

Módulo 6

Fundamentos do JavaScript



Conteúdo do módulo

Sintaxe

Variáveis e constantes

Tipos de dados

Laços

Condicionais

Funções





Módulo 6 / Aula 1

Sintaxe

Conteúdo a aula

Sintaxe do JavaScript

Comentários



Sintaxe do JavaScript



É case sensitive: duas palavras escritas com o mesmo texto mas com a caixa diferente, são diferentes, exemplo: nomeDoMeio e nomedomeio.

algumas palavras que não podemos usar na hora <u>de nomear variáveis ou criar funções, pois são </u> reservados pelo interpretador do JavaScript. Existência de palavras reservadas: existem

Sintaxe do JavaScript

em algumas linguagens de programação o uso de Uso de ponto e vírgula ao final de uma sentença: código é obrigatório, já foi assim no JavaScript, atualmente o uso do ponto e vírgula é opcional. ponto em vírgula ao final de uma sentença de



MÓDULO 6 – Aula 1 - Sintaxe

Comentários

comentários por linha e comentários em bloco, que podem conter várias linhas. No JavaScript podemos escrever

Comentários em linha: para comentar uma única linha usamos o sinal //

// sou um comentário

bloco, começamos com: /* e determinamos em */ Comentários em bloco: para comentários em

um comentário Também sou



Módulo 6 / Aula 2

ESCUER ESCUER Británica de Artes creativas y tecnología

Variáveis e constantes

Conteúdo a aula

· Variáveis

Constantes

Nomeando variáveis e constantes



Variáveis



*let pode não funcionar em navegadores mais antigos, variável no JavaScript podemos usar as <u>armazenam um valor, para criar uma</u> Variáveis são como caixas que palavras-reservada var e let. esse recurso foi introduzido em 2015

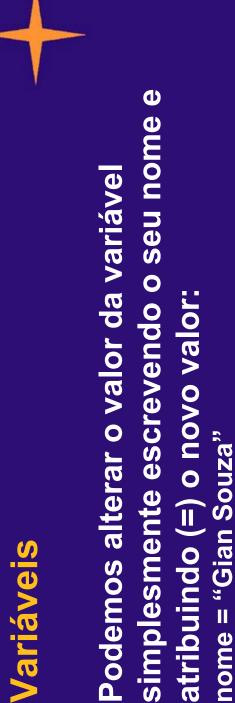
Nossa primeira variável:

et nome = "Gian"

Composição de uma variável:

let ou var NOME_DA_VARIAVEL = VALOR

Variáveis



quer dizer, se quisermos sobrescrever o O JavaScript não possui tipagem, isso exemplo anterior, com:

nome = 32

Constantes



é que uma vez inicializada não poderá ter Constantes assim como as variáveis são além do uso da palavra reservada const, containers de valor, a grande diferença o seu valor trocado.

const cpf = 12345678910 cpf = 123 // irá gerar um erro

Nomeando variáveis e constantes



Podemos escrever o nome das variáveis seguindo alguns padrões de

nomenclatura:

Underscore: var primeiro nome = "Gian"

V PascalCase: var PrimeiroNome = "Gian" CamelCase: var primeiroNome = "Gian"

especiais ou usar traço hífen no meio das Não podemos começar uma variável ou constante com um número, caracteres palavras



Módulo 6 / Aula 3

Tipos de dados dados



Conteúdo a aula

Sobre o tipo de dados no JavaScript



Literais

Lógicos

Númericos

Objetos

Sobre o tipo de dados no JavaScript



na hora de declaramos uma variável por fortemente tipagem, isso quer dizer que exemplo, não preciso informar que será numérica em determinado momento do alguns problemas, como por exemplo, programa pode se tornar um número. um texto ou número, mas isso gera uma variável que originalmente era O JavaScript não é uma linguagem

Sobre o tipo de dados no JavaScript



Os tipos de dados que temos disponíveis no JavaScript são:

String: tipo literal, usado para textos e

caracteres;

Number: tipo numérico, usado tanto para

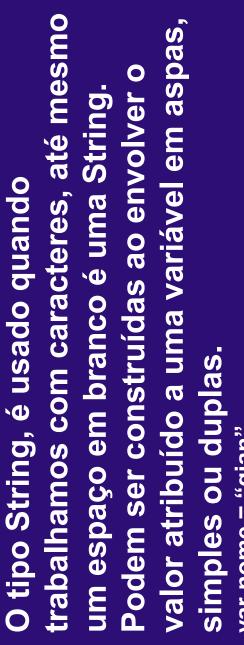
números inteiros e reais;

Boolean: tipo lógico, que indica se

Object: tipo mais básico do JavaScript, todos determinada afirmação é verdadeira ou falsa;

os tipos nasceram a partir do Object.

