Matemática básica no JavaScript - números e operadores

Samuel Amaro

2 de abril de 2021

Neste ponto do curso estaremos discutindo matemática em JavaScript — Como podemos usar operadores e outros recursos para manipular números e fazer cálculos.

Objetivo: Ganhar familiaridade com o básico em matemática no JavaScript.

Todo mundo ama matemática

Ok, talvez não. Alguns de nós gostam de matemática, alguns de nós tem a odiado desde que tivemos que aprender a tabuada de multiplicação e divisão na escola, e alguns de nós estão em algum lugar no meio dos dois cenários. Mas nenhum de nós pode negar que a matemática é uma parte fundamental da vida sem a qual não poderíamos ir muito longe. Isso é especialmente verdadeiro quando estamos aprendendo a programar em JavaScript (ou em qualquer outra linguagem, diga-se de passagem) — muito do que fazemos se baseia no processamento de dados numéricos, cálculo de novos valores, etc. Assim você não ficará surpreso em aprender que o JavaScript tem uma configuração completa de funções matemáticas disponíveis.

Este artigo discute apenas as partes básicas que você precisa saber agora.

Tipos de números

Em programação, até mesmo o humilde sistema de números decimais que todos nós conhecemos tão bem é mais complicado do que você possa pensar. Usa-

mos diferentes termos para descrever diferentes tipos de números decimais, por exemplo:

- **Integers**(inteiros) são números inteiros, ex. 10, 400 ou -5.
- **Números de ponto flutuante**(floats) tem pontos e casas decimais, por exemplo 12.5 e 56.7786543.
- **Doubles** são tipos de floats que tem uma precisão maior do que os números de ponto flutuante padrões (significando que eles são mais precisos, possuindo uma grande quantidade de casas decimais).

Temos até mesmo diferentes tipos de sistemas numéricos! O decimal tem por base 10 (o que significa que ele usa um número entre 0–9 em cada casa), mas temos também algo como:

- **Binário** — A linguagem de menor level dos computadores; 0s e 1s.
- **Octal** — Base 8, usa um caractere entre 0–7 em cada coluna.
- **Hexadecimal** — Base 16, usa um caractere entre 0–9 e depois um entre a–f em cada coluna.

Antes de se preocupar com seu cérebro estar derretendo, pare agora mesmo! Para um começo, vamos nos ater aos números decimais no decorrer desse curso; você raramente irá se deparar com a necessidade de começar a pensar sobre os outros tipos, se é que vai precisar algum dia.

A segunda boa notícia é que, diferentemente de outras linguagens de programação, o JavaScript tem apenas um tipo de dados para números, você advinhou, `Number|`. Isso significa que qualquer que seja o tipo de números com os quais você está lidando em JavaScript, você os manipula do mesmo jeito.

Tudo é número para min

Vamos brincar rapidamente com alguns números para nos familiarizarmos com a sintaxe básica que precisamos. Insira os comandos listados abaixo em seu console JavaScript.

1. Primeiramente, vamos declarar duas variáveis e as inicializar com um integer e um float, respectivamente, então digitaremos os nomes das variávei para verificar se está tudo em ordem:

```
1
2           var meuInt = 5;
3           var meuFloat = 6.667;
4           meuInt;
5           meuFloat;
6
7
```

Listing 1: Exemplo de variavel

2. Valores numéricos são inseridos sem aspas — tente declarar e inicializar mais duas variáveis contendo números antes de seguir em frente.

3. Agora vamos checar se nossas duas variáveis originais são do mesmo tipo de dados. Há um operador chamado `typeof` no JavaScript que faz isso. Insira as duas linhas conforme mostradas abaixo:

```
1
2           typeof meuInt;
3           typeof meuFloat;
4
5
```

Listing 2: checando tipos da variaveis

Você deve obter `"number"` de volta nos dois casos — isso torna as coisas muito mais fáceis para nós do que se diferentes tipos de números tivessem diferentes tipos de dados, e tivéssemos que lidar com eles em diferentes maneiras. Ufa!

