Curso de Java Script – Roger Mello

# Aula 001

40:01

Aula inaugural

Visão do Roger sobre como realizar um curso, suas ideias e regras.

Comece e não pare, devagar ou rápido, não pare, não pule as aulas e não faça diferente do professor perdendo o foco

Entender é diferente de executar, pratique, mesmo que vc tenha entendido tudo

Nove marcos para chegar na fluência

Alicerce

Marco 1 – Assistir aula inaugural

Marco 2 – Realizar todas as atividades – Postar resumo/fazer perguntas das aulas no repositório github = roger-melo-treinamentos/curso-de-js-roger-melo

Marco 3 – Como realizar as atividades: 0 – Faça os exercícios – 2 – Faça os exercícios corrigidos junto com o professor até que vc entenda os procedimentos – Implemente novamente, sozinho, do jeito do professor – 3 – Se tiver dúvidas, procure tirá-las usando suas anotações, e tente fazer sozinho novamente do zero.

Fazer uma revisão raio-x antes de cada implementar cada aplicação – Tem link explicando

Validação

Marco 4 – Análise das aplicações pelo professor

Marco 5 – Montar um portifólio com as 3 primeiras aplicações, no github e website. Pelo jeito o Roger não sabe da existência do github pages por citou que as pessoas podem não saber mexer com o github e baixar os seus projetos

Marco 6 – Se candidatar as vagas – Quando o medo é maior que o desejo vc paralisa, senão vc age. A cura do medo é ação

Evolução

Marco 7 – Implementar os feedbacks dos recrutadores

Marco 8 – Ser contratado

Comemoração

Marco 9 – Postar sua comemoração no repositório do treinamento

Plantão da fluência. atendimento individualizado de questões não técnicas

# Aula 002

01:05

Objetivos do treinamento

# Aula 003

03:01

Definindo JS

# Aula 004

07:10

Configurando o ambiente – VSCode – A extensão Live Server

<html lang=”pt-br”>

Instalar o f.flux inibe a luz azul do Monitor para descansar a visão

VSCode ctrl + shift + p -> Atalho para todas as funções

VSCode ctrl + b -> Fecha/Abre a aba de arquivos

Emmet ! <tab> -> Estrutura html

# Aula 005

03:36

alert(‘Mensagem’)

Mensagem popup no browser com botão OK

<script src=”App.js”></script>

# Aula 006

02:28

Chrome f12 -> Abrir aba Devtools

Configurar o Chrome DevTools com temas diferentes

DevTools -> ‘3 pontinhos’ -> Setings -> Aparencia -> Theme

console.log(‘mensagem’) -> exibir mensagens em terminais ou no console do devtools no chrome

Javascript é executado de forma sequencial de cima para baixo

# Aula 007

09:11

Constante, variáveis e comentários

let nomeVar

const nomeVar2

var nomeVar3 -> var não é mais usado

console.log(nomeVar, ‘Texto’, nomeVar2, nomeVar3)

let e const vieram no JS5

regras de nomeação de variáveis - camelCase

// Comentário de 1 linha

/\*

Cometários de várias linhas

\*/

VSCode ctrl + ; Coloca/retira comentários de linha

# Aula 008

04:44

Primeiros exercícios

No github – Em um olhinho escrito watch vc pode clicar nele e selecionar uwatch para ser avisado de alteração naquele repositório

Instalação do git

Instalação do terminal Hyper feito em html css js

hyper.is

git –version

git version 1.9.1

VSCode shift + ctrl + e -> Só abre a aba arquivos, não fecha

# Aula 009

08:04

Correção dos exercícios da aula 001

# Aula 010

03:21

Overview sobre tipos de dados

Number, String, Boolean, Undefined, Null, Object, Symbol, BigInt

Null -> quando nós queremos deixar explicitamente uma variável sem valor

Undefined -> O JS declara quando uma variável não possui valor e nem foi declarada com null

# Aula 011

12:52

Strings

“um texto normal”

‘um texto normal’

