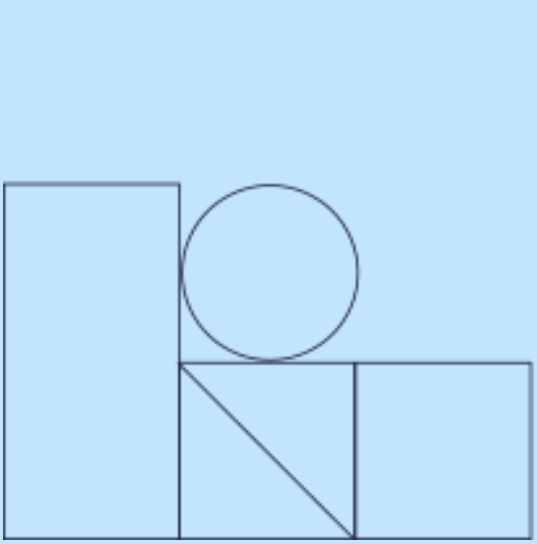
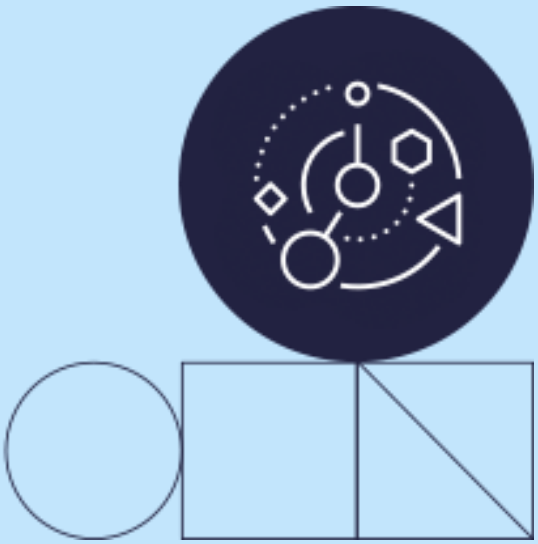
IBM SkillsBuild

IBM

Noções básicas de programação

Definição de variável

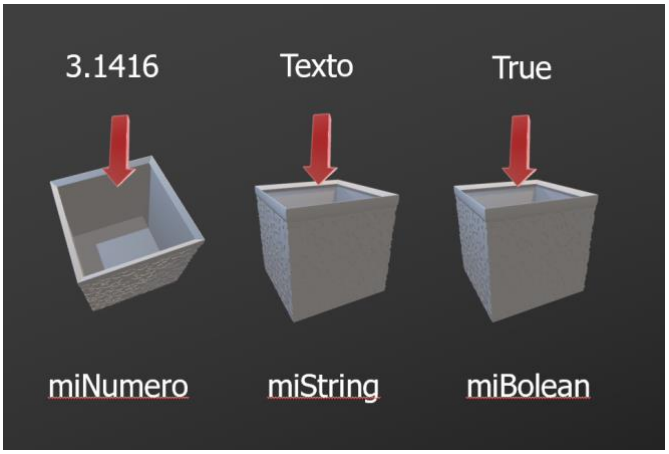




Índice

Introdução

Uma variável é um espaço de memória onde um valor será armazenado, que pode mudar durante todo o programa. Algo como uma "caixa" onde armazenamos dados:



Regras para definir uma variável

Estas regras são comuns à maioria das linguagens de programação:

- Deve começar com uma letra, o símbolo **$**, ou o símbolo _.
- Só pode conter letras, números, o símbolo **$**, ou o símbolo _.
- As variáveis não podem conter espaços em branco.
- Você pode usar sotaques, mas não é recomendado. Também não é recomendado o uso do **"ñ"**.
- Alguns idiomas são sensíveis a maiúsculas e minúsculas. Estes são conhecidos como idiomas sensíveis a casos.
- Como regra geral, elas são em minúsculas.
- Elas não podem ser palavras reservadas.

Tipos de dados

Como vimos na lição anterior, os seguintes tipos de dados primitivos são possuídos pela maioria das linguagens de programação de alto nível.

Vamos nos concentrar na linguagem Javascript, que é o assunto deste curso.

Analisaremos os tipos de dados em JavaScript em detalhes mais tarde, mas isto é apenas uma rápida visão geral.

