**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.

**25ª aula:**

Mostradas mais algumas funções relacionadas a arrays. Filter() irá verificar e retornar um array com os elementos que atenderem a condição especificada em seu parâmetro (como ele retorna um novo array, é necessário que o mesmo seja armazenado em outra variável). A função every() verifica a condição especificada em seu parâmetro para todos os elementos do array, retornando true ou false. some() realiza a mesma coisa, mas precisa somente de um elemento que atenda a condição e não todos. A função includes() procura dentro de um array pela string ou number que for especificada no parâmetro, retornando true ou false.

**Módulo 3: Eventos e manipulação do DOM**

**1ª aula:**

Introdução do DOM. Nada mais são que informações armazenadas na memória do navegador, contendo a definição dos elementos e seus conteúdos, podendo ser selecionados. Esses seletores tornam possível a alteração em tempo real da página observada.

**2ª aula:**

Introdução ao seletores. Existem dois tipos de seletores: window e document. Window se refere aos elementos dos “meta-dados” da página, como o endereço da página e afins. Já o document aborda os elementos que estão dentro da página, basicamente a body. Mostrado o seletor document.getElementsByTagName(), que irá retornar um array com todos as tags encontradas que foram definidas no parâmetro.

**3ª aula:**

Mostrados mais alguns seletores. O document.getElementById() é um seletor que pega diretamente o id de um elemento, retornando diretamente o seu conteúdo. Já o document.getElementsByClassName() também irá retornar um array, com todos os elementos encontrados da classe especificada no parâmetro. Após isso, é mostrado o seletor mais volátil, o document.querySelector(“”), que seleciona os elementos igualmente é realizado no css, podendo-se aprofundar em elementos por suas tags e demais seleções desejadas, sendo de certa forma a mais fácil. O document.querySelectorAll(“”) irá retornar um array com todos os elementos encontrados com o parâmetro fornecido.

**4ª aula:**

Mostrados event listeners. A função seletor.addEventListener(“evento”, função()) irá criar um listener, geralmente sendo de click. No caso do evento de click, é possível definir no próprio elemento o onclick=””, que irá executar o js ou função definida em seu parâmetro.

**5ª aula:**

Mostrados alguns manipuladores de elementos. O seletor.children, irá retornar um array com os elementos filhos do elemento que for selecionado, valendo lembrar que caso tenha algo escrito no elemento, nada será retornado, porém só os filhos importam. O seletor.innerHTML irá retornar o código html interior, ou seja, filhos e elementos, como texto, dentro daquele elemento, podendo alterá-los também. Seletor.outerHTML raramente é utilizado, faz a mesma função do innerHTML, porém também seleciona e altera o próprio elemento passado no parâmetro e não somente o que está em seu interior. Também existe o .innerText, que faz o mesmo que innerHTML, porém não interpreta como marcação, recebendo e aplicando uma string do que estiver em seu conteúdo, incluindo tags.

**6ª aula:**

Mostrado .append() e prepend(). Append pode ser reproduzido por innerHTML += conteúdo, porém este último irá substituir todos os elementos e adicionar os próximos, fazendo com que o processamento seja maior, enquanto append somente adiciona um elemento após o que foi selecionado, porém, sendo somente texto e não marcação, também valendo para prepend, que adiciona antes do elemento selecionado. Para criar uma marcação com append, deve-se criar um elemento com document.createElement(“tag”) e armazená-lo em uma varável, para então com innerHTML adicionar o conteúdo em texto e por último definir a variável no parâmetro de seletor.appendChild(variável).

**7ª aula:**

Mostrado o after() e before(). Eles adicionam um novo elemento após ou antes, respectivamente, ao elemento seletor.

**8ª aula:**

Mostrado os seletores hasAttribute(), getAttribute() e setAttribute, além do seletor.value. HasAttribute verifica se há ou não um determinado atributo dentro de uma tag, retornando um array. GetAttribute seleciona o conteúdo de um determinado atributo e setAttribute seta o conteúdo de um atributo (deve-se definir o atributo a ser selecionado depois por uma vírgula e definir o novo conteúdo do atributo selecionado).

**9ª aula:**

Alteração de css com js. Basta colocar seletor.style.tagCss e defini-la.

**10ª aula:**

Mostrado o operador seletor.classList. Por si só, retorna um array com as classes presentes. Tem também os operadores filhos, como classList.add(‘classe’), classList.remove(‘classe’), classList.contains(‘classe’), classList.replace(‘classe a ser tirada’, ‘classe a ser colocada’) e classList.toggle(‘classe’).

**11ª aula:**

Mostrados eventos de keyup(), keypress() e keydown(). Podem ser diretamente definidos no html com onkeyup= , onkeydown= e onkeypress ou também podem ser utilizados em um addEventListener, porém sem o on na frente.

