**Módulo 1: Primeiros Passos**

**1ª aula:**

JavaScript é de fato uma linguagem de programação e não de marcação como html e css. Em seus primórdios possui somente a função de realizar interações e modificações em tempo real em um site web. Porém atualmente, possui várias utilidades e sendo suportado por vários dispositivos, como arduinos por exemplo.

**2ª aula:**

Professor explica sobre como funciona vscode e questão de atualizar página, console do navegador e tals.

**3ª aula:**

Professor mostra a criação e link de scripts. O JavaScript deve ser botado no final da página porque com grandes códigos js, como seu carregamento seria primeiro caso fosse botado na head, acaba fazendo com que a página fique em branco enquanto está o script está sendo carregado, botando o js no final da página garante com que o carregamento inteiro da página seja feito antes do carregamento do js, dando a impressão de mais velocidade do carregamento da página pro usuário.

**4ª aula:**

Dois códigos são ensinados. Primeiro é o alert(“”); que abre um pop-up exibindo o conteúdo digitado e o segundo é o console.log(“”); que imprime os valores no console. Também é possível executar js direto no navegador digitando o código no console e então apertando enter.

**5ª aula:**

Para se criar uma variável deve se usar a sintaxe let. Strings devem ser armazenadas entre aspas, números não precisam. Ao realizar o console.log ou alert, caso estejam entre parênteses, variáveis não serão executadas, tarefa de criar duas variáveis para estado e para cidade.

**6ª aula:**

São explicados os tipos de variáveis em js, sendo eles string, number (não sendo int ou float em js), boolean, array e objeto. Utilizando a sintaxe typeof no console.log, será exibido o tipo da variável, exemplo: console.log(typeof nome). Interessante destacar que tanto objeto quanto array são classificados como object em js. Para iniciar um array utiliza-se [ ], para objetos é usado { }.

**7ª aula:**

Mostrado os comentários em js, que funcionam do mesmo jeito que em php.

**8ª aula:**

Exercícios sobre variáveis. Arquivo “exercicio1.js”

**9ª aula:**

Explicação do ponto e vírgula, que no js não é obrigatório, porém para realizar mais de um comando

em uma linha, é necessário separar os códigos com um ;.

**10ª aula:**

É explicado brevemente o conceito dos 3 maneiras de criar uma variável em js: let, var e const. O const se refere a uma constante, que não poderá ser alterada durante a execução do código, caso haja alguma alteração na mesma, será retornado um erro. Js não possui classificações como private ou public, porém existe a sua classificação quando o assunto são variáveis. O let funciona como um private, permitindo que a variável seja utilizada somente no contexto em que foi criada, já o var funciona como um public, que permite a utilização da variável pelo código inteiro.

**11ª aula:**

São mostrados os operadores matemáticos de maneira básica, além de como funcionam as concatenações, que no js funcionam com um símbolo de mais (no caso de strings, com numbers irá funcionar como o operador matemático normal, realizando a soma). Não possuí uma quantidade máxima de concatenações que podem ser realizadas ao mesmo tempo, podendo ser feitas diversas vezes dentro de uma mesmo operação.

**12ª aula:**

É mostrado a TemplateString, que se trata da string iniciada por crâses, sendo interpretada, facilitando assim a concatenação de elementos, sempre retornando strings. Exemplo: let nomeCompleto = `${nome} ${sobrenome}`;

**13ª aula:**

Explicação de if e else, além dos operadores condicionais para os mesmos. Já fiz isso vezes de mais em minha vida para ter de realizar essa atividade de escrita para fixação das informações.

**14ª aula:**

Mostrado os símbolos de igualdade == e ===. A igualdade “simples” (==), não necessita que os dois lados sejam do mesmo tipo, de um lado pode haver uma string e do outro um number, contanto que ambos possuam o mesmo valor, a igualdade será verdadeira. Porém, a igualdade “real” exige que os dois lados possuam o mesmo valor e mesmo tipo, só serão aceitas strings que possuem o mesmo valor, ou numbers que possuem o mesmo valor.

