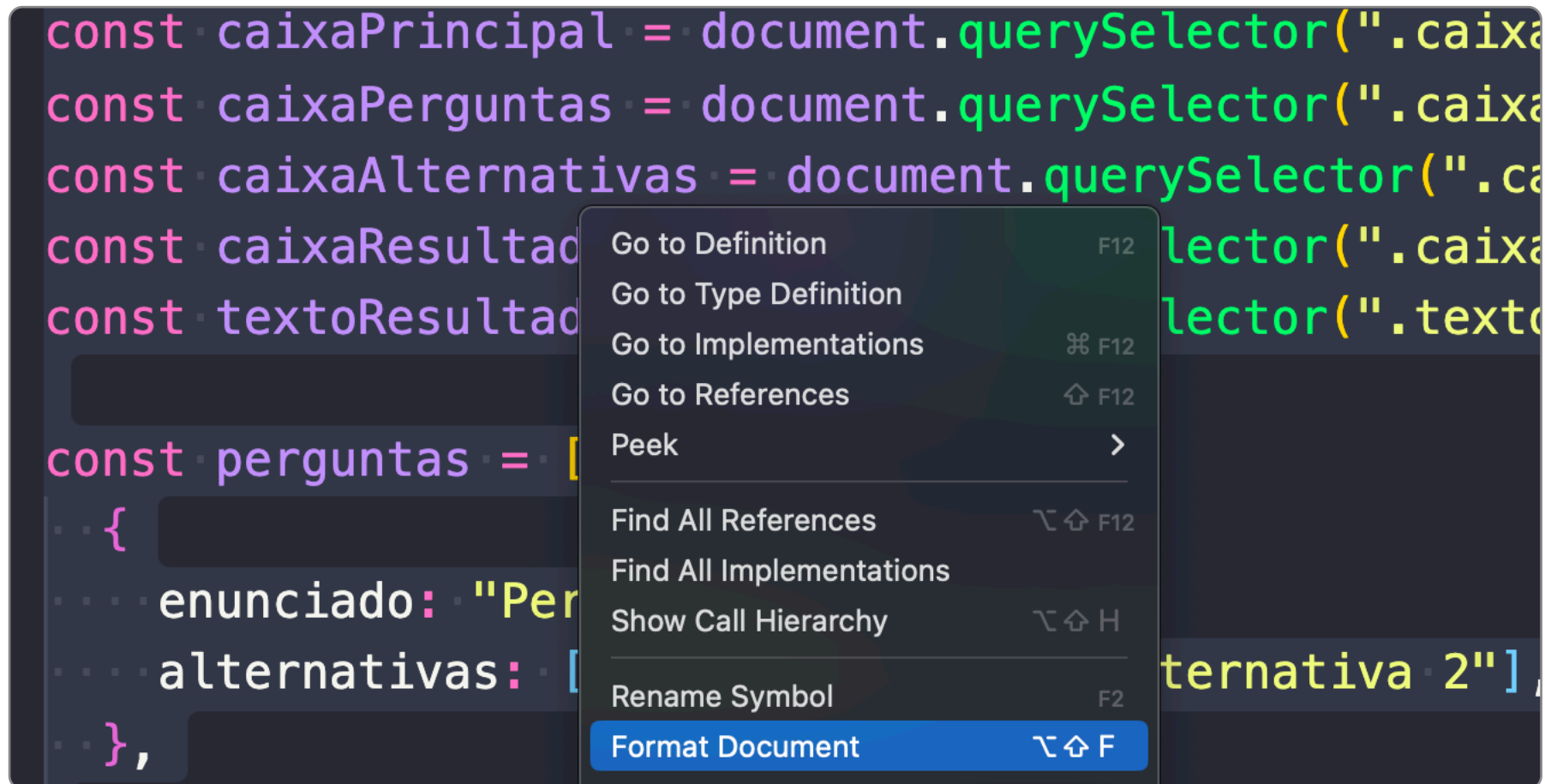
Preste bastante atenção na estrutura que montamos: temos uma lista chamada perguntas e, dentro dela, um objeto de pergunta com dois atributos: enunciado e alternativas. Observe:

```
const perguntas = [ //abre lista de perguntas
  { //abre objeto de pergunta
    enunciado: "Pergunta 1",
    alternativas: [
      "Alternativa 1",
      "Alternativa 2"
    ]
  } //fecha objeto de pergunta
] //fecha lista de perguntas
```

Para adicionarmos mais perguntas, basta copiarmos nosso objeto. Lembre-se de que, em uma lista, os itens são separados por vírgula.

```
const perguntas = [  
  {  
    enunciado: "Pergunta 1",  
    alternativas: [  
      "Alternativa 1",  
      "Alternativa 2"  
    ]  
  },  
  {  
    enunciado: "Pergunta 2",  
    alternativas: [  
      "Alternativa 1",  
      "Alternativa 2"  
    ]  
  }  
]
```

Para garantir a formatação do seu código, deixando-o fácil de ler, você pode selecionar o texto que quer formatar, clicar com o botão direito sobre ele e escolher a opção *Format Document* (Formatar Documento):



```
const caixaPrincipal = document.querySelector(".caixaPrincipal");
const caixaPerguntas = document.querySelector(".caixaPerguntas");
const caixaAlternativas = document.querySelector(".caixaAlternativas");
const caixaResultado = document.querySelector(".caixaResultado");
const textoResultado = document.querySelector(".textoResultado");

const perguntas = [
  {
    enunciado: "Pergunta 1",
    alternativas: [
      { texto: "Alternativa 1", correta: false },
      { texto: "Alternativa 2", correta: true },
      { texto: "Alternativa 3", correta: false },
      { texto: "Alternativa 4", correta: false }
    ]
  }
];
```



Agora, seu código deve estar semelhante ao código a seguir (observe que a formatação pode variar um pouco):

```
const caixaPrincipal = document.querySelector(".caixa-  
principal");  
const caixaPerguntas = document.querySelector(".caixa-  
perguntas");  
const caixaAlternativas = document.querySelector(".caixa-  
alternativas");  
const caixaResultado = document.querySelector(".caixa-  
resultado");  
const textoResultado = document.querySelector(".texto-  
resultado");  
const perguntas = [  
  {  
    enunciado: "Pergunta 1",  
    alternativas: ["Alternativa 1", "Alternativa 2"],  
  },  
  {  
    enunciado: "Pergunta 2",  
    alternativas: ["Alternativa 1", "Alternativa 2"],  
  },  
];
```

► Desafio

Nesta aula, iniciamos nosso código em JavaScript. Fizemos as referências às classes do HTML e salvamos seu conteúdo em variáveis constantes, além de criarmos uma lista de perguntas contendo objetos de pergunta com os atributos enunciado e alternativas.

Seu desafio, portanto, é criar seus objetos, um dos conceitos mais importantes da programação. Pense em algo que você goste e use diariamente e que tenha um código por trás. Pode ser um aplicativo, como o Instagram ou o WhatsApp, um jogo que você joga no computador, qualquer coisa que desperte seu interesse. Tente refletir sobre quais elementos desses aplicativos ou *softwares* podem ser objetos. Utilizando o próprio VSCode, crie seu objeto com, pelo menos, dois elementos.



CLIQUE AQUI PARA AVALIAR ESTE MATERIAL