**12ª aula:**

Quando se usa uma função é realizada a partir de um evento key, é possível atribuir uma variável no parâmetro com informações sobre a tecla pressionada. Event.code irá mostrar a tecla pressionada, diferenciada no lado da tecla caso seja a de funções e também com maior descrição das letras e números pressionados, enquanto event.key irá retornar somente a tecla que foi pressionada, sem maiores detalhes. Também existem event.ctrlKey, event.altKey e event.shiftKey que irão retornar um boolean retornando true ou false dependendo se o botão estava ou apertado quando o evento foi realizado.

**13ª aula:**

Exercício sobre manipulação de eventos. Fazer com que quando seja apertado enter no input criado, seja adicionado uma li na ul com o texto inserido no input.

**14ª aula:**

Resolução do exercício. Arquivo “exercicio8.html”.

**Módulo 4: Objetos e Orientação a Objetos**

**1ª aula:**

Aula de introdução a orientação a objetos.

**2ª aula:**

Explicado a existência dos paradigmas de programação procedural, POO e programação funcional.

**3ª aula:**

Explicado o constructor e o this. Constructor é o nome da classe construtora no javascript. Sua sintaxe é de class Pessoa { constructor(value){ this.value = value}}. Também falado sobre o this, que é utilizado para garantir que o próprio objeto de vários possivelmente criados, será o referenciado.

**4ª aula:**

Ensinado a criar instâncias de uma classe. Let objeto = new Classe e tals.

**5ª aula:**

Aula sobre ações. Basicamente funções, porém no js não é necessário declarar uma function, basta colocar o nome da ação e ().

**6ª aula:**

Explicação sobre getters e setters no js. Sua sintaxes são get elemento() {} e set elemento(){}, respectivamente, get retornando um valor e set setando-o.

**7ª aula:**

Explicado questão de herança e polimorfia. Quando for se referir a um emento de uma classe pai, deve-se utilizar o super(elemento) ou super.método, para métodos.

**8ª aula:**

Falado sobre static. Ao utilizar essa sintaxe na criação de um atributo ou método, quer dizer que aquele elemento irá se referir diretamente a classe e não a um objeto em específico, portanto, sendo o mesmo para todos os objetos. Para chamar um elemento static, deve-se usar Classe.elemento ao invés de objeto.elemento.

**9ª aula:**

Mostrado o conceito de factory. Se trata de uma função, fora da classe, que irá criar um objeto daquela classe, criando com base nos parâmetros construtores da classe e outro também, de acordo a factory.

**10ª aula:**

Inicio do projeto de criador de formulários. Criada classe do input e definido atributos type e required e o método construtor.

**11ª aula:**

Criação das classes de botão e formulário. Botão é uma classe filha de input, herdando seus valores e a classe formulário recebe os objetos de input e botão.

**12ª aula:**

Finalização do projeto. Criado os métodos de render, exibindo os elementos na tela.

OBS: O curso havia sido refeito, então as aulas dos módulos 5 e 6 são de conteúdos já vistos, então decidi pular eles.

**Módulo 7: DOM**

**12ª aula:**

São mostrados os comandos elemento.clientHeight, elemento.clientWidth além dos comandos elemento.scrollHeight e elemento.scrollWidth. Os dois retornam a quantidade de pixeis dos respectivos lados da página, porém, o scroll também inclui a rolagem do elemento, caso ela exista no mesmo, dando o tamanho real do conteúdo e não somente o visual. Ambos os comando também incluem o padding e margin dos elementos. Também existe o elemento.offsetHeight que retorna o padding-top e do padding-bottom.

**13ª aula:**

Mostrado como verificar a posição de um scroll presente na página em si ou de um elemento. Em um elemento, se trata do elemento.scrollTop ou elemento.scrollLeft, ambos irão retornar quantos pixeis existem acima ou a esquerda de um scroll. Para a página em si, seria os comandos window.scrollX para horizontal e window.scrollY para a vertical, retornando os mesmos valores, porém referente a página. Também foi mostrado comando elemento.scrollTo(a, b). Ao colocar os dois valores como 0, irá retornar ao topo do scroll, geralmente sendo a função necessária. Também é possível passar um objeto dentro do comando, para definir a opção smooth, fazendo com que a mudança seja suave. Exemplo: elemento.scrollTo({top: 0; left: 0; behavior: ‘smooth’}).

**14ª aula:**

Criada função que verifica o scroll, fazendo com que o botão suma caso esteja no topo da página.