`texto ${variável} texto texto ${variável}`

‘texto’ + “texto2” + ‘texto’

textotexto2texto

Concatenação de strings

“texto”.length -> exibe a quantidade de caracteres

let novo = ‘texto’

console.log(novo[1])

e

JS – Zero based

console.log(“texto”.toUpperCase())

TEXTO

console.log(“TEXTO”.toLowerCase())

texto

Strings possui métodos que agem em cima do objeto original e outros em cópias destas strings

O Professor declarou que as Strings são imutáveis porém depois citou que existe métodos que agem no original e outros que agem na cópia

console.log(“xxx@xxx.com”.indexOf(‘@’));

3

# Aula 012

08:06

Métodos comuns de strings

‘texto’.lastIndexOfA(‘t’)

3

Retorna o índice da última ocorrência do parâmetro na string

const email = 'laracroft@gmail.com'

console.log(email.slice(0, 9))

laracroft

Retorna a string entre o primeiro parâmetro até o segundo parâmetro

Pode ser feito desta forma tumém

const email = 'laracroft@gmail.com'

console.log(email.slice(0, email.indexOf('@')))

laracroft

Não usando o Segundo parâmetro é retornado a string entre o valor do parâmetro usado até o fim da string

const email = 'laracroft@gmail.com'

console.log(email.slice(9))

@gmail.com

string.replace(‘string’,’string’)

const email = 'laracroft@gmail.com'

console.log(email,'\n',email.replace('lara', 'sarada'))

laracroft@gmail.com

saradacroft@gmail.com

Troca a primeira ocorrência da string do primeiro parâmetro pela do segundo parâmetro

# Aula 013

11:34

Numbers

Operadores aritméticos

/ -> Divisão

% -> Modulo

Área de um círculo

const radius = 10

const pi = 3.14

const area= pi \* radius \*\* 2

‘pi vezes (\*) radius Elevado (\*\*) ao quadrado (2)’

Ordem de precendência

1 -> Parêntese

2 -> expoentes (exponenciação) ou raízes

3 -> Multiplicação e divisão

4 -> Adição e subtração

Ou seja

radius \*\* 2 -> Foi resolvido primeiro

pi \* ‘resultado da expressão acima’

Incremento e decremento

let temp = 10;

console.log(temp++);

console.log(temp)

console.log('--------------')

temp = 10;

console.log(++temp);

console.log(temp)

console.log('--------------')

Resultado:

10

11

--------------

11

11

--------------

-------------------------------------------

Addition/Subtraction/mult/div assignment

destination assignment

let temp = 10

temp += 10

cosole.log(temp)

20

= temp = temp + 10

------------------------------------

NaN

# Aula 014

09:12

Template strings ou templates literals

`${variavel} texto texto `${variavel} texto `${variavel}`

\’ -> Escapando aspa simples

|” -> Escapando aspando enters duplas

\n ->Escapando enter

\t -> Escapando tabulação

Crase ou Back strict?

Templates html

const html =

`

<h2>${posTitle}$</h2>

<p>Autor: ${posAutor}$</p>

<span>Este post contém: ${postComents} comentarios</span>

`

Ai é só usar no html com .textHtml = html

# Aula 015

1,29

Exercícios 2

# Aula 016

27:56

Correção dos exercícios 2

EMMET script[src=”./app.js”] <tab>

# Aula 017

10:10

Arrays (objetos array)

let heroes =['batman', 'catwoman', 'iron man']

console.log(heroes)

[ 'batman', 'catwoman', 'iron man' ]

heroes[3] = 'Spiderman'

console.log(heroes)

[ 'batman', 'catwoman', 'iron man', 'Spiderman' ]

heroes[0] = 'BobSponja'

console.log(heroes)

[ 'BobSponja', 'catwoman', 'iron man', 'Spiderman' ]

É possível armazenar dados de tipos diferentes em arrays, porém o array foi criado para armazenar dados que possuem uma relação entre si, e o professor acredita que isso significa dados do mesmo tipo

let arrayRandom = [100, 'catwoman', 'iron man',13 ]