O Javascript nos fornece uma série de dados primitivos com os quais podemos trabalhar, como por exemplo:

- **Number**: Numérico, seja inteiro ou ponto flutuante (3,4 , 2,5, etc.).
  Nesta linguagem, ao declarar variáveis, não é feita distinção entre números inteiros e decimais. O Javascript está encarregado de decidir se são de um ou de outro tipo, dependendo dos dados que introduzimos em nossa variável.
- **String**: Cordas de caracteres, ou seja, texto. Sejam letras, palavras ou frases.
- **Boolean**: Boleanos, ou seja **true** ou **false**.

Há momentos em que pode ser necessário conhecer o tipo de uma variável em um determinado momento. Para este fim, o JavaScript fornece um operador que, quando aplicado a uma determinada variável, pode obter o tipo de dados que está atualmente atribuído à variável. Este operador é *typeof()*:

```
typeof(nombreVariable);
```

Se uma variável tiver sido declarada mas não rubricada, ou seja, não lhe demos nenhum valor, seu valor é *undefined*. Seu valor é desconhecido. Não é que seja zero, é desconhecido.

Tipos complexos

Elas são criadas pelo usuário a partir de estruturas lingüísticas e/ou agrupamentos de elementos do tipo simples. Assim, é possível encontrar o tipo de dados da função (*function*), que permite atribuir uma função a uma variável (as funções são discutidas posteriormente), e o tipo de dados do objeto, que engloba um conjunto de tipos de dados possíveis (matrizes ou tabelas, por exemplo, que também são discutidas posteriormente, juntamente com o próprio tipo *Object*, que se refere a um objeto genérico).

Declaração variável

As palavras-chave JavaScript são palavras reservadas. As palavras reservadas não podem ser usadas como nomes de variáveis.

As variáveis não declaradas são sempre globais e não existem até que o código que as atribui seja executado. Uma variável não declarada é configurável (por exemplo, ela pode ser apagada).

Recomenda-se sempre declarar as variáveis a serem utilizadas, sejam elas locais ou globais. Quando uma variável não é declarada, o JS a cria no Window Object. Pode até mesmo reatribuir o valor da variável sem pedir permissão, causando um comportamento inesperado.

A primeira coisa que devemos fazer para usar uma variável é "defini-la", (criá-la):

```
var numero;
```

Nós já criamos nossa variável, mas não lhe demos nenhum valor.

Para poder trabalhar com ele, temos que "rubricá-lo", ou seja, dar-lhe um valor inicial.

Para dar-lhe um valor:

```
numero = 5;
```

Uma vez que tenhamos a variável definida, não precisamos usar a palavra reservada *var* para lhe dar um valor.

Também podemos **definir** e **rubricar** uma variável em uma única etapa:

```
var numero = 5;
```

Variáveis múltiplas podem ser definidas ou inicializadas em uma única linha, mas isto é desencorajado:

```
var nombre, apellido, DNI;

var nombre="Angel", apellido="García", DNI="12345678A";
```

Uma variável declarada mas não rubricada deve ser *undefined*.

Constantes

Uma constante é definida como um espaço na memória do computador cujo valor **não** pode mudar durante a execução do programa. Em outras palavras, uma "variável" cujo valor nunca varia.

Por convenção, os nomes das constantes são capitalizados.

Sintaxe:

```
const NOMBRE = "Angel";
```

Uma constante não pode compartilhar seu nome com uma função ou variável no mesmo escopo.

Scope, escopo

O escopo (scope) de uma variável é o contexto (a área do programa) no qual a variável é visível/acessível (declarada), e portanto pode ser usada. O escopo determina a acessibilidade de nosso código, ou seja, de onde podemos acessar nossas variáveis.

Temos dois escopos para as variáveis:

- **Global**: Podemos acessá-los de qualquer lugar em nosso código.
- **Local**: Podemos acessá-los apenas dentro do escopo em que foram definidos. Normalmente funciona. Eles serão acessíveis de dentro da própria função ou de funções aninhadas em níveis mais altos do que a função anterior. Ou seja, variáveis locais podem ser vistas de dentro para fora, mas nunca de fora para dentro da função.

As variáveis locais são definidas dentro de uma função e só podem ser acessadas dentro da própria função. Uma variável local é definida através do prefixo do nome da variável com a palavra-chave *let*.