**Módulo 8: Manipulações, Arrow, Ajax, etc**

**1ª aula:**

Mostrados comandos elemento.length e elemento.indexOf(‘string’). IndexOf irá pesquisar em uma string o valor inserido no seu parâmetro, caso ache, retornará uma int referente a posição do que foi encontrado na string. Caso não ache, será retornado o valor -1.

**2ª aula:**

Mostrado elemento.slice, elemento.substring e elemento.substr. As 3 funções recebem o primeiro e um segundo parâmetro do tipo int, que nos dois primeiros irá determinar a posição inicial e final do corte, podendo ser negativo no slice, iniciando a contagem o último caractere. Nos dois primeiros, caso somente um número seja passado, será pegado tudo que está depois da index passada. O substr em específico, o segundo número irá se referir a quantos caracteres serão pegos depois da index passado, funcionando de maneira diferente do slice e substring. O mais recomendado é o substr, pois ele realiza tudo que os outros fazem, sem limitações, porém está depreciado.

**3ª aula:**

Passado algumas funções que manipulam strings. Elemento.replace recebera dois parâmetros, o primeiro é qual string deverá ser substituída e o segundo parâmetro será o que irá substituir. Elemento.toUpperCase e elemento.toLowerCase transformam toda a string em maiúscula e minúscula, respectivamente. Elemento.concat é bem inútil, pois irá concatenar a string com o que for passado em seu parâmetro, porém há outras maneiras até melhores do que utilizando uma função. A função elemento.trim() é extremamente útil, pois retira todos os espaços em branco de uma string. A função elemento.charAt() irá retornar a string que estiver na index passada no parâmetro. A função elemento.split() irá receber como parâmetro o que irá definir o corte dos elementos, transformando a string em um array.

**4ª aula:**

Mostradas funções de manipulação de números. Elemento.toString() transforma uma string que contém um número em uma string. Elemento.toFixed(x) irá mostrar as casas decimais de acordo com a quantidade inserida em seu parâmetro, em um float (n = 12.213123, n.toFixed(3) irá mostrar 12.213). A função parseInt(string) irá transformar uma string em um int e parseFloat(string) irá transformar uma string em um float.

**5ª aula:**

Mostradas algumas funções de array. Array.toString() irá transformar um array em uma string, separando os elementos por virgula. Array.join(delimitador) irá transformar um array em uma string separando os elementos por meio de delimitador definido em seu parâmetro.

**6ª aula:**

Continuação de funções da array. Array.splice(x, y) irá receber dois parâmetros, o primeiro sendo a index do elemento que se deseja excluir e o y sendo quantos itens a partir desse deverão ser excluídos, para excluir um elemento basta que y = 1. Caso somente o primeiro parâmetro seja fornecido, serão excluídos todos os elementos a partir da index passada. Array.concat(array2) irá juntar os dois arrays, deixando os elementos do segundo logo após o primeiro. Array.sort() organiza os elementos alfabeticamente e array.reverse() inverte a ordem dos elementos, não importando como estão ou não organizados.

**7ª aula:**

Array.map( function () {}) geralmente é utilizada com uma função, irá executar o que estiver em seu parâmetro, retornando os valores que passarem pela função em um novo array.

**8ª aula:**

Array.find(function (item) {}) irá retornar completamente o primeiro elemento que atender a condição passada em seu parâmetro. Array.findIndex() fará a mesma coisa, porém só retornando a index do elemento.

**9ª aula:**

A classe Date() é utilizado para gerar um elemento do tipo data, podendo receber vários parâmetros como ano, mês, dia, hora, minutos, segundos. Há duas formas de passar uma data: definindo os valores separados por virgula (dessa maneira, os meses também se iniciam pelo 0, ou seja, janeiro é 0 e dezembro é 11, tomar cuidado), ou passando uma string, separando por hífen os elementos. Data.toDateString() irá retornar de forma organizada a data passada. Data.toUTCString() irá retornar o horário de Greenwich, junto a data passada.

**10ª aula:**

Data.getFullYear() irá retornar o ano, data.getMonth() retorna mês, data.getDay() retorna o dia da semana de 0 a 6, sendo domingo 0. data.getDate() retorna o dia do mês. Data.getHours() retorna a hora do dia, data.getMinutes() pega os minutos e data.getSeconds() retorna os segundos. Data.getTime() retorna um time stamp de milissegundos que se passaram desde 1970.

**11ª aula:**

Mostradas funções de alteração das datas. Mesmas do get da aula anterior porém com set no lugar. Recebem um parâmetro int para alterar o horário, sendo possível fazer operações matemáticas em seu parâmetro.

