**JAVASCRIPT**

**OUTPUT:**

* Manipulando um item html pelo seu id através do JS;

Document.getElementNyId(“id”).innerHTML = “novo texto”;

* Janela de alerta da página;

Window.alert(“mensagem”);

* Exibir a mensagem no console;

console.log(‘o que quiser’);

**VARIÁVEIS:**

* Use o “let” ou “var” pra criar variáveis.
* Var pode ser usada em qualquer escopo global
* Let só pode ser usada no escopo local, não se vê ele fora do código, por exemplo, no navegador.
* Não pode começar com números
* Frases tem que estar entre parênteses
* Número não precisa
* Pra aparecer as informações no console, digite ‘console.log(variável)’
* ‘const’ é uma constante, uma variável que não pode ser modificada, é útil na utilização de arrays e objetos.
* Concatenação; adição, juntar, somar.

**IF E ELSE:**

* > maior
* < menor
* == igual
* === exatamente igual
* != diferente
* !== totalmente diferente
* >= maior ou igual
* <= menor ou igual
* && E, and
* || Ou, or

**CONDICIONAL TERNÁRIA:**

* Let isMember = true;

Let shipping = isMember ? 2 : 10;

Desmembrando:

isMember **{deu certo?}** ? **{então é 2}** 2 **{deu errado?}** : **{então é 10}** 10;

* ? se sim
* : caso contrário

**SWITCH:**

* let profession= “policial”

switch (profession) {

case ‘fiscal’ :

console.log(sua camisa será verde);

break;

case ‘policial’ :

console.log(‘Sua camisa será azul);

break;

default:

console.log(‘sua camisa será preta)

break;

Desmembrando:

switch (profession) **{a variável que será analisada}** {

case **{caso seja}** ‘fiscal’ :

console.log(sua camisa será verde);

break **{acabou esta case aqui}** ;

default **{normal, caso não satisfaça nenhuma das anteriores}** :

console.log(sua camisa será preta)

break;

}

* switch; troque, trocar
* case; caso, caso seja, if, true
* break: quebrar, separar, terminar
* default; padrão, else

**FUNÇÕES:**

**Arrow Function:** Diferença pra função normal é deixar o código mais limpo, ex;

Function;

Function somar(x, y) {

Return x + y;

}

Arrow Function;

Const somar = (x, y) => x + y;

**Variável de escopo global**: criada fora da função, pode ser usada em qualquer function

**Variável de escopo local**: criada dentro da função, é usada somente naquela situação;

**Função dentro de uma função:**

Function addPtc(a, b) {

Const square = (x) +> x \* x; **[aqui não precisa ser arrow function mas fica melhor pra não se perder]**

Let ptcA = square(a);

Let ptcB = square(B);

Return ptcA + ptcB;

}

**ARRAY´S**

**Lista de algo**

Como criar um array?

* Let colors = [‘blue’, ‘red’, ‘green’];
* Console.log(colors[1])

Vai exibir a cor red.

Da pra pegar as informações de um array e usar em outro? **Sim**

Da pra criar um array dentro de outro array? **Sim**

Como eu faço pra adicionar um item no array depois de tê-lo feito já?

* Nomedoarray.push(‘oquequeradd’);

Pop = remove o último item do array

Shift = remove o último item do array

Array.length = para visualizar quantos itens tem o array

Array[1] = visualizar o item que está no 1

**OBJECT´S**

**Grupo de informações**

Diferença dele pro array, nele eu posso dar características, ex;

Let personagem = {

Nome: ‘Arthur’,

Idade: 20,

Pais: ‘Brasil’

}

Como visualizar somente uma característica?

* personagem.nome

Da pra criar um objeto dentro de um objeto? **Sim, mesmo esquema do array.**

Da pra criar um array dentro de um objeto ou vice-versa? **Sim**

Mudar algum item:

Personagem.nome = ‘pedro’

Adicionar mais idade:

Personagem.idade += 10

Exemplo de um Objeto => Array => Objeto (lista de carros e cores deles:

Objeto = Azul

Array = Verde

Let personagem = {

Nome: ‘Arthur’,

Idade: 20,

Carros: [

{ modelo: ‘Fiat’, cor: ‘preto’ },

{ modelo: ‘Ferrari’, cor: ‘vermelha’ }

]

}

E pra eu acessar somente a cor do Fiat?

* Console.log(personagem.carro[0].cor);

Utilizando uma função dentro de um objeto:

Function: Verde

Let pessoa = {

Nome: ‘Arthur’,

Sobrenome: ‘miranda’,

Idade: 20

nomeCompleto: function () {

return `${this.nome} ${this.sobrenome};

}

Ou ao invés de function se quiser usar arrow function pode também, da seguinte maneira:

nomeCompleto: () => {

return ‘Arthur Miranda’;

}

**LOOP´S**

**Quando eu quero executar algo várias vezes.**

Como é demonstrado o loop? **For**

**Ex:**

for (let n = 1; n <= 10; n++) {

console.log(‘Arthur’ + n);

}

**Iria fazer isto:**

Arthur 1

Arthur 2

Arthur 3

Arthur 4

Arthur 5

Arthur 6

Arthur 7

Arthur 8

Arthur 9

Arthur 10

Laranja: por onde vai começar a minha contagem

Azul: Até onde vai a minha contagem

Roxo: Vai avançar de quanto em quanto (no caso ali de 1 em 1)

Vermelho: pra mostrar o número depois do arthur pra ficar mais intuitivo, mas não é necessário

**LOOP´S EM ARRAY**

Let cores = [

{ nome: ‘preto’, quantidade: 10 }

{ nome: ‘vermelho’, quantidade: 12 }

];

**Maneiras de usar o loop nesse array:**

for (let = 0; n < cores.lenght; n++) {

console.log(cores[n].quantidade[0]);

}

**Outra maneira:**

For (let i in cores) {

Console.log(cores[i].quantidade[0]);

}

**Outra maneira:**

For (let cor of cores) {

Console.log(cor.quantidade[0]);

}

Vermelho: usamos o cores.length pra ele dar loop na quantidade de vezes em que se temos objetos, por exemplo, ali seriam duas vezes, preto e vermelho.

Se eu quiser visualizar só a quantidade do vermelho? Adiciona o verde no código

**EVENTOS:**

Eventos de clique;

1°: Coloque na sua tag ‘button’ no html, o ‘onclick=”função”’

2°: No JS crie uma função para quando o evento for disparado o que você programou aconteça.

Ou

1°: Usar o ‘this’ pra mudar o próprio botão

2°: onclick=”this.innerHTML=’Clicou’”

Eventos de passar o mouse;

1°: Mesmo esquema do onclick, muda-se para ‘onmouseover=”função”’ ou ‘onmouseout=”função”’

2°: No JS crie uma função para quando o evento for disparado o que você programou aconteça.

Existem diversos de eventos com o mouse, se você digitar ‘on’ no vscode ele já te dará diversas opções de eventos.

Eventos de teclado;

1°: Coloque na sua tag ‘button’ no html, o ‘onkey=”função”’, ai pode ser, onkeyup. onkeydown e onkey

2°: No JS crie uma função para quando o evento for disparado o que você programou aconteça.