Operadores aritméticos

São os operadores que utilizamos para fazer as operações aritiméticas básicas:

Operador	Nome	Propósito	Exemplo
+	Adição	Adiciona um número a outro	6 + 9
-	Subtração	Subtrai o número da direita do número da esquerda.	20 - 15
*	Multiplicação	Multiplica um número pelo outro	3 * 7
/	Divisão	Divide o número da esquerda pelo número da direita	10 / 5
%	Restante(Remainder - as vezes chamado de modulo)	Retorna o resto da divisão em números inteiros do número da esquerda pelo número da direita	8 % 3 (retorna2; como três cabe duas vezes em 8, deixando 2 como resto.)

Nota: Você verá algumas vezes números envolvidos em aritimética chamados de operandos.

Nós provavelmente não precisamos ensinar a você como fazer matemática básica, mas gostaríamos de testar seu entendimento da sintaxe envolvida. Tente inserir os exemplos abaixo no seu console JavaScript

1. Primeiro tente inserir alguns exemplos simples por sua conta, como

```
1
2           10 + 7
3           9 * 8
4           60 % 3
5
6
```

Listing 3: Operacoes Matematicas Simples

2. Você pode tentar declarar e inicializar alguns números dentro de variáveis, e tentar usá-los nas operações — as variáveis irão se comportar exatamente como os valores que elas armazenam para a finalidade das operações. Por exemplo:

```
1
2           var num1 = 10;
3           var num2  = 50;
4           9 * num1;
5           num2 / num1;
6
```

```
7
```

Listing 4: Operacoes Matematicas Simples

3. Por último, nesta seção, tente inserir algumas expressões mais complexas, como:

```
1
2           5 + 10 * 3;
3           num2 % 9 * num1;
4           num2 + num1 / 8 + 2;
5
6
```

Listing 5: Operacoes Matematicas Simples

Alguns dos exemplos do último bloco podem não ter retornado os valores que você estava esperando; a seção abaixo pode lhe explicar o porquê

Precedência de operador

Vamos olhar para o último exemplo, assumindo que `num2` guarda o valor 50 e `num1` possui o valor 10 (como iniciado acima):

```
1
2       num2 + num1 / 8 + 2;
3
4
```

Como um ser humano, talvez você leia isso como "50 mais 10 é igual a 60", depois "8 mais 2 é igual a 10", e então "60 dividido por 10 é igual a 6".

No entanto seu navegador vai ler "10 dividido por 8 é igual a 1.25", então "50 mais 1.25 mais 2 é igual a 53.25".

Isto acontence por causa da **precedência de operadores** — algumas operações serão aplicadas antes de outras quando calcularmos o resultado de uma soma (referida em programação como uma expressão). A precedência de operadores em JavaScript é semelhante ao ensinado nas aulas de matemática na escola — Multiplicação e divisão são realizados primeiro, depois a adição e subtração (a soma é sempre realizada da esquerda para a direita).

Se você quiser substituir a precedência do operador, poderá colocar parênteses em volta das partes que desejar serem explicitamente tratadas primeiro. Então, para obter um resultado de 6, poderíamos fazer isso:

```
1
2       (num2 + num1) / (8 + 2);
3
4
```

Tente fazer e veja como fica.

Operadores de incremento e decremento

Às vezes você desejará adicionar ou subtrair, repetidamente, um valor de uma variável numérica. Convenientemente isto pode ser feito usando os operadores incremento `++` e decremento `--`. Usamos `++` em nosso jogo "Adivinhe o número"no primeiro artigo Um primeiro mergulho no JavaScript, quando adicionamos 1 ao nosso contagemPalpites para saber quantas adivinhações o usuário deixou após cada turno.

```
1
2         contagemPalpites++;
3
4
```

Vamos tentar brincar com eles no seu console. Para começar, note que você não pode aplicá-las diretamente a um número, o que pode parecer estranho, mas estamos atribuindo a variável um novo valor atualizado, não operando no próprio valor. O seguinte retornará um erro:

```
1
2          3++;
3
4
```

Então, você só pode incrementar uma variável existente. Tente isto:

```
1
2          var num1 = 4;
3          num1++;
4
5
```

Ok, segunda coisa estranha! Quando você fizer isso, verá um valor 4 retornado - isso ocorre porque o navegador retorna o valor atual e, em seguida, incrementa a variável. Você pode ver que ele foi incrementado se você retornar o valor da variável novamente:

```
1
2          num1;
3
4
```

Acontece a mesma coisa com `--`: tente o seguinte

```
1
2          var num2 = 6;
3          num2--;
```

```
4           num2;
5
6
```

Nota: Você pode fazer o navegador fazer o contrário - incrementar/-decrementar a variável e depois retornar o valor, colocando o operador no início da variável ao invés do final. Tente os exemplos acima novamente, mas desta vez use `++num1` e `--num2`.