**12ª aula:**

A classe Math apresenta várias operações matemáticas. Math.min() irá retornar o menor dos números do conjunto de números passado em seu parâmetro, Math.max fará o oposto. Math.round irá arredondar um float com base em se é maior ou menor que 5. Math.floor irá sempre retornar o arredondado para baixo e Math.ceil irá fazer o oposto. Math.random() irá fornecer um número quebrado entre 0 e 1. É possível manipulá-lo usando Math.floor(Math.random() \* x), retornando um número aleatório entre 0 e o valor definido para x.

**13ª aula:**

Mostrado o setInterval(). Recebe dois parâmetros, o primeiro é o nome da função que será executada, sem parênteses e o segundo parâmetro e a quantidade de milissegundos de intervalo entre a execução da função, 1000 é igual a um segundo. Para parar o setInterval, basta usar a função clearInterval(), passando como parâmetro a variável que contem o setInterval.

**14ª aula:**

Passada função setTimeout que diferente do setInterval, ao passar a função no primeiro parâmetro e depois o tempo em milissegundos no segundo, o timeout irá aguardar o tempo definido e quando o mesmo acabar irá executar uma única vez o que está definido e somente chamando o timeout novamente para ser executado mais de uma vez. É possível interrompe-lo antes que ele execute a função definida com clearTimeout() e o nome da variável que contem o timeout.

**15ª aula:**

Aula sobre Template Strings, conteúdo já mostrado anteriormente no curso.

**16ª aula:**

A desconstrução de objetos se trata de uma nova maneira de armazenar valores de um objeto, introduzidos no ecmascript 6. Sua sintaxe é let { atributo1:novoNome , atributo2 = valorPadrão, atributo3 } = objeto. Como mostrado em novoNome, caso queira alterar o nome da variável para outro além do próprio nome do atributo, basta utilizar os dois pontos e colocar o novo nome para a variável. Ademais, caso algum dos atributos não estiver definido, é possível colocar um valor padrão com igual(=) e o valor definido será pego somente se nada estiver definido no atributo, pois se estiver, o valor do atributo terá prioridade.

**17ª aula:**

Para pegar objetos dentro de um objeto na desconstrução, existem duas maneiras. A primeira é definindo o objeto filho no final da desconstrução: let { atributo1, atributo2 } = objeto.objetoFilho. Porém, dessa maneira não é possível pegar atributos do objeto pai, pois dará erro. A outra maneira é declarando o objeto filho no atributo: let { atributo1, objetoFilho:{ atributo2 } } = objeto.

**18ª aula:**

Mostrada a desconstrução de arrays. Bem similar com a de um objeto só que mais simples. O que muda é que é necessário definir um novo nome para a variável e ela automaticamente receberá o valor que corresponde a index do array. Como array não possui nenhuma forma de diferenciar os valores, somente a index, ela será o que irá definir qual variável da desconstrução ira receber, sendo possível ignorar um valor que não se deseja somente colocando uma vírgula em branco let [ a,, b, c] = [ 1, 2, 3, 4], fazendo com que a = 1, b=3 e c=4. Também é possível desconstruir um array assim que o mesmo é criado: let [ nome, sobrenome ] = [ ‘santhiago’ , ‘pereira’ ].

**19ª aula:**

Explicação de arrow functions, já feita anteriormente.

**20ª aula:**

Mostrado operador spread, que poupa tempo na hora de complementar arrays ou objetos. Após definir um array ou objeto, pode-se pegar todos seus elementos e coloca-los em outros array ou objeto, com o operador spread: let array1 = [1,2,3,4]; let array2 = [...array1, 5,6,7,8]. Ao imprimir o array2 seu resultado será: [1,2,3,4,5,6,7,8]. O mesmo vale para objetos, colocando o operador spread, é possível recuperar todos os métodos e atributos do objeto anterior, além de adicionar novos elementos, sendo essa sua principal funcionalidade.

**21ª aula:**

Mostrado o operador rest. É utilizado quando não se sabe ao certo quantos elementos serão passado no parâmetro de uma função. Com o rest, todos os parâmetros serão recebidos: function val(...valores) {}; val(1,2,3,4,5,6). No caso, a função receberia todos os números passados no parâmetro valores.

**22ª aula:**

Mostrado includes e repeat. Includes já foi mostrado anteriormente, elemento.repeat(x) irá repetir o elemento quantas vezes for definido em seu parâmetro, sendo muito útil para testes.

**23ª aula:**

Mostrado algumas funções da classe Object. Object.keys(array/objeto) irá retornar as chaves do array ou objeto, sendo as index do primeiro e os atributos do segundo. Object.values(array/objeto) retornará os valores das index ou dos atributos. Object.entries retornará um array para cada chave, com duas index geralmente, a primeira sendo a chave e a segunda com o valor.

