- ideia de guardar valores dentro de uma palavra, isto é, uma **variável**. `cosnt idade = 26;`
- `console.log(idade);`
- a convenção de escrita para variáveis compostas, isto é, que contém duas palavras: a primeira palavra com letra minúscula e a segunda com maiúscula. Esta convenção é conhecida como Cammel Case. `idadeSomada`
- Dizemos que linguagens sensíveis a maiúsculas e minúsculas são case sensitives, e é caso do JavaScripit.
- Quando não inserimos a palavra `const` antes da variável, dizemos que ela pode ir para um escopo global.Existem outros tipos de declaração, como o `let`.
- variáveis podem ser letras ou números e deve se ter padrão(se vari começam com minuscula segue para as outras) e preferen const
- o computador sempre executa as operações de divisão e multiplicação antes das somas e subtrações Mas como podemos garantir que a soma seja executada primeiramente? Utilizamos parênteses: `console.log((10 + 8) * 2);`
- `console.log("ano" + 2020);`
- Para realizarmos a conversão de tipos utilizaremos o `paseInt`, este "int" ser refere a um número inteiro. `console.log(parseInt("2") + praseInt("2"));`
- Quando adicionamos um texto a outro, não chamamos essa operação de soma, mas sim de **concatenação**
- No caso de divisões, o comportamento do leitor é outro. `console.log("10"/ "2");`
- `onsole.log("Ricardo" / "2"); NaN`
- `3.5`, um número com casa decimal ou os chamados "pontos flutuantes". Lembrado que se queremos escrever números com casa decimal, devemos sempre utilizar pontos (.) e não vírgulas
- e existe o parseFloat para converter em ponto flutuante;
- ara resolver essa questão poderemos simplesmente adicionar um espaço entre `nome` e o sinal de concatenação `+`, algo como `(nome + " " sobrenome)`ou utilizar a vírgula: `console.log(nome, sobrenome);`
- Podemos declarar nossos textos entre crases. Nas versões mais novas do JavaScript conseguimos fazer a interpolação de variáveis por meio de `${}`: `console.log(`Meu nome é ${nome} ${sobrenome}`);`
- Existem variáveis que devem ter um estado inconstante, e neste caso usamos o `let` para defini-las, como por exemplo um `contador` que varia de números.
- O JavaScript permite que mudemos o tipo das variáveis livremente, mas essa não é uma boa prática, o correto é a atribuição seja mais constante,

uma vez que a mudança de estados na programação é algo complexo que pode desencadiar em muitos erros.

- script é fracamente tipada ela permite que uma variável texto vire numero
- Sobrescrever uma variável significa mudar o conteúdo dela e para isso precisamos atribuir com o = o novo valor que desejamos.
- Sempre que declaramos uma variável estamos reservando um espaço de memória no computador.
- `let idade; // declarando variável`
- `idade = 26; // atribuindo valor`
- Para facilitar e agiliar o processo, existe uma estrutura de dados chamada array, em que podemos armazenar diversos dados. A maneira de declarar um array é um pouco diferente de textos e números: utilizamos uma palavra chave new e o tipo da estrutura Array, e assim podemos criar uma nova lista com nossos destinos:
  `constlistaDeDestinos = new array (`
- `    `Salvador`,`
- `    `São Paulo`,`
- `    `Rio de Janeiro``
-
- `);`
- írgula simboliza um espaço entre as informações
- "Ctrl + K + C" faz todos selecionados virar comentário
- o comando push, que irá adicionar itens dentro de listaDeDestinos:
- `listaDeDestionos.push(`Curitiba`)`
- `tem q ser antes do console.log`
- conseguimos adicionar novos elementos a lista mas atribuir algo diferente a const
- Para remover um elemento da lista utilizamos o splice, um comando que possibilitará a remoção em listaDeDestinos.
- se considera a posição "0" de maneira implícita. Portanto a posição real na cidade de São Paulo na lista é 3.
- `listaDeDestinos.splice(1, 1); posição 1, 1 elemento`
- omo exibir destinos específicos ao nosso cliente? console.log() adicionaremos entre colchetes a posição da cidade`console.log(listaDeDestinos[1]);`
- `console.log(listaDeDestinos[1], listaDeDestinos[0]);`
- Para acessar a documentação basta fazermos uma busca simples no Google pelo termo "array js"
- new para criar um array, trata-se de uma palavra reservada da linguagem, portanto não conseguimos criar uma variável com este nome.
- A palavra que utilizaremos equivale a um "se", o if. Portanto "se" o valor for menor de 18 algo acontece, no caso, um outro bloco de código é

acionado e uma remoção de elemento é feita da lista, simbolizando uma compra. Utilizaremos como separador de cídigo as chaves {}.

