JavaScript Básico

- Material das Aulas
- Introdução ao Javascript e Variáveis
  - Documentação
  - O que é?
  - DevTools
- var
  - exibindo no console com `console.log`
  - entrada de texto com `prompt`
- Números e operadores
  - Documentação
  - Operadores.js
  - Com Strings
  - NaN
- Boas práticas em JavaScript
  - Material Complementar
  - camelCase
  - Boas práticas
- Condicionais
  - O que é?
  - If...else
  - If...Else If... else (aninhado)
  - Simplificando If..else aninhandos
  - Switch case
- Estrutura de Repetição: For
  - O que é?
  - Exemplo de Tabuada
  - Condicional e Repetição
- Estrutura de Repetição: While
  - O que é?
  - Exemplo de Loop Infinito
  - Exemplo funcional
- Funções
  - Para que serve `function soma(a,b)`
  - Exemplo

Classes
O que são?
Instanciando a classe com `new Class`

Material das Aulas

- O que é JavaScript?

Introdução ao Javascript e Variáveis

Documentação

- MDN Mozilla

O que é?

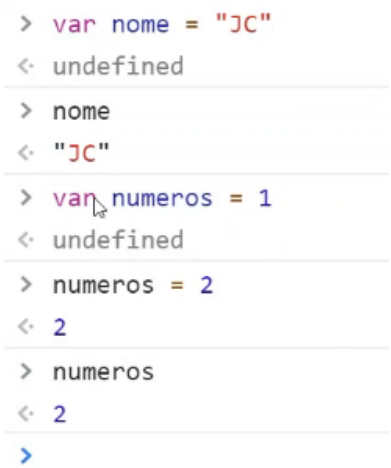
- Linguagem da Web
- Vamos começar a codar e essa será nossa dinâmica

DevTools

- **F12, barra lateral**
- Usaremos o console do Browser nessa primeira aula.

var

- Armazena valores para usarmos quando quiser



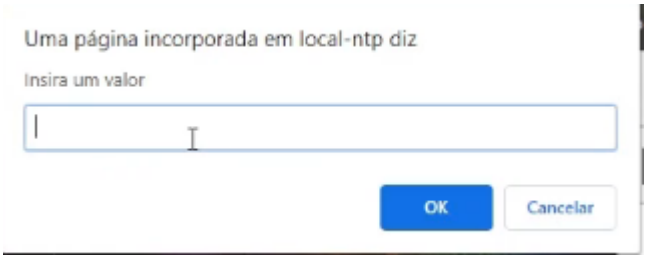
exibindo no console com `console.log`

- Função do javascript que mostra saídas do nosso código
- Juntaremos as frases com `+`

```
> var todos = "valores armazenados"
<- undefined
> console.log("a frase armazenada é" + todos)
  a frase armazenada évalores armazenados  VM2077:1
<- undefined
>
```

entrada de texto com `prompt`

- Usamos quando esperamos um Input do usuário



- Valor armazenado em memória e o console exibe no console do JS

```
> var entrada = prompt("Insira um valor")
<- undefined
> console.log(entrada)
  10                                    VM2244:1
<- undefined
```

Números e operadores

Documentação

- MDN Mozilla

Operadores.js

- Arquivo que pode ser interpretado por um motor JS

```
var operador = 100;
var operando = 50;

var resultado = operador * operando;

console.log(resultado);
```

- No debug console do VSCode temos o resultado.

Com Strings

- Colocar espaço para separar as varávies

1ª Alternativa

```
var nome = 'JC';
var sobrenome = 'Bombardelli';

var nomeCompleto = nome + sobrenome;

console.log(nomeCompleto);
```

```
JCBombardelli
```

2ª alternativa

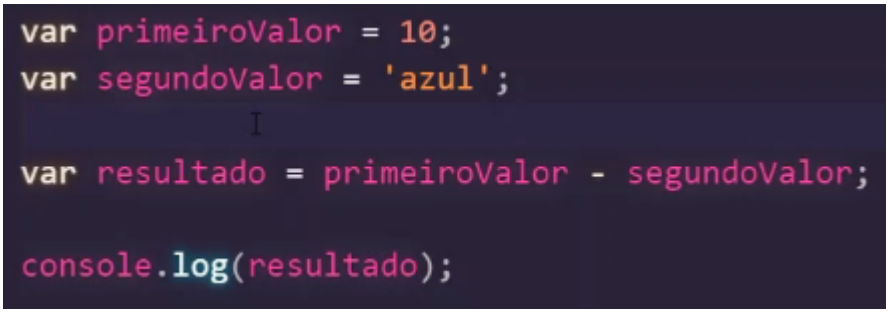
```
var nome = 'JC';
var sobrenome = 'Bombardelli';

var nomeCompleto = nome + ' ' + sobrenome;

console.log(nomeCompleto);
```

