Armazenando as informações que você precisa - Variáveis

Samuel Amaro

30 de março de 2021

Neste artigo, iremos ao básico, vendo como trabalhar com a maioria dos blocos de construção básicos de JavaScript - Variáveis.

Oque é uma variavel ?

Uma variável é um container para um valor, como um número que podemos usar em uma operação de adição, ou uma sequência de texto que possamos usar como parte de uma oração. Mas uma coisa especial a respeito das variáveis é que seu conteúdo pode mudar. Vamos ver um exemplo simples:

```
1
2       <button>Aperte-me</button>
3
4
```

Listing 1: Exemplo de variavel

```
1
2       var button = document.querySelector('button');
3
4       button.onclick = function() {
5           var nome = prompt('Qual    o seu nome?');
6           alert('Ol   ' + nome + ',    um prazer te ver!');
7       }
8
9
```

Listing 2: Exemplo de variavel

Nesse exemplo, apertar o botão executa algumas linhas de código. A primeira linha exibe uma caixa pop-up na tela que pede ao leitor para inserir o seu nome, e então armazena o valor na variável. A segunda linha exibe uma mensagem de boas vindas que inclui seu nome, obtido do valor da variável.

Para entender porque isso é útil, vamos pensar sobre como nós escreveríamos esse exemplo sem usar uma variável. Iria acabar se parecendo com algo do tipo:

```
1
2         var nome = prompt('Qual    o seu nome?');
3
4         if (nome === 'Ad o') {
5           alert('Ol   Ad o,    um prazer te ver!');
6         } else if (nome === 'Alan') {
7           alert('Ol   Alan,    um prazer te ver!');
8         } else if (nome === 'Bella') {
9           alert('Ol   Bella,    um prazer te ver!');
10        } else if (nome === 'Bianca') {
11          alert('Ol   Bianca,    um prazer te ver!');
12        } else if (nome === 'Chris') {
13          alert('Ol   Chris,    um prazer te ver !');
14        }
15
16        // ... e assim por diante ...
17
18
```

Você talvez não entenda totalmente a sintaxe que estamos usando (ainda!), mas deve ter pegado a ideia — se nós não tivermos variáveis disponíveis teríamos que implementar um bloco de código gigantesco para conferir qual era o nome inserido, e então exibir a mensagem apropriada para aquele nome. Isso é obviamente muito ineficiente (o código é muito maior, mesmo para apenas quatro opções de nome), e simplesmente não funcionaria — você não poderia armazenar todas as possíveis opções de nome.

Variáveis simplesmente fazem sentido, e a medida que você for aprendendo mais sobre JavaScript elas começarão a se tornar uma coisa natural.

Outra coisa especial sobra as variáveis é que elas podem conter praticamente qualquer coisa — não apenas cadeias de texto e números. Variáveis também podem conter dados complexos e até mesmo funções completas para fazer coisas incríveis. Você irá aprender mais sobre isso a medida que continuarmos.

Nota: Perceba que dissemos que as variáveis contém valores. Essa é uma distinção importante a se fazer. Elas não são os valores; e sim os containers para eles. Você pode pensar nelas sendo pequenas caixas de papelão nas quais você pode guardar coisas.

boxes.png Ilustração de como as variaveis podem ser

Declarando uma variável

Para usar uma variável primeiro tem que criá-la — mais precisamente, chamamos isso de declarar a variável. Para fazê-lo digitamos a palavra chave `var` seguido do nome que desejamos dar à variável:

```
var meuNome;
var minhaIdade;
```

Aqui, estamos criando duas variáveis chamadas `meuNome` e `minhaIdade`. Tente agora digitar essas linhas no console do seu navegador. Depois disso, tente criar uma variável (ou duas) com suas próprias escolhas de nomes.