Literais - String



var nome = "gian" Nome = 'gian'

Literais - String



Podemos usar alguns recursos da String apenas um pedaço ou procurar por um para contar o seu tamanho, selecionar pedaço de texto ou uma letra.

Consideramos:

var nome = "JavaScript"

Contando o tamanho de uma String nome.length // 10

Tornando todo o texto em caixa alta nome.toLocaleUpperCase() // JAVASCRIPT

MÓDULO 6 - Aula 3 - Tipos de dados

Literais - String

Procurando por uma letra ou palavra:

nome.search('v') // 2

O 2 indica a posição da letra V em JavaScript, naturalmente seria

3, mas a posição inicial é 0 e não 1

nome.slice(0, 2) // Ja

Retorna um pedaço da String, 0 sendo a posição inicial e 2 o número de caracteres que queremos selecionar.

Unindo Strings

Podemos unir Strings + Strings ou Strings + Números, com

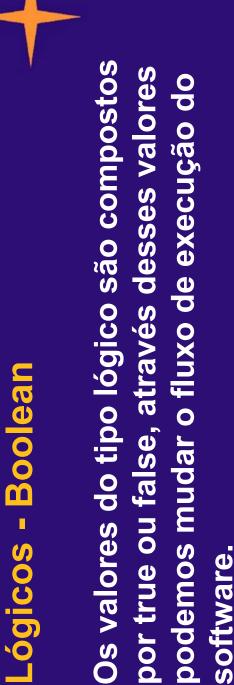
o operador +

"Olá " + nome // Olá JavaScript nome + " é " + 10 + "!" // JavaScript é 10!

Chamamos essa união de concatenação

+

Lógicos - Boolean



var podeEntrar = foiConvidado var foiConvidado = true

pode entrar num local apenas se for convidado, a primeira variável diz se a pessoa foi convidada, e No exemplo acima temos uma situação onde se <u>como o valor da segunda é condicionada a ela, </u> podemos simples repassar seu valor, logo:

foiConvidado = false ENTÃO podeEntrar = false foiConvidado = true ENTÃO podeEntrar = true

Numéricos - Number



existe uma separação interna entre o tipo Para criar uma variável do tipo numérico em JavaScript, basta atribuir o número considerados ambos do tipo number, integer (inteiro) e float (fracionado): ao valor. Apesar de tanto o número inteiro como o fracionado serem var altura = 1.72 // Number mas Float var idade = 32 // Number mas Integer

Numéricos - Number



Com os números podemos fazer cálculos

aritméticos:

var ano = 2022 var idade = 32

idade * 2 = 64

idade / 2 = 16

Podemos usar operações para atribuir o valor às variáveis. var anoDeNascimento = ano - idade

Numéricos - Number



Assim como nas Strings os elementos do tipo Number possuem alguns recursos especiais, como:

Verificar se um número é Integer. Number.isInteger(10) // true Number.isInteger(10.5) // false

parseFloat("10,5632") // 10 – não devemos usar vírgula Converter uma String para Float: parseFloat("10.5632") // 10.5632

parseInt(10.43) // 10 - o restante do número é descartado Converter um Float para Integer:

Objetos - Object



Um ponto importante é que o JavaScript trata os vetores como objetos.

conjunto é representado pelos símbolos de colchetes ([.....]), e seus itens são Vetores são conjuntos de dados, o separados por vírgula.