**15ª aula:**

Explicado sobre as multi-condicionais, sendo o && e o ||. O && representa “e”, no sentido em que as duas ou mais condições devem ser verdadeiras para a operação ser verdadeira. Já o || representa o “ou”, em que somente umas das condicionais precisa ser verdadeira para a operação ser verdadeira.

**16ª aula:**

Mostrado o else if, que fornece uma condição para um else, que é acionado caso a operação inicial não seja atendida.

**17ª aula:**

Explicado como o if nada mais nada menos verifica se uma operação é verdadeira ou falsa, como um boolean. É possível especificar uma operação de verificação em uma variável, o que tornará a mesma em tipo boolean.

**18ª aula:**

Exercícios sobre condicionais. Arquivo “exercicios2.js”.

**19ª aula:**

Mostrado if ternário, sua sintaxe é de “let verificacao = afirmacao ? true : false”. Serve para encurtar a construção de um if.

**20ª aula:**

Explicado sobre switch. Sua sintaxe inicia com switch(\*variável a ser analisada\*), em seguida, aberto {}, será inseridos os cases, que verificam se o valor corresponde ao da variável (strings também podem ser analisadas). Deve-se por um break ao final de cada case, para que os próximos cases não sejam executados. Pode-se também por um default, que representa o que será feito caso nenhum dos cases seja atendido.

**Módulo 2: JavaScript Básico**

**1ª aula:**

Introdução a funções. Uma função é um trecho de código geralmente independente que pode ou não receber algum valor, realizando operações definidas com base nesse valor. Funções não necessariamente precisam receber um valor, pois podem também realizar operações próprias, retornando valores quanto solicitados (select de sql).

**2ª aula:**

Criada a primeira função. Funções não são executadas após serem criadas, pois é necessário que as mesmas sejam chamadas. Uma função inicialmente é um trecho completamente separado de código.

**3ª aula:**

Chamada da função. Para chamar uma função, basta repetir o nome da função e e passar seus devidos parâmetros, caso haja algum.

**4ª aula:**

Mostrado como colocar parâmetros para a função.

**5ª aula:**

Mostrado como receber o retorno de uma função. Basta inserir return na última linha da função. Todo código que vier após o return não será executado.

**6ª aula:**

Feito uma função com return condicional, dependendo do resultado do if, será dado um determinado retorno.

**7ª aula:**

Exercício sobre funções. Arquivo “exercicios3.js”.

**8ª aula:**

Exercício sobre funções. Arquivo “exercicios4.js”.

**9ª aula:**

Exercício sobre funções. Arquivo “exercicios5.js”.

**10ª aula:**

Explicada a arrow function. Se trata de uma maneira simplificada de criar uma função. Basta declarar uma variável e em seu resultado abrir (), passar os parâmetros e utilizar a seta \*=>\* depois, definindo a função. Caso a função possua só uma linha, nem é necessário o return, a própria operação será o retorno. Caso seja somente um parâmetro, também não é necessário dos (), basta declarar o parâmetro e usar a seta.

**11ª aula:**

Explicadas as variáveis globais e as locais. As variáveis globais são aquelas que são declaradas fora de qualquer função e podem ser usadas diretamente por qualquer função, sendo alteradas conforme a função. Já as variáveis locais, são as declaradas dentro de funções e podem ser utilizadas somente pelas funções. As globais, caso sejam alteradas por uma função, serão exibidas corretamente caso sejam imprimidas depois fora do escopo de qualquer função, já as locais, deverão ser retornadas ou imprimidas dentro da função. Caso exista uma variável global e local com o mesmo nome dentro de uma função, a que será alterada dentro da função será a local, tendo sua prioridade.

**12ª aula:**

Explicada a criação de uma função dentro de outra. Funciona normalmente, tanto a declaração como utilização da função são exatamente iguais ao normal, sem alterações. Opinião do professor que concordo, preferir criar uma arrow function dentro de uma função, pois da uma maior aparência de organização, quando se tem uma function dentro de function.

**13ª aula:**

Introdução a arrays, são instanciados por [] e podem conter diversos itens diferentes, de tipos diferentes, também sendo possível criar arrays dentro de arrays. Para pegar um index de um array dentro de outro, o primeiro [] irá tratar do primeiro array e o segundo [] irá selecionar o array interior, sendo que o segundo array pode ser armazenado dentro de uma variável, para maior facilidade na hora de acessar.