**24ª aula:**

Mostrado as funções padStart() e padEnd(). São usados quando precisa que uma string tenha pelo menos determinados número de caracteres. Essas funções recebem como primeiro parâmetro a quantidade de caracteres necessários e como segundo, o caractere que será repetido até completar a quantidade necessária. Caso a string passada para essas funções já for maior ou igual ao número definido, nada irá acontecer. Muito utilizado para mascarar números de cartão e afins, quando exibido para o usuário.

**Módulo 9: Requisições e assíncronismo**

**1ª aula:**

Explicação básica sobre requisições. Acontecem quando o computador do usuário tenta acessar o site de um servidor por exemplo ou quando site busca dados presentes no servidor.

**2ª aula:**

Explicado que o JSON utiliza de objetos javascript para a comunicação.

**3ª aula:**

A função JSON.parse() transforma uma string contendo um objeto(pois é dessa maneira que um JSON é passado pelas requisições, exemplo: ‘{“nome”: “Santhiago”, “idade” = 20}’) em um objeto novamente.

**4ª aula:**

A função JSON.stringify() realiza exatamente o oposto do parse, transformando um objeto em uma string para ser enviada para a requisição.

**5ª aula:**

Explicação sobre execuções síncronas e assíncronas. As primeiras são linhas de código que devem ser executadas e concluídas, para que a próxima linha seja executada e código tenha continuidade, a maioria do código é assim. Já as segundas são aquelas em que quando executadas, não esperam o retorno e conclusão da execução para a continuação do código, elas são executadas e o código já segue logo em seguida e quando são concluídas posteriormente, é realizada alguma operação com os dados recebidos.

**6ª aula:**

Explicação sobre callback. Se trata do que será realizado quando um código assíncrono terminar de executar. O que derá feito nesse momento é decidido por um callback, geralmente fazendo com que o resultado do código assíncrono vá para uma função.

**7ª aula:**

Mostrado a classe Promise. Funciona basicamente como um try catch do php. Como seu parâmetro, geralmente recebe uma funcão, que terá como parâmetros resolve e reject, sendo o resolve quando a execução der certo e reject quando ela der errado, nem sempre sendo utilizada. Exemplo: let promise = new Promise(function(resolve, reject) { resolve(console.log(‘deu certo!’)); reject(console.log(‘deu errado’)); };. Depois a promise pode ser recuperada com promise.then(function(){}) e promise.catch(function() {}).

**8ª aula:**

Explicação sobre fetch(URL, method). É utilizado para realizar conexões com APIs, geralmente, sendo o prmieiro parâmetro a URL do servidor que contém os JSONs a serem capturados, e o segundo parâmetro a forma que será realizada a comunicação, seja por GET, POST, PUT, etc. Também deve ser utilizado then e catch em sua execução. Após sua execução, fetch irá capturar os JSON presentes na URL, a partir daí deve-se decidir o que será feito com tais informações. API fake para testes: <https://jsonplaceholder.typicode.com>.

**9ª aula:**

Criada função para exibir os posts, recebidos pela api na aula passsada.

**10ª aula:**

Mostrado o async. Deve ser utilizado em funções assíncronas, dessa maneira não presicando do then e catch. Para tal, é necessário armazenar o fetch ou a execução assíncrona em uma variável e depois chama-la usando o termo await, que como o nome diz, irá esperar a conclusão do código, já realizando o trabalho do then e catch.

**11ª aula:**

Explicação do uso do método post no fetch. Para passar o post, deve-se colocar no segundo parâmetro do fetch um objeto, definindo o method como post, também sendo possível atribuir informações ao corpo da requisição (body) e o cabeçalho (headers).

**12ª aula:**

Para capturar um arquivo com javascript, deve-se selecionar o elemento que está com o arquivo, pode ser com getElementById, então usar seletor.files[0] (usa-se o a index 0 para caso de arquivos unicos, pois o files retorna um objeto contento os arquivos enviados, sendo o index 0 o primeiro arquivo). Para mandar um arquivo com fetch, precisa-se criar uma variável da classe FormData() e nela passar com append, por exemplo, o arquivo e demais informações, caso seja de utilidade. Também é necessário definir em seu cabeçalho o content-type como multipart/form-data.

**13ª aula:**

Criação de thumbnails com js. Para fazer isso, deve-se capturar o arquivo desejado, depois criando um elemento do tipo img, colocando como src um objeto URL.createObjectURL(arquivo). Após isso, basta fazer um append no local em que deseja inserir a thumb.