Nota: No JavaScript, todas as intruções em código deve terminar com um ponto e vírgula (;) — seu código pode até funcionar sem o ponto e vírgula em linhas únicas, mas provavelmente não irá funcionar quando estiver escrevendo várias linhas de código juntas. Tente pegar o hábito de sempre incluir o ponto e vírgula.

Você pode testar se os valores agora existem no ambiente de execução digitando apenas os nomes das variáveis, ex.:

```
meuNome;
minhaIdade;
```

```
scoobyDoo;
```

Nota: Não confunda uma variável que existe mas não tenho valor definido com uma variável que não existe — são coisas bem diferentes.

Inicializando uma variável

Uma vez que você declarou uma variável, você pode inicializá-la com um valor. Você faz isso digitando o nome da variável, seguido do sinal de igual (=) e o valor que deseja atribuir a ela. Por exemplo:

```
meuNome = 'Chris';
minhaIdade = '37';
```

```
```

Tente voltar ao console agora e digitar essas linhas acima. Você deve ver o valor que atribuiu à variável retornado no console confirmando-o, em cada caso. De novo, você pode retornar os valores de suas variáveis simplesmente digitando seus nomes no console — tente isso novamente:

```
meuNome;
minhaIdade;
```

Você pode declarar e inicializar uma variável ao mesmo tempo, assim:

```
var meuNome = 'Chris';
```

Isso provavelmente é como irá fazer na maioria das vezes, já que é mais rápido do que fazer as duas ações em duas linhas separadas.

A diferença entre var e let

Agora você pode estar pensando "por que precisamos de duas palavras-chave para definir variáveis? Por que `var` e `let`?".

As razões são um tanto históricas. Quando o JavaScript foi criado, havia apenas `var`. Isso funciona basicamente bem na maioria dos casos, mas tem alguns problemas na maneira como funciona — seu design pode ser confuso ou totalmente irritante. Portanto, `let` foi criada nas versões modernas de JavaScript, uma nova palavra-chave para criar variáveis que funcionam de maneira um pouco diferente para `var`, corrigindo seus problemas no processo.

Algumas diferenças simples são explicadas abaixo.

Para começar, se você escrever um programa JavaScript de várias linhas que declare e inicialize uma variável, poderá realmente declarar uma variável com `var` depois de inicializá-la e ainda funcionará. Por exemplo:

```
meuNome = 'Chris';

function logNome() {
  console.log(meuNome);
}
```

```
7
8         logNome();
9
10        var meuNome;
11
12
```

Nota: Isso não funcionará ao digitar linhas individuais em um console JavaScript, apenas ao executar várias linhas de JavaScript em um documento da web.

Isso funciona por causa do **hoisting**

Hoisting não funciona mais com `let`. Se mudássemos de `var` para `let` no exemplo acima, teríamos um erro. Isso é bom — declarar uma variável depois que você a inicializa resulta em código confuso e difícil de entender.

E depois, ao usar var, você pode declarar a mesma variável quantas vezes quiser, mas com let você não consegue. Isso pode funcionar:

```
1
2         var meuNome = 'Chris';
3         var meuNome = 'Bob';
4
5
```

Mas isso geraria um erro na segunda linha:

```
1
2       let meuNome = 'Chris';
3       let meuNome = 'Bob';
4
5
```

Você precisaria fazer assim:

```
1
2       let meuNome = 'Chris';
3       meuNome = 'Bob';
4
5
```

Novamente, essa é uma decisão sensata da linguagem. Não há razão para redeclarar variáveis — isso apenas torna as coisas mais confusas.

Por esses motivos e mais, recomendamos que você use `let` o máximo possível em seu código, em vez de `var`. Não há motivo para usar `var`, a menos que você precise oferecer suporte para versões antigas do Internet Explorer com seu código (ele não suporta `let` até a versão 11; o navegador mais moderno do Windows, o Edge, suporta `let`).