Também são chamados de arrays.

szuo2 nsiƏ

Objetos – Array

Exemplo de um array de Strings var nomes = ["gian", "carlos", "maria"]

itens do mesmo tipo em um conjunto, por JavaScript não nos obriga a adicionar Por possuir uma fraca tipagem o exemplo:

nomes = ["gian", "carlos", "maria", "joão", 15, 32]

indice, onde o primeiro item possui o índice 0. Para acessar um item do array usamos seu

nomes[0] // gian

nomes[4] // 15



Módulo 6 / Aula 4

Conteúdo a aula

Sobre laços

Laço While

Laço do while

Laço for in Laço for

Sian Souza

Sobre laços

iteração, podemos ter um conjunto com itens, também chamamos de loop ou queremos percorrer um conjunto de Laço é o nome que damos quando os nomes dos ganhadores de uma promoção:

var ganhadores = ["Gabriela", "João", "Luana"]

acessar o índice de item no conjunto Sem o uso de um laço, teríamos que ganhadores[0] // Gabriela

ganhadores[1] // João

ganhadores[2] // Luana



Laço for

O tipo de laço mais comum é o for, sua estrutura é:

```
for (var i = 0; i < qtd de itens; i++) {
// ...
}
```

Onde:

var i = criamos uma variável que irá contar as vezes em que "entramos" no laço

geralmente é o tamanho do conjunto, que pode ser obtido qtd de itens = quantas vezes queremos entrar no laço, através de *ganhadores.length*, por exemplo. ++ = Informamos que a cada vez que o laço for acionado o

valor de i deve ser incrementado + 1.

= Será executada apenas quando i for menor que a qtd de

Laço for

Exemplo prático:

var ganhadores = ["Gabriela", "João", "Luana"]

console.log("Parabéns: " + ganhadores[i]); for (var i = 0; i < ganhadores.length; i++) {

Parabéns: Gabriela

Parabéns: Luana

entre colchetes que ele terá o valor atualizado do informar o índice do elemento, basta informar o i Desta forma não precisamos nos preocupar em item atual.

Parabéns: João

Laço for in



Exemplo considerando o conjunto:

ganhadores = ["Gabriela", "João", "Luana"]

```
console.log("Parabéns: " + ganhadores[ganhador]);
                                        for (ganhador in ganhadores) {
var saudacoes = 0;
```

Laço while

Outra forma de iterarmos um conjunto de dados é através do laço while, sua estrutura é:

while (condição verdadeira) {

while (condição vel

verdadeira o código dentro do bloco será Enquanto (while) uma condição for executado.

Laço while

Exemplo considerando o conjunto:

ganhadores = ["Gabriela", "João", "Luana"]

```
console.log("Parabéns :" + ganhadores[saudacoes]);
                                      while (saudacoes < ganhadores.length) {
var saudacoes = 0;
                                                                                                                             saudacoes++;
```

quantidade de itens o bloco será executado, ao final de <u>que no final da terceira saudação saudações = 3 e isso</u> cada o número de saudações será incrementado para <u>Enquanto o número de saudações for menor que a </u> já não satisfaz mais a condição do laço while.

Laço do while

O laço do while é bem parecido com o laço while, apenas inverte o bloco de execução com a condição:

do { // ...

} while (condição verdadeira);

código enquanto a condição for verdadeira. Resumidamente, execute esse bloco de



Módulo 6 / Aula 5

Condicionais

Conteúdo a aula

Sobre condicionais

Else

If else

Ternário

Sian Souza

Sobre condicionais

Em conjunto com os verificadores temos executar diferentes fluxos de instruções. Com o uso das condicionais podemos os operadores lógicos e os comparadores.



MÓDULO 6 – Aula 5 – Condicionais

Comparadores

< Menor que

5 < 4 // false

> Maior que 5 > 5 // false

<= Menor ou igual que

2 <= 3 // true

>= Maior ou igual que 5 >= 5 // true

== Igual a

5 == "5" // true

=== |gual a // compara além do conteúdo o tipo do dado

5 === "5" // false

!= Diferente

de !== Diferente de

5 !== "5" // true

! Negação

5 != "5" // false

!true // false, estamos negando o true, logo false

Operadores lógicos

Com os operadores lógicos conseguimos construir uma cadeia de condicionais.

No JavaScript possuímos os operadores && (E), || (Ou) e! (Negação).