- `if(idadeComprador >= 18){`
- `console.log("Comprador maior de idade");`
- `listaDeDestinos.splice(1, 1);`
- `}`

- ara manter a organização do código no caso das duas situações (comprador maior e menor de idade), utilizaremos a palavra `else`, isto é, o que realizar quando a proposição de `if` não for verdadeira:
`if(maiorDeIdade >= 18){`
- `console.log("Comprador maior de idade");`
- `listaDeDestinos.splice(1, 1);`
- `}else{`
- `    console.log("Não é maior de idade e não posso vender");`
- `}`
- tilizamos os operadores lógicos `=` e `>`, mas existem diversas possibilidades de configuração:
- `>18`
- `>18`
- `<=18`
- `>18`
- `==18`

- o **booleano**, que representa valores verdadeiros ou falsos. Portanto se `estaAcompanhada` for verdadeiro, a compra poderá ser efetuada. `const estaAcompanhada  = true;`
- `if (idadeComprador >= 18) {`
- `console.log("Comprador maior de idadade");`
- `listaDeDestinos.splice(1, 1); // removendo item`
- `}   else if (estaAcompanhada == true) {`
- `console.log("Comprador está acompanhado");`
- `listaDeDestinos.splice(1, 1); //removedo item`
- `}else {`
- `    console.log("Não é maior de idade e não posso vender");`
- `}`
- `}`

- Para escrevermos algo como "ou" está acompanhado ou é maior de idade para fazer compra de pacotes, utilizamos `||`no JavaScrip: `if (idadeComprador >= 18 || estaAcompanhada == true) {`
- `console.log("Comprador maior de idade");`

- O \n é um caractere especial que possibilita que pulemos uma linha no momento de imprimir a mensagem de texto. Temos desde o embarque uma verificação para saber se de fato o cliente pode embarcar, e esta é uma maneira de adicionar mais condicionais dentro de um único if.
- `console.log("Embarque: \n\n")`
- `if(idadeComprador >= 18 && temPassagemComprada)`
- `    console.log("Boa viagem);`
- `}else{`
- `    console.log("Você não pode embarcar");`
- `}`

- Shift + Alt + F. para formatar código
- Os operadores lógicos devem ter no lado esquerdo e direito uma expressão booleana. `if (idade > 18 && idade < 65)`
- `{`
-
- `}`
- Deveremos mapear a nossa lista, e para isso precisaremos de um contador para elencarmos a posição dos itens. A variável `contador` se iniciará com o valor `0`, o elemento inicial da lista. Enquanto o contador for menor do que o tamanho da lista (3 elementos) faremos a impressão do item apontado na lista. Se não queremos que o contador tenha esse comportamento, precisamos coletar o valor dentro de `contador` e somar `1` e realizar uma reatribuição. Ao executarmos novamente o código teremos: O nome de todas as cidades está sendo exibido, mas queremos exibir apenas aquela a ser verificada para a compra. Então queremos criar uma condição de visualização. Portanto escrever
- `let contador = 0`
- `while(contador<3){`
- `if(listaDeDestinos[contador] == destino){`
- `console.log("Destino existe")`
- `    }else{`
- `console.log("Destino não existe");`
- `    }`
- `    contador += 1;`
- `}`

- while :quer dizer enquanto
- Na expressão condicional do `while` é possível utilizar qualquer operador de comparação (`< [menor]`, `> [maior]`, `<= [menor ou igual]`, `>= [maior ou igual]`, `== [igual a]` e `!= [diferente de]`) e qualquer operador lógico (`&& [and]`, `|| [ou]`).
- breakpoint é o ponto de parada de execução do código(bolinha vermelha do lado do numero da linha é preciso usar o modo debug , lá em variáveis vao ter global(que todos podem usar) e local(que são as que estou usando)
- No momento em chegarmos ao `while()`, configuraremos para que ao encontrar o elemento de buscado, a

execução deve ser interrompida.Para isso, utilizaremos o comando `break`

- `...destinoExiste = true;`
- `break;`
- Exatamente, ao colocarmos um `break` dentro do laço estamos falando para o interpretador que quando ele chegar nessa linha ele deve sair do laço independentemente de outras condições.
- `while` é um dos mais simples que existem e tranquilos de trabalhar, pois uma vez que determinada condição for verdadeira, um bloco de código específico será executado:
- Outro laço muito comum que encontraremos bastante no dia a dia da programação é o `for`. A ideia é a mesma: será executado em looping alguma operação, mas ele apresenta um formato de escrita mais complexo.
- Para o caso do `for`, a contagem deverá ser dividida em três partes: precisaremos inicializar o contador , colocar sua condição (no casao, < 3) e por fim o comando que deverá ser executado ao final do loop, neste caso para evitar o loop infinito, somaremos + 1 np contador. Poder[iamos escrever `cont += 1`, mas a forma mais comum de se encontrar no `for` é `cont++`.
- `for(let cont = 0 ; cont <3 ; cont++){`
- `if(listaDeDestinos[contador] == destino){`
- `destinoExiste = true;`
- `}`
- tipo de operação é utilizar o nome `i` como auxiliar do `for`:
- `for(let i = 0 ; i <3 ; i++){`
- `if(listaDeDestinos[contador] == destino){`
- `destinoExiste = true;`
- `}`
-