**14ª aula:**

Mostrada 4 funções de arrays no js: length(), push(), pop() e shift(). Todas são instanciadas por [array].função(). Lenght() conta a quantidade de itens dentro do array, de certa forma medindo o comprimento do array conforme o nome sugere. Push(), irá adicionar um valor dentro do array, que irá possuir o último local (array.push(‘salve’)), só lembrar do que o push no git faz. Pop() irá remover o último valor do array, excluindo sua index, função que não precisa de parâmetro, igual shift. Shift() irá remover o primeiro item do array, removendo sua index, ou seja, valor que o a index 1 irá passar a assumir a index 0.

**15ª aula:**

Exercícios sobre arrays. Arquivo “exercicios6.js”.

**16ª aula:**

Explicação de objetos. Funcionam como array porém a index deve ser definida também, dando maior descrição ao elemento. Sua sintaxe é parecida com e de um array, porém com {}. Index também deve ser definida (let objeto = {nome: ‘sla’, idade: 28}). Para chamar os itens, deve-se referir ao objeto.index.

**17ª aula:**

Maior aprofundamento sobre objetos. Para alterar algum valor presente em um objeto, basta chama-lo e usar o = para definir o valor. Pode-se haver um objeto com um array de outros objetos, quando deseja-se maior especificação de um objeto.

**18ª aula:**

É possível criar funções dentro de um objeto, basta colocar function () {} depois de ter nomeado a index. Interessante notar que parar acessar os valores do próprio objeto dentro dessa função será necessário utilizar o this.valor, porém, a arrow function é conhecida como função anônima, portanto não reconhece seu pai, não funcionando o this. Não é necessário nomear a função pois a própria index será o nome a ser usado na chamada da função.

**19ª aula:**

Explicação do for. Igual if, já realizei muitas vezes para precisar da atividade de escrita para fixação do conteúdo.

**20ª aula:**

Explicada a utilização de in e of dentro de um for, passando em um array. A sintaxe é de for (let cont in array){}, desta forma, cont se torna o contador do for, de acordo com quantos elementos houverem no array. Of funciona da mesma maneira, porém a diferença entre os dois é que o of funciona como um foreach, em que a variável definida se torna o próprio valor daquele loop da execução, enquanto o contador de in deve ser instanciado na hora de acessar o valor do array. Exemplos: for (let produto in produtos) { console.log(`${produtos[produto].valor} é tal coisa` }; for (let produto of produtos) { console.log(`${produto.valor} é tal coisa` }; (prática no arquivo exercicio7.js)

**21ª aula:**

Explicação do while. Idem a aula anterior. Tomar cuidado em sua execução, pois pode acabar executando para sempre, o que também é possível com for.

**22ª aula:**

Exercícios de loops. Arquivo “exercicio7.js”

**23ª aula:**

Mostrado alguma funcionalidades e funções. A função join() irá juntar TODOS os elementos de um array em uma única string, elementos esses que serão separado pelo que for definido como parâmetro (array.join(‘ -> ‘) irá separar os elementos do array pela seta indicada). Também foi mostrado que para alterar o último valor de uma array por exemplo, pode-se usar array[array.length - 1] = ‘Novo valor’. Ou seja, é possível realizar operações matemáticas dentro da index de um array. Maneiro : )

**24ª aula:**

Mostrada mais duas funções. Array.sort() irá organizar os valores de forma decrescente e função array.reverse() irá inverter a ordem dos valores do array, alterando a index dos valores também, tornando o novo primeiro valor para index 0. Quando deseja-se ser mais específico com que item ser o parâmetro para sort, pode-se criar uma função dentro do mesmo para realizar a organização do array, fazendo uma operação de subtração entre um item e o próximo, caso o valor seja negativo, indica que o valor de tal index deve ficar atrás do valor próximo a ele, caso o valor seja positivo ele irá ir para frente e se o valor for igual, nada irá acontecer. Confuso, sendo mais compreensível em utilização prática. Array.sort((a, b) => a.index - b.index.