Operadores lógicos

Ш ॐ ॐ

Com esse operador todos os termos devem ser verdadeiros para a condição ser verdadeira.

true && true // true

5 > 10 && true // false, cinco não é maior que 10

Com esse operador apenas um dos termos precisam ser verdadeiros para a condição ser verdadeira. 5 > 10 || true // true, o termo da direita é verdadeiro true || false // true

! Negação

Com esse operador negamos um termo.

!false // true, a negação de um termo falso é ele ser verdadeiro itrue // false

szuo2 nsi

condição e se ela for verdadeira o código Com o uso do IF (se) verificamos uma dentro do IF será executado:

if (condição) {

MÓDULO 6 – Aula 5 – Condicionais

Else

criar um fluxo alternativo para quando a Com o uso do Else (senão) podemos condição do IF não for atendida:

```
if (condição) {
// ...
} else {
// ...
```

Logo, se a condição for false, o código executado será o dentro do bloco else.

MÓDULO 6 – Aula 5 – Condicionais

vamos ver uma instrução onde só se pode entrar para tratar nosso fluxo de execução, quando a Com o else if podemos criar outras condições condição inicial não é satisfeita. Para ilustrar, se tiver mais ou 18 anos ou menor com acompanhante.

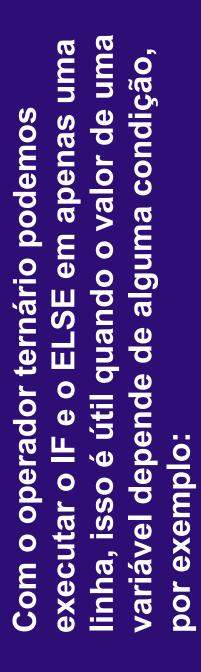
```
} else if (estaAcompanhado) {
                       podeEntrar = true
                                                                   podeEntrar = true
if (idade >= 18) {
                                                                                          } else {
```

Else if

Gian Souza

podeEntrar = false

Operador ternário



var saudacao = eFeriado ? "Estamos fechados hoje" : "Seja var eFeriado = true bem-vindo";

Com o ternário o IF é executado após o sinal? E o else após o :





Módulo 6 / Aula 6

Gian Souza

Conteúdo a aula

- Sobre funções
- Criando uma função
- **Parâmetros**
- Retornando dados

Sobre funções

Uma função, também pode ser chamada forma de separamos partes específicas do código e que podemos reutiliza-las. Podem receber parâmetros, também de método ou procedimento, é uma chamados de argumentos e podem retornar algum valor ou não.



Criando uma função

variável ou usando a palavra reservada função no JavaScript, através de uma Temos dois caminhos para criar uma function.

```
var nome = "JavaScript"
function dizOi() {
    console.log("Oi, " + nome);
}
```

```
var dizOi = function() {
   console.log("Oi, " + nome);
```

~

Gian Souza

Criando uma função

Podemos dizer que estamos chamando Para executar a função escrevemos o nome dela seguido de (), dizOi(). ou invocando um função.



MÓDULO 6 – Aula 6 – Funções

Parâmetros

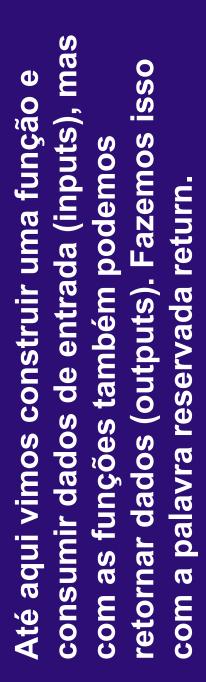


Com o uso de parâmetros podemos tornar nossa função independente de recursos externos.

```
console.log("Oi, " + nome + " de " + cidadeNatal);
function dizOi(nome, cidadeNatal) {
```

Gian Souza

Retornando dados



Um ponto importante é que todo o código escrito após o return não será executado.



MÓDULO 6 – Aula 6 – Funções

Retornando dados

Exemplo com return:

function numeroAoQuadrado(numero) { return numero * numero;

~

Podemos usar o retorno para atribuir o valor a uma variável, por exemplo:

var num1 = 4;

var num2 = numeroAoQuadrado(num1); // 16