Propriedades e métodos

heroes.length -> exibe a quantidade de itens

console.log(heroes.join())

BobSponja,catwoman,iron man,Spiderman

console.log(heroes.join('---'))

BobSponja---catwoman---iron man---Spiderman

console.log(heroes.indexOf('catwoman'))

1

console.log(heroes.concat(['hombre raspadinha', 'melão das trevas']))

[

'BobSponja',

'catwoman',

'iron man',

'Spiderman',

'hombre raspadinha',

'melão das trevas'

]

heroes.concat(...) – return um novo array

console.log(heroes.push('he-man', 'Descarga fatal')) -> Não foi necessário colocar os itens entre colchetes

6

Porque retornou 6? porque o push altera o array original e return a nova quantidade de elementos. Se colocar novamente:

console.log(heroes)

[

'BobSponja',

'catwoman',

'iron man',

'Spiderman',

'he-man',

'Descarga fatal'

]

Observe que apareceu os novos itens incluídos com o push mas não os itens incluídos com o concat porque o concat return um novo array e para visualizá-los teríamos que guardar o return do concat em uma variável

console.log(heroes.pop())

Descarga fatal

heroes.pop() exclui último elemento do array original e return o elemento excluído

console.log(heroes)

[ 'BobSponja', 'catwoman', 'iron man', 'Spiderman', 'he-man' ]

O 'Descarga fatal' não existe mais

# Aula 018

04:23

null -> atribuído intencionalmente em variáveis que não possuir conteúdo – O null convertido em string return 0 (zero) por isso null + 10 é igual a 10

undefined -> quando o js não localiza o conteúdo da variável que deveria ter conteúdo. Undefined + 10 é igual a NaN

# Aula 019

01:25

Exercícios 3

# Aula 020

19:50

Correção exercícios 3

Sempre verificar o retorno de cada expressão e funções/métodos antes de usar, nem que seja apenas para vc saber e assim trabalhar de forma correta. Porém, em muitos casos esta verificação deve ocorrer também em sua implementação

# Aula 021

11:40

Booleans true/false

const email = 'brucewayne@xxx.com'

const include = email.includes('@') -> Retorna true/false se o parâmetro existe dentro da string email

console.log(include)

true

O método ‘string’.includes(‘caracter procurado’) também pode ser usado em arrays, no caso o ‘caracter procurado’ deve corresponder a um valor do array, deixando claro que o valor deve ser exato ao que se procura e não apenas parte do nome do valor. Exemplo:

const names = ['carlos', 'elaine', 'davi']

let arrayInclude = names.includes('elaine')

console.log(arrayInclude)

true

arrayInclude = names.includes('elai')

console.log(arrayInclude)

false

Comparações entres strings com > < = etc

Leva em conta:

inicias finais do alfabeto sempre serão Maiores > que as letras iniciais do alfabeto

Letras minúsculas são sempre maiores que letras maiúsculas mesmo sendo ‘iguais’

console.log('bela' > 'roger')

false

console.log('bela' > 'Roger')

true

console.log('bela' > 'Bela')

true

# Aula 022

05:36

Booleans true/false

== -> Compara apenas valores

=== -> Compara valores e tipos

const age = 31

console.log(age == 31)

true

console.log(age == '31')

true

console.log(age === ‘31’)

false

Isso acontece porque o JS faz conversão implícita de dados. Para evitar problemas sempre use os operadores ‘exclusivos’ ( comparação estrita ) que comparam valores e tipos e não faz conversões

===

!==

# Aula 023

08:02

Conversão de tipos

let score = '100'

console.log(score + 1 + '\n')

console.log(typeof score + '\n')

score = Number(score)

console.log(score + 1 + '\n')

console.log(typeof score + '\n')

1001

string

101

number

Segundo o professor, o typeof não é a melhor escolha para se definir o tipo de algum elemento

Conversão de um string para número que contiver letras ou símbolos da erro retornando NaN