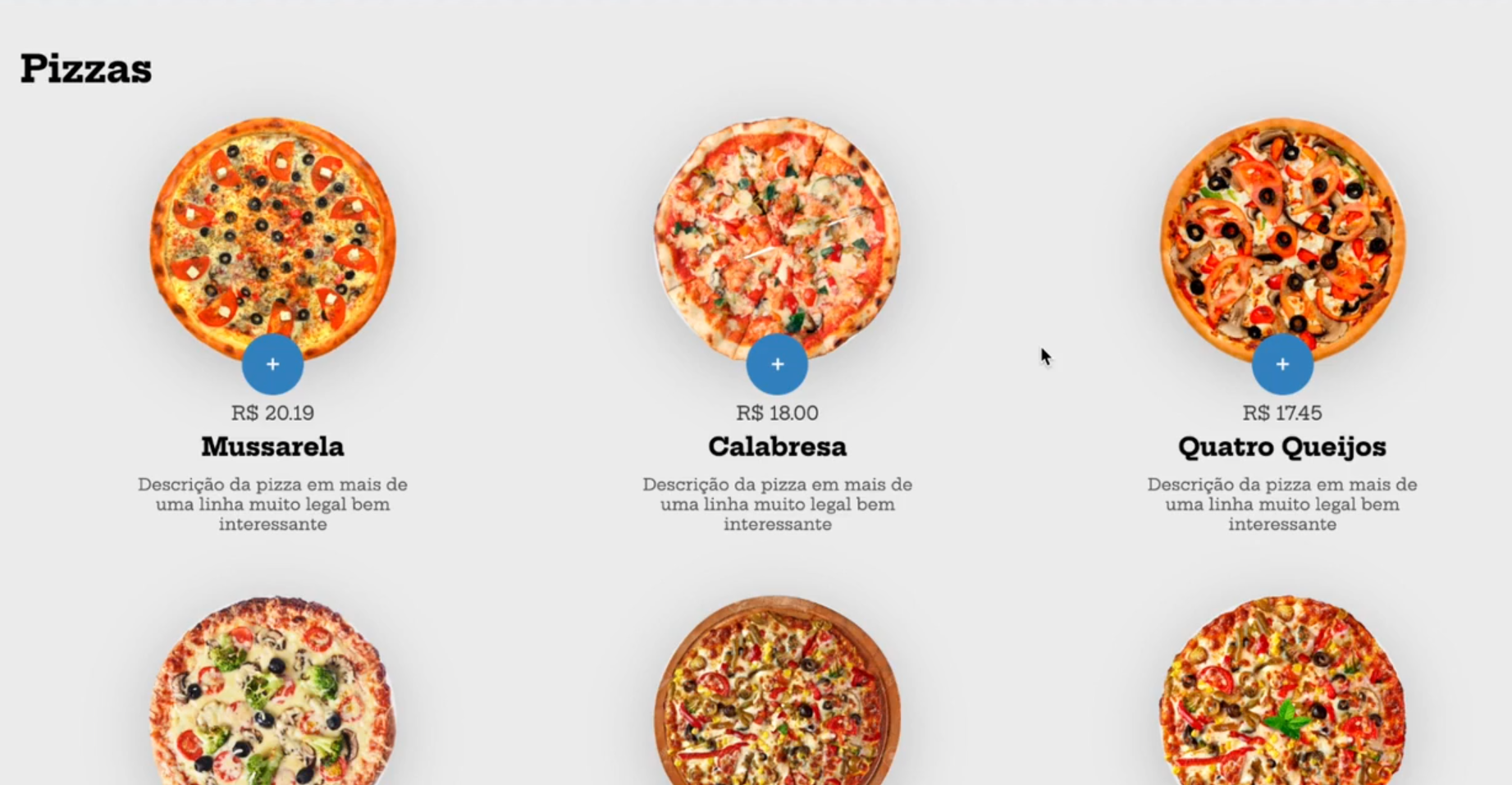
**14ª aula:**

Mostrado o file reader, para realizar a thumbnail. A principal diferença entre as duas se trata de que utilizando o file reade, a operação se torna assíncrona. Ao criar uma variável que contem a classe File Reader(), deve-se então criar uma função para fileReader.onloadend. Dentro desta função, basta fazer a criação do elemento da mesma forma como feito na aula anteiror, porém na src, colocar fileReader.result. Após isso, deve instanciar o arquivo em fileReader.readAsDataURL(arquivo).