Operadores de atribuição

Operadores de atribuição são os que atribuem um valor à uma variável. Nós já usamos o básico, `=`, muitas vezes, simplesmente atribuindo à variável do lado ersquerdo o valor indicado do lado direito do operador:

```
1
2           var x = 3; // x cont m o valor 3
3           var y = 4; // y cont m o valor 4
4           x = y; // x agora cont m o mesmo valor de y, 4
5
6
```

Mas existem alguns tipos mais complexos, que fornecem atalhos úteis para manter seu código mais puro e mais eficiente. Os mais comuns estão listados abaixo:

Operador	Nome	Proposito	Exemplo	Atalho para
+=	Atribuição de adição	Adiciona o valor á direita ao valor da variavel á esquerda e, em Seguida, retorna o novo valor da variável	x = 3; x += 4;	x = 3; x = x + 4;
-=	Atribuição de Subtração	Subtrai o valor á direita do valor da variavel á esquerda e, retorna o novo valor da variável	x = 6; x -= 3;	x = 6; x = x - 3;
*=	Atribuição de multiplicação	Multiplica o valor da variável á esquerda pelo valor a direita e, retorna o novo valor da variável	x = 2; x *= 3;	x = 2; x = x * 3;
/=	Atribuição de divisão	Dividde o valor da variável á esquerda pelo valor a direita e, retorna o novo valor da variável	x = 10; x /= 5;	x = 10; x = x / 5;

Tente digitar alguns dos exemplos acima em seu console para ter uma ideia de como eles funcionam. Em cada caso, veja se você pode adivinhar qual é o valor antes de digitar a segunda linha.

Note que você pode muito bem usar outros valores no lado direito de cada expressão, por exemplo:

```
1
2         var x = 3; // x cont m o valor 3
3         var y = 4; // y cont m o valor 4
4         x *= y; // x agora cont m o valor 12
5
6
```

Operadores de comparação

Às vezes, queremos executar testes verdadeiro / falso e, em seguida, agir de acordo, dependendo do resultado desse teste, para fazer isso, usamos **operadores de comparação**.

Operador	Nome	Proposito	Exemplo
===	igualdade estrita	Verifica se o valor da esquerda e o da direta são idênticos entre si.	5 == 2 + 4
!==	Não-igualdade-estrita	Verifica se o valor da esquerda e o da direita não são idênticos entre si.	5 !== 2 + 3
—¡—	Menor que	Verifica se o valor da esquerda é menor que o valor da direita.	10 < 6
>	Maior que	Verifica se o valor da esquerda é maior que o valor da direita.	10 > 20
<=	Menor ou igual que	Verifica se o valor da esquerda é menor que ou igual o valor da direita.	3 <= 2
>=	Maior ou igual que	Verifica se o valor da esquerda é maior que ou igual o valor da direita.	5 >= 4

Nota: Talvez você já tenha visto alguém usando == e != afim de testar igualdade ou desigualdade em JavaScript. Estes são operadores válidos em JavaScript, mas que diferem de ===/!==. As versões anteriores testam se os

valores são os mesmos, mas não se os tipos de dados dos valores são os mesmos. As últimas versões estritas testam a igualdade de ambos os valores e seus tipos de dados. Elas tendem a resultar em menos erros, por isso recomendamos que você as use.

Se você tentar inserir alguns desses valores em um console, verá que todos eles retornam `true`/`false` — aqueles booleanos que mencionamos no último artigo. Eles são muito úteis, pois nos permitem tomar decisões em nosso código, e eles são usados toda vez que queremos fazer uma escolha de algum tipo. Por exemplo, booleanos podem ser usados para:

- Exibir o text label em um botão, dependendo se um recurso está ativado ou desativado
- Exibir uma mensagem de 'game over' se um jogo ja acabou ou uma mensagem de vitória se o jogo foi vencido
- Exibir uma saudação sazonal correta, dependendo da época de festas
- Aumentar ou diminuir o zoom do mapa, dependendo do nível de zoom selecionado