NaN

- Motor do JS não consegue fazer operações aritméticas entre números e textos
- No entanto se trocarmos para `'3'` ele faz a correção de tipo e soma



Boas práticas em JavaScript

Material Complementar

- Dev Media

camelCase

- Para aplicarmos nos nomes de variáveis usamos esse padrão

Boas práticas

- Todas variáveis devem começar com uma letra.
- Sempre colocar espaço entre os operadores e depois da vírgula
- Descrever sua variável em inglês e de forma clara no que ela faz.
- Para identação use sempre dois espaços
- Sempre termine uma instrução simples com ponto e vírgula.
- Sempre coloque a **abertura da chave** na mesma linha da função ou de um objeto
  - Use um espaço entre a declaração da função e a abertura da chave
  - Coloque a chave final da função em uma nova linha isolada
- Cada linha de código não deve ultrapassar muitos caracteres, use **Prettier**
- Sempre utilize arquivos externos para a sua página HTML invocar JavaScript
- Nomes de arquivos evem ser sempre com **caractéres minúsculas.**

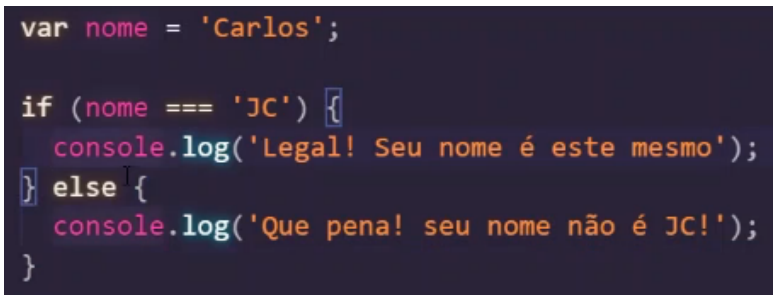
Condicionais

O que é?

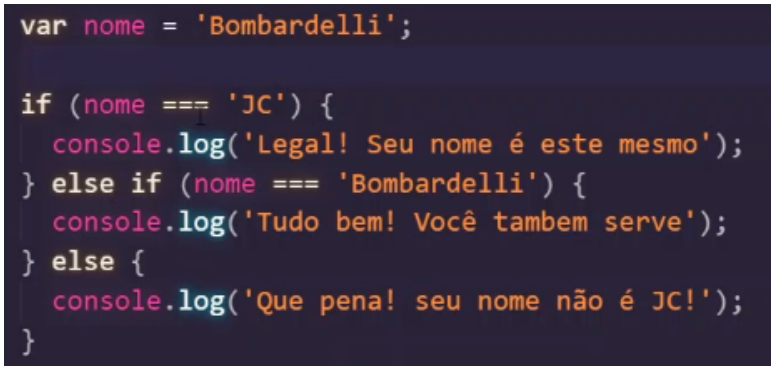
- Tomar uma decisão sobre uma pergunta ou uma informação que chegou
- Código lido de linha a linha e as condicionais podem mudar esse fluxo

If...else

- = é atribuir um valor a uma variável
- === comparação de tipo e valor.



If...Else If... else (aninhado)



Simplificando If..else aninhandos

- Fica mais legível.

```
var nome = 'Pedro'

if (nome === 'Peter') {
  console.log('oi')
}
if (nome === 'Pitrv') {
  console.log('oi')
}
if (nome === 'Pedro') {
  console.log('oi')
}
```

Switch case

- Tornar o código enxuto simplificando uma cadeia de If else.
- Default é executada sempre que nenhum dos casos testados retornar verdadeiro.

```
var nome = 'JC';

switch (nome) {
  case 'JC':
    console.log('Legal! Você é o JC mesmo!');
    break;
  case 'Bombardelli':
    console.loh('Ah! Você tambem serve!');
    break;
  default:
    console.log('Que Pena! Você não é quem eu estou procurando');
    break;
}
```

Estrutura de Repetição: For

O que é?