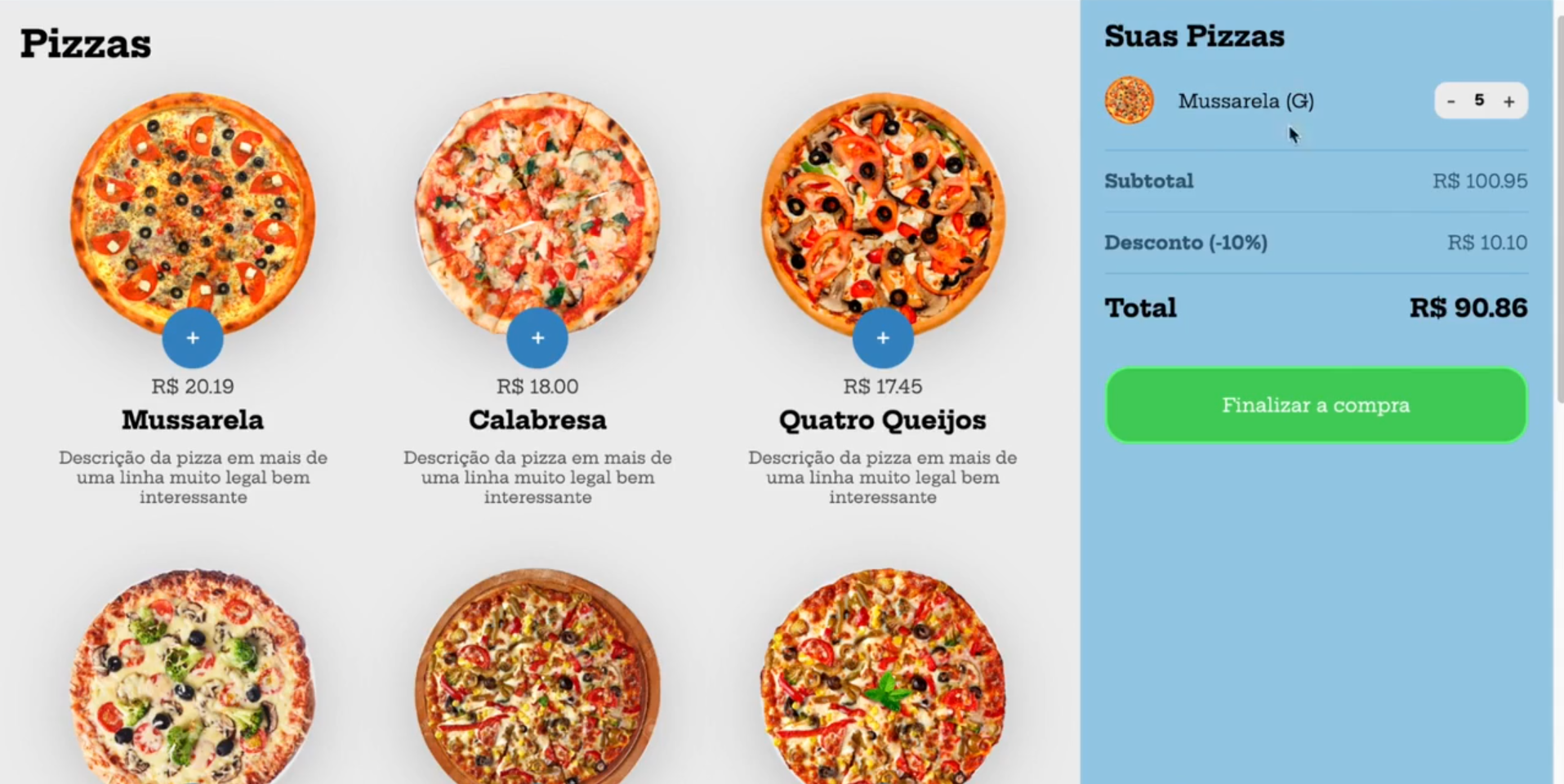
**Módulo 10: Projeto: Compra de Pizzas**

**1ª aula:**

Explicação do funcionamento do projeto:









**2ª aula:**

Feita uma estrutura inicial para apresentar as pizzas na tela, porém sem imagens ou valores das pizzas. Utilizado map na json das pizzas em pizzas.js. O map realiza uma clonagem dos elementos dentro da classe pizza-item, com o método cloneNode(true) e depois imprime as informações na div de classe pizza-area (lembrando que a classe pizza-item está com display none, pois a mesma é encapsulada, pelo menos em meu entendimento). Também criados métodos auxiliares que reduzem a escrita de querySelector e querySelectorAll.

**3ª aula:**

Colocada as demais informações sobre as pizzas, coletadas no JSON. Basta fazer uma querySelector no pizzaItem e realizar um innerHTML no determinado campo de informação (nome, preço, descriçao, etc). Única diferença ocorre no preço, que deve receber um fixed(2), para que todos os preços fiquem padronizados.

