**Arquitetura de Sistemas Computacionais**

Em Javascript, o tipo de uma expressão (number, string, object, ...) é inferido em tempo de execução, diferentemente de algumas outras linguagens como Java, por exemplo.

Para se declarar constantes se utiliza o ‘const’ e para declarar variáveis, se utiliza o ‘var’ e o ‘let’. DAR PREFERÊNCIA PARA O LET visto que o escopo do var pode ser contraintuitivo as vezes (podendo ser global ou a função em que foi declarada)

Exemplo:

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

Quando se utiliza o let, se espera que a variável seja tratada como local, com escopo de bloco

Texto

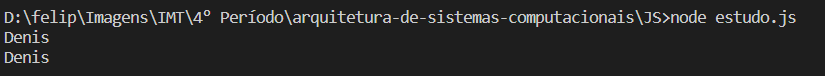
Descrição gerada automaticamente

Já quando utilizamos o ‘var’...

Texto

Descrição gerada automaticamente

Intuitivamente, esperamos que o mesmo ocorra, visto que a variável ‘nome’ com o valor ‘Denis’ foi definida apenas dentro de uma condição if.



Mas não é o que acontece... ele trata ela globalmente. Por isso se da a preferência em utilizar o ‘let’, pela questão da intuitividade.

Utilizando o ‘var’, a variável pode inclusive ser também redeclarada (o que é desnecessário visto que o Javascript permite que você sobrescreva o valor apenas atribuindo à variável direto, sem redeclará-la). Exemplo:

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Resultado:



Já utilizando o ‘let’ ...

Interface gráfica do usuário

Descrição gerada automaticamente

Uma imagem contendo Texto

Descrição gerada automaticamente

**Tipos em Javascript**

Existem possíveis tipos de um valor em Javascript:

- boolean: True or false

- number: int or float

- string: ‘ola mundo’ ; ‘Felipe Matos Silvieri’ ; ‘521.240.288-39’

- null: sem valor (intencionalmente atribuído [para iniciar variáveis, por exemplo])

- undefined: sem valor (provavelmente causado por variável não utilizada ou ‘erro’ do programador)

...Existem também os objetos... (aparecem como resultado do typeof de um vetor, pro exemplo)

Objetos:

* JSON (Javascript object notation)
* Array ( [1,2,3,4,5,’tso’,12.7]
* Classes Wrapper (String, Number, Boolean)
* Date
* Math
* Funções ( () => { console.log(‘teste’) } )

Função setTimeout(param1,param2): Param1 = função a ser chamada depois de ‘Param2’ milissegundos. Se não for passado nenhum valor de tempo, a função executa instantaneamente

**COERÇÃO**

*A coerção é basicamente a troca de tipo de um primitivo (number, string, etc..) durante o script após a sua declaração.*

Coerção implícita: quando o programador não ‘avisa’ o js que duas variáveis, por exemplo possuem tipos diferentes na hora de ‘soma-las’. Exemplo:

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Coerção explícita: quando o programador avisa o js qual o tipo da variável que ele vai utilizar. Exemplo:

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

**Comparação**

Existem dois operadores de comparação. São eles: ‘==’ e ‘===’. Preferencialmente, utilizaremos o ‘===’ visto que ele não utiliza coerções contraintuitivas. No entanto, o operador ‘==’ tem um comportamento na prática que não é muito interessante na maioria das vezes. Exemplos:

Texto

Descrição gerada automaticamente

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

No entanto, se utilizarmos o comparador ‘===’:

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

Outros Exemplos:

Texto

Descrição gerada automaticamente

O comportamento da ultima linha se da pelo fato de que uma comparação estabelecida entre dois arrays vazios não resultará em True, visto que cada array tem uma posição de endereçamento na memória do computador, e isso acaba causando esse comportamento. O Javascript identifica os dois arrays como objetos distintos.

Boa parte dos sistemas modelados em javascript ainda adotam o operador ‘==’ por motivos de retrocompatibilidade, mas não é recomendável a sua utilização.

**VETORES**

Os vetores são declarados assim: (let a = []).

Para se adicionar valores ao vetor, é possível utilizar simplesmente a notação de colocar o valor e o endereço desejado no vetor. Exemplo:

Texto

Descrição gerada automaticamente

Como é possível ver acima, consegui adicionar um valor na posição ‘5’ do vetor e na posição ‘0’ e ‘1’, fazendo assim com que os valores das posições 2,3,4 fiquem vazios.

O vetor ilustrado acima possui um tamanho de 6, o que é possível confirmar através do comando a.lenght:





O vetor não precisa ser declarado através de uma variável ou constante:

Texto

Descrição gerada automaticamente com confiança média



Percorrer:

Texto

Descrição gerada automaticamente

**Outras Aplicações:**

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Reduce() = reduz um vetor a um numero, tendo um acumulador e um numero (valor na posição de iteração do vetor)

Map() = itera sobre os elementos do vetor, retornando um array com o mesmo numero de elementos que o original, que podem ser alterados a gosto do desenvolvedor.

**Funções**

Texto

Descrição gerada automaticamente



Existem algumas formas de se definir uma função em Javascript:

**Anônimamente:**

Texto

Descrição gerada automaticamente

Forma

Descrição gerada automaticamente com confiança média

\*repare que no caso acima, ‘triplo’ pode ter um valor de parâmetro padrão\*

**Arrow Function:**

Interface gráfica do usuário, Texto, Aplicativo, chat ou mensagem de texto

Descrição gerada automaticamente

Nas arrow functions pode-se omitir as chaves {}, mas para isso não utilize o ‘return’!

Texto

Descrição gerada automaticamente com confiança média

Arrow Function 🡪 const a = () => {console.log(‘tso’)} (não precisa das chaves)

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

FICA ESPERTO:

Texto

Descrição gerada automaticamente 

Texto

Descrição gerada automaticamente 

Questão da prova:

Texto

Descrição gerada automaticamente

Uma imagem contendo Interface gráfica do usuário

Descrição gerada automaticamente

JSON

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Fica esperto ta? Interface gráfica do usuário, Texto, Aplicativo, chat ou mensagem de texto

Descrição gerada automaticamente

3 é valido pq const nome + const pessoa da um bagulho valido...

**setTimeout**

Interface gráfica do usuário, Texto, Aplicativo, chat ou mensagem de texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente com confiança média

\*\*\* Mesmo não admitindo nenhum valor para o parametro de tempo do setTimeout, ele coloca a função para ser executada por ultimo

Texto, Aplicativo, chat ou mensagem de texto

Descrição gerada automaticamente

Interface gráfica do usuário

Descrição gerada automaticamente com confiança média

**Promises**

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

Uma promise pode se encontrar em 3 estados:

Pending, fullfilled e rejected

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

Tela de jogo de vídeo game

Descrição gerada automaticamente com confiança média