Atualizando uma variável

Uma vez que uma tenha um valor atribuido, você pode atualizar esse valor simplesmente dando a ela um valor diferente. Tente inserir as seguintes linhas no seu console:

```
1
2       meuNome = 'Bob';
3       minhaIdade = 40;
4
5
```

Um adendo sobre regras de nomeação de variáveis

Você pode chamar uma variável praticamente qualquer nome que queira, mas há limitações. Geralmente você deve se limitar a utilizar somente caracteres latinos (0-9, a-z, A-Z) e o caractere underline (_).

- Você não deve usar outros caracteres porque eles podem causar erros ou ser difíceis de entender por uma audiência internacional.
- Não use underline no início do nome de variáveis — isso é utilizado em certos construtores JavaScript para significar coisas específicas, então pode deixar as coisas confusas.
- Não use número no início do nome de variáveis. Isso não é permitido e irá causar um erro.
- Uma conveção segura e se ater é a chamada "lower camel case, onde você junta várias palavras, usando minúscula para a primeira palavra inteira e, em seguita, maiusculiza a primeira letra das palavras subsequentes. Temos utilizado esse procedimento para os nomes das nossas variáveis nesse artigo até aqui.
- Faça nomes de variáveis intuitivos, para que descrevam o dado que ela contém. Não use letras ou números únicos, ou frases muito longas.
- As variáveis diferenciam letras maiúsculas e minúsculas — então `minhaidade` é uma variável diferente de `minhaIdade`.
- Um último ponto — você também precisa evitar utilizar palavras reservadas pelo JavaScript como nome para suas variáveis — com isso, queremos dizer as palavras que fazem parte da sintaxe do JavaScript! Então você não pode usar palavras como `var`, `function`, `let` e `for` como nome de variáveis. Os navegadores vão reconhecê-las como itens de código diferentes e, portanto, você terá erros.

```
idade
minhaIdade
inicio
corInicial
valorFinalDeSaida
audio1
audio2
```

Listing 3: Exemplo de bons nomes

```
1
a
_12
minhaidade
MINHAIDADE
var
Document
skjfndskjfnbdskjfb
esseeumnomedevariavelbemlongoeestupido
```

Listing 4: Exemplo ruins de nomes

Tente criar mais algumas variáveis agora, com a orientação acima em mente.

Tipos de variáveis

Existem alguns diferentes tipos de dados que podemos armazenar em variáveis. Nessa seção iremos descrevê-los brevemente, e então em artigos futuros você aprenderá sobre eles em mais detalhes.

Até agora nós vimos os dois primeiros, mas há outros.

Números

Você pode armazenar números em variáveis, tanto números inteiros, como por exemplo 30 (também chamados de integers) como números decimais, por exemplo 2.456 (também chamados de floats ou floating point numbers). Você não precisa declarar tipos de variáveis no JavaScript, diferentemente de outras linguagens de programação. Quando você atribui a uma variável o valor em número, você não inclui as aspas:

```
1
2           var minhaIdade = 17;
3
4
```

Listing 5: Tipo numeros

Strings(cadeias de texto)

Strings são sequências de texto. Quando você dá a uma variável um valor em texto (string), você precisa envolver o texto em aspas simples ou duplas; caso contrário, o JavaScript vai tentar interpretá-lo como sendo outro nome de variável.

```
1
2           var despedidaGolfinho = 'At  logo e obrigado por todos
    os peixes!';
3
4
```

Listing 6: Tipo Strings

Booleans(boleanos)

Booleans são valores verdadeiro/falso (true/false) — eles podem ter dois valores, `true` (verdadeiro) ou `false` (falso). São geralmente usados para verificar uma condição, que em seguida o código é executado apopriadamente. Por exemplo, um caso simples seria:

```
1
2           var estouVivo = true;
3
4
```

Listing 7: Tipo booleanos

Enquanto na realidade seria utlizado mais da seguinte forma:

```
1
2           var teste = 6 < 3;
3
4
```

Listing 8: Tipo booleanos

Esse exemplo está usando o operador "menor que"`(<)` para testar se 6 é menor que 3. Como você pode esperar, irá retornar `false` (falso), porque 6 não é menor que 3! Você aprenderá mais sobre tais operadores mais tarde no curso.