**4ª aula:**

Adicionado eventLisntener nas pizzas para quando serem clicadas, exibir o modal, que está originalmente com display none. Após clicar na pizza que possui o trigger, a primeira operação é a de previnir a operação padrão, pois a taga dessse trecho se trata de um link(<a>), atualizando a página. Depois é definido a opacidade do modal como 0, em seguida é alterado o display da modal para flex. Por último, é criado um setInterval de 1 milissegundo aplicando a opcidade do modal para 1. Isso acontece pois no css, está definido que entre alteração de estilos na div, tera uma animação de meio segundo fazendo um efeito visualmente agradável, quando a opacidade é alterada de 0 para 1. O setInterval é necessário poconta da velocidade do javascript em ser executado, que não deixaria com que a animação ocorresse, pois pratiacamente instantaneamente seria executada a segunda linha de opacidade, então o setInterval garante que havera uma espera, mesmo que impercptivel para o usuário, que irá permitir com que ocorra a transição.

**5ª aula:**

Feita a adição das informações da pizza clicada, conforme sua modal. Para isso é necessário setar o id de cada pizza, nesse caso, utilizando um atributo html data-key, para fazer isso basta realizar um setAttribute, a partir desse momento utilizando pizzaJson[index].id. Após isso, basta recuperar o data-key em uma variável, utilizando o getAttribute, então exibir nos campos as informações referente aquela key ou index, se tratando da pizzaJSON.

**6ª aula:**

Feito com que assim que um modal seja fechado, tire a seleção do tamanho da pizza, e a sete de novo para grande, quando o modal for aberto novamente. Para fazer isso, primeiro é selecionado o elemento html que possui a classe selected com classList.remove. Depois, é iniciado um forEach(), nos elementos selecionados com querySelectorAll referente aos 3 tamanhos da pizza, contendo uma função que recebe size e sizeIndex como parâmetro, que adiciona a classe selected ao elemento de index 2 (3º). Nesse forEach, também é passado o valor do peso de cada tamanho, presente em pissaJson[key].sizes[sizeIndex]. Também é criada uma variável global modalQntde com o valor de um, que é setada novamente para 1 assim que um modal é aberto, garantindo que independentemente de qual pizza for aberta, a quantidade voltará para 1.

**7ª aula:**

Criada uma função para fechar o modal que muda a opacidade do modal para 0 (realizando a animação de meio segundo), depois é colocado um setTimeout, alterando o display para none que recebe meio segundo. A função então é executada quando as classes pizzaInfo--cancelButton e pizzaInfo--cancelMobileButton são clicadas.

**8ª aula:**

Adicionado funções aos botões de aumentar e diminuir a quantidade de itens. Quando for para diminuir, há uma verificação que garante que nada será executado caso o valor seja menor que 1. Também feito com que seja possível selecionar o tamanho das pizzas. Basicamente o que é feito é que ao clicar em um tamanho, o que está selecionado atualmente é deselecionado e o que foi clicado passa a ser o selecionado.

**9ª aula:**

Inserção das pizzas no carrinho. Criado um array cart, que vai receber as informações do id, size e quantidade da pizza. Duas dessas informações já estão disponíveis faltando somente o size, que é capturado usando o mesmo seletor para desmarcá-lo, na função anterior, porém dessa vez recuperando o valor armazendo em data-key e colocando-o em uma variável. Após isso, basta realizar um push no array cart com um objeto com as informações necessárias.

**10ª aula:**

Feita a adição de uma função que verifica caso uma pizza, com o mesmo tamanho foi adiciona novamente, fazendo com que seja apenas adicionada a quantidade da pizza e não um novo item inteiro no cart. Para isso, é criado um identificador que consiste de uma string contendo o id separado por um @ no caso e o tamanho da pizza, quando um novo item está prestes a ser adicionado, é verificado se o identificador é o mesmo, caso seja, a realizado somente uma adição na quantidade do item que possui aquele determinado identificador.

**11ª aula:**

Feita a função updateCart, que tem como objetivo verificar se há ou não itens no carrinho. Caso tenha, deve abrir a tela de carrinho e caso não tenha deve fecha-lo.

**12ª aula:**

Feita a recuperação dos itens do cart para serem exibidos no cart, fazendo um for para cada item no array cart, fazendo um processo parecido com as etapas inicias desse projeto. Como no carrinho não tem nenhum campo específico para tamanha, foi realizado um switch para cada um dos tamanhos, indicando o tamanho por um char.

**13ª aula:**

Adicionada a funcionalidade dos botões de adicionar e remover itens do carrinho. Procedimento parecido com o do modal, porém dessa vez, caso um item seja igual a 1, quando for diminuído o mesmo deve ser retirado do carrinho. Também realizada as operações básicas do subtotal, desconto e total, mostrando os elementos na tela.

**14ª aula:**

Criadas as funções referentes a abertura e o fechamento do carrinho para celulares. Feito colocando para o menu ficar com o estilo left= 0vw para aberto e left=100vw para quando o mesmo é fechado. Também adicionado a verificação de quantidade itens no carrinho para setar left para 100vw caso não possua nenhum item no carrinho. Projeto concluído!