String(97)

Boolean(10)

Retorna true

Valores falsy – Que retornam false

false

0 – Zero

“” – ‘’ - ``

null

undefined

NaN

Valores truthy

Todos os que não são falsy

# Aula 024

20:11

Segunda etapa – Controle de fluxo – Só citado

Resolução dos exercícios 4

Boolean( null + 1 )

null é avaliado como zero, então ficaria

Boolean( 0 + 1 )

true -> Porque 1 é true

# Aula 025

01:32

O que é controle de fluxo – Explicação

loops

Estruturas condicionais

# Aula 026

12:05

for loop

for(let i = 0; i < 10; i++){

console.log(i);

}

console.log('Cabô o for');

# Aula 027

05:47

whilw loop

let i = 0

while(i < 10){

console.log(i);

i++

}

console.log('Cabô o while');

# Aula 028

19:01

Correção exercícios aula 01

Fluxo - loops

# Aula 029

05:11

if – Primeira parte, sem o else

# Aula 030

05:45

if – Segunda parte – else e else if

# Aula 031

07:12

Operadores lógicos ‘ou’ || e ‘e’ &&

# Aula 032

25:01

Correção aula passada

const hour = 10

const isMorning = hour >=6 && hour <= 11

// Entre 6 e 11

const isAfternoon = hour >=12 && hour <= 17

// Entre 12 e 18

if(isMorning){

console.log('Bom dia')

}else if(isAfternoon){

console.log('Boa tarde')

}else{

console.log('Boa noite')

}

// O else, 'boa noite' será executado sempre

// que não estiver entre isMorning e isAfternoon

// isso significa, 23 -97 4400 210 etc





O Código fica maior, mas fica mais legível



Estes três itens fazem a mesma coisa

ings `)

Último exercício



odd = ímpares

even = pares

# Aula 033

03:22

Operador lógico not ( ! )

Inverte o boolean

let isUserLoggedIn = false

if(!isUserLoggedIn){

// !isUserLoogedIn não tem seu valor reatribuído

// o resultado da expressão com o ! que é usado

console.log('você precisa fazer login para continuar')

}

console.log(true);

console.log(false);

// true

// false

console.log(!true);

console.log(!false);

// false

// true

# Aula 034

03:52

Break e continue

Relacionado a loops

// Exibindo pontuações de um jogo

// Pontuação mínima, 0, não deve ser exibida

// Pontuação máxima atingida, não deve ser

// exibido scores posteriores a pontuação

// máxima

const scores = [50,25,0,30,100,20,10]

let scoresShow = []

console.log('Pontuações: ', scores.toString())

for( let i = 0; i < scores.length; i++){

// Pontuação mínima

if( scores[i] === 0){

console.log('Você atingiu a pontuação máxima');

continue

};

// Pontuação máxima

if(scores[i] === 100){

break

}

// Coninuando loop - Pontuação

scoresShow.push(scores[i])

}

console.log('Pontuações exibidas: ', scoresShow.toString())

Pontuações: 50,25,0,30,100,20,10

Você atingiu a pontuação máxima

Pontuações exibidas: 50,25,30

# Aula 035

A condicional switch e exercícios

05:54

O switch sem o break exibe todas as opções após encontrar o resultado da condição

switch(condição){

case ‘A’:

console.log(“A”)

break

case ‘B’:

console.log(“A”)

Não tem break no default porque ele só vai ser ativado se nenhuma case for válido

break

default:

console.log(“???”)

}

# Aula 036

Correção dos exercícios da Aula 03 da etapa 02

29:16

Usar uma const dentro de um loop é possível porque o valor da variável do tipo const não é alterado e sim é recriado a cada iteração

Oque é mais performático recriar a const ou alterar uma let definida fora do loop?

for(let i = 0; I < vetor.length; i++){

const word = vetor[i]

}

# Aula 037

# Aula 038

# Aula 039

# Aula 040

# Aula 041

# Aula 042

# Aula 043

# Aula 044

# Aula 045