Arrays

Um array é um único objeto que contém valores múltiplos inseridos entre colchetes e separados por vírgulas. Tente inserir as seguintes linhas de código no seu console:

```
var meuNomeArray = ['Chris', 'Bob', 'Jim'];
var meuNumeroArray = [10,15,40];
```

Listing 9: Tipo arrays

Uma vez que esses arrays estejam definidos, você pode acessar cada um de seus valores através de sua localização dentro do array. Tente essas linhas:

```
meuNomeArray[0]; // deve retornar 'Chris'
meuNumeroArray[2]; // deve retornar 40
```

Listing 10: Tipo arrays

Os colchetes especificam um valor do índice correspondente à posição do valor que você deseja retornado. Você talvez tenha notado que os arrays em JavaScript são indexados a partir do zero: o primeiro elemento está na posíção 0 do índice.

Objetos

Em programação, um objeto é uma estrutura de código que representa um objeto da vida real. Você pode ter um simples objeto que representa um estacionamento de carro contendo informações sobre sobre sua largura e comprimento, ou você poderia ter um objeto que representa uma pessoa, e contém dados sobre seu nome, altura, peso, que idioma ela fala, como dizer olá a ela, e mais.

Tente inserir a seguinte linha em seu console:

```
var cachorro = {
     nome : 'Tot  ',
     raca : 'D lmata'
 };
```

Listing 11: Tipo objetos

Para obeter a informação armazenada no objeto, você pode usar a seguinte sintaxe:

```
1
2           cachorro.nome
3
4
```

Listing 12: Tipo objetos

Digitação permissiva

JavaScript é uma "dynamically typed language", o que significa que, diferente de outras linguagens, você não precisa especificar que tipo de dados uma variável irá conter (ex.: números, cadeias de texto, arrays, etc).

Por exemplo, se você declarar uma variável e dar a ela um valor encapsulado em aspas, o navegador irá tratar a variável como sendo uma string (cadeia de texto):

```
1
2           var minhaString = 'Ol  ';
3
4
```

Irá continuar sendo uma string, mesmo que dentro das apas contenha um número, então seja cuidadoso:

```
1
2           var meuNumero = '500'; // opa, isso continua sendo uma
    string
3           typeof(meuNumero);
4           meuNumero = 500; // bem melhor     agora isso    um
    n mero
5           typeof(meuNumero);
6
7
```

Tente inserir as quatro linhas acima em seu console uma por uma, e veja quais são os resultados. Você notará que estamos usando uma função especial chamada `typeof()` — ela irá retornar o tipo de dado da variável que você passar. Da primeira vez que for executada, deve retornar `string`, como naquele ponto a variável `meuNumero` contém uma string, `'500'`. Dê uma olhada e veja o que é retornado da segunda vez que você a utilizar.

Constants em JavaScript

Muitas linguagens de programação têm o conceito de constant — um valor que uma vez declarado não pode ser alterado. Há muitas razões pelas quais você deseja fazer isso, desde segurança (se um script de terceiros alterou esses valores, poderia causar problemas) até a depuração e a compreensão do código (é mais difícil alterar acidentalmente valores que não devem ser alterados e bagunçar as coisas).

Nos primeiros dias do JavaScript, não existiam constants. No JavaScript moderno, temos a palavra-chave `const`, que nos permite armazenar valores que nunca podem ser alterados:

```
1
2           const diasNaSemana = 7;
3           const horasNoDia = 24;
4
5
```

Listing 13: Constants JavaScript

`const` funciona exatamente da mesma maneira que `let`, exceto que você não pode atribuir um novo valor a `const`. No exemplo a seguir, a segunda linha geraria um erro:

```
1
2           const diasNaSemana = 7;
3           diasNaSemana = 8;
4
5
```