Se tentarmos acessar a variável fora da função na qual ela foi declarada, ocorre um erro.

O exemplo a seguir ilustra o comportamento dos escopos:

```
function creaMensaje() {
    var mensaje = "Mensaje de prueba";
  }
  creaMensaje();
  alert(mensaje);
```

O exemplo acima define primeiro uma função chamada *createMessage* que cria uma variável chamada mensagem. A função é então executada chamando *createMessage()*; e então o valor de uma variável chamada mensagem é exibido pela função *alert()*.

Entretanto, ao executar o código acima, nenhuma mensagem é exibida na tela. A razão é que a mensagem variável foi definida dentro da função *creaMensaje()* e, portanto, é uma variável local que só é definida dentro da função.

Qualquer instrução dentro da função pode fazer uso desta variável, mas todas as instruções em outras funções ou fora de qualquer função não terão a mensagem variável definida. Assim, para exibir a mensagem no código acima, a função alert() deve ser chamada de dentro da função creaMensaje():

```
function creaMensaje() {
    var mensaje = "Mensaje de prueba";
    alert(mensaje);
  }
  creaMensaje();
```

Além das variáveis locais, há também o conceito de uma **variável global**, que é definida em qualquer parte do programa (mesmo dentro de qualquer função).

```
var mensaje = "Mensaje de prueba";

function muestraMensaje() {
  alert(mensaje);
}
```

O código acima é o inverso do exemplo mostrado acima. Dentro da função *muestraMensaje()* queremos fazer uso de uma variável chamada mensagem que não foi definida dentro da própria função. Entretanto, se o código anterior for executado, a mensagem definida pela mensagem variável é exibida.

A razão é que no código JavaScript acima, a mensagem variável foi definida fora de qualquer função. Tais variáveis tornam-se automaticamente variáveis globais e estão disponíveis em qualquer parte do programa (mesmo dentro de qualquer função).

Desta forma, embora nenhuma variável chamada mensagem tenha sido definida dentro da função, a variável global criada acima permite que a instrução *alert()* dentro da função exiba a mensagem corretamente.

Se uma variável é declarada fora de qualquer função, ela se torna automaticamente uma variável global independentemente de ser definida usando ou não a palavra reservada *var*. Entretanto, as variáveis definidas dentro de uma função podem ser globais ou locais.

Se as variáveis são declaradas dentro de uma função usando *var*, elas são consideradas locais e variáveis que não foram declaradas usando *var* são automaticamente transformadas em variáveis globais.

Portanto, você pode refazer o código do primeiro exemplo para que ele exiba a mensagem corretamente. Para fazer isso, basta definir a variável dentro da função sem a palavra reservada *var*, de modo que ela se torne uma variável global:

```
function creaMensaje() {
    mensaje = "Mensaje de prueba";
  }

  creaMensaje();
  alert(mensaje);
```

O que acontece se uma função define uma variável local com o mesmo nome de uma variável global que já existe? Neste caso, as variáveis locais têm precedência sobre as variáveis globais, mas somente dentro da função:

```
var mensaje = "gana la de fuera";

function muestraMensaje() {
  var mensaje = "gana la de dentro";
  alert(mensaje);
}

alert(mensaje);
muestraMensaje();
alert(mensaje);
```

O código acima exibe as seguintes mensagens na tela:

- gana la de fuera
- gana la de dentro
- gana la de fuera

Dentro da função, a variável local chamada mensagem tem maior prioridade do que a variável global com o mesmo nome, mas somente dentro da função.

O que acontece se uma variável global for definida dentro de uma função com o mesmo nome que outra variável global que já existe? Neste outro caso, a variável global definida dentro da função simplesmente modifica o valor da variável global previamente definida:

```
var mensaje = "gana la de fuera";
function muestraMensaje() {
  mensaje = "gana la de dentro";
  alert(mensaje);
}

alert(mensaje);
muestraMensaje();
alert(mensaje);
```

Neste caso, as mensagens exibidas são:

- gana la de fuera
- gana la de dentro
- gana la de dentro

A recomendação geral é definir como variáveis locais todas as variáveis que são usadas exclusivamente para executar as tarefas atribuídas a cada função. As variáveis globais são usadas para compartilhar variáveis entre funções de uma maneira simples.

Tudo isso será abordado com muito mais detalhes quando discutirmos as variáveis no módulo **Javascript**.