- Serve para percorrer valores dentro de uma estrutura grande.
- É uma **laço de repetição,** vai executar o que estiver dentro até a condição de parada ser atingida.
- O valor i é chamado de iterador ou incrementador, serve para o controle do For apenas.
- Os parâmetros são
  1. Declaração de um valor inicial
  2. Uma condição de parada
  3. Incremento

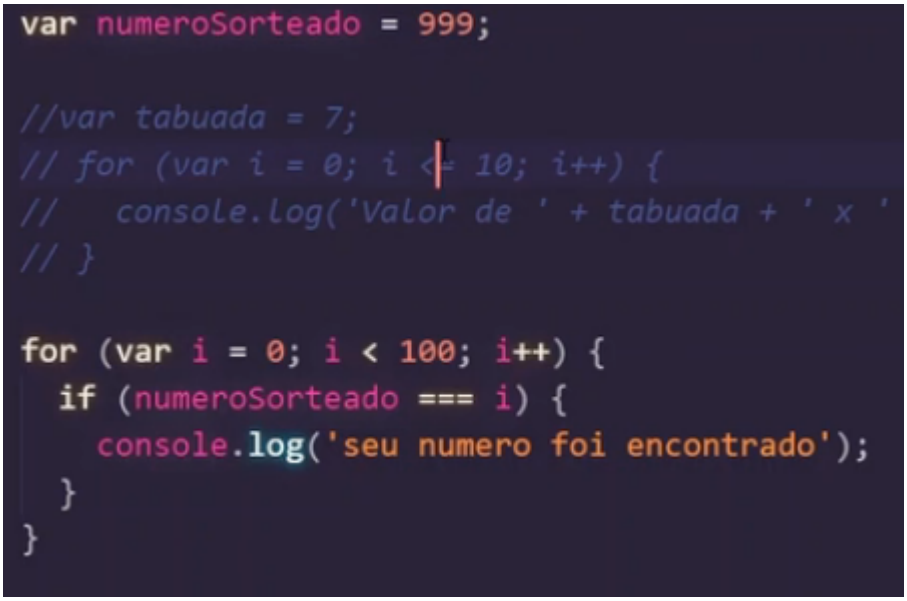
Exemplo de Tabuada

```
var tabuada = 7;

for (var i = 0; i <= 10; i++) {
  console.log('Valor de ' + tabuada + ' x ' + i + ' = ' + tabuada * i);
}
```

Condicional e Repetição

- Vamos fazer um exemplo para sortear um número



```
var numeroSorteado = 999;

//var tabuada = 7;
// for (var i = 0; i <= 10; i++) {
//    console.log('Valor de ' + tabuada + ' x '
// }

for (var i = 0; i < 100; i++) {
  if (numeroSorteado === i) {
    console.log('seu numero foi encontrado');
  }
}
```

Estrutura de Repetição: While

O que é?

- É uma estrutura de repetição que significa Enquanto
- Enquanto satisfazer a condição, ou seja for **verdadeiro**, vai repetir o código dentro.
- **Não há condição de parada,** diferente do **for**
- Ficará em LOOP infinito, até a condicional for **false**

Exemplo de Loop Infinito

- **!Variável**
    - Significa que estamos fazendo a negação desse valor, se for true vira false, se false vira true.

```
var achou = false;

while (!achou) {
  console.log('achou');
}
```

Exemplo funcional

- Temos que fazer uma variável de incrementação na mão, diferente do for, que já temos ali disponível.
- Criamos uma flag para quando sairmos dele, mudando o valor de uma variável para `false.`

```
var numeroSorteado = 10;
var possivelValor = 0;

while (!achou) {
  possivelValor += 1;
  if (numeroSorteado === possivelValor) {
    achou = true;
  } else {
    console.log(
      'Possivel valor não corresponde ao numero sorteado ' + possivelValo
```

Funções

Para que serve

- Serve para armazenar um bloco de código que pode ser usado e reutilizado quantas vezes quisermos

`function soma(a,b)`

- Passamos dentro da função dela **parâmetros**
- Chamamos ou invocamos a função passando **argumentos**
- Armazenamos numa variável e mostramos no `console.log`

```
function soma(operadorA, operadorB) {
  var resultadoC = operadorA + operadorB;
  return resultadoC;
}

var resultadoDaSoma = soma(1, 2);
console.log(resultadoDaSoma);
```

Exemplo

- Não armazenamos em nenhum lugar, só invocamos a função.

```
function olaGama(nome) {
  console.log('Olá Gama! você é o ' + nome);
}
```

```
olaGama('JC');
```

Classes

O que são?

- Muito comum no POO, mas no JS não é uma obrigatoriedade
- A comunidade usa muito mais Objetos e Funções, já que praticamente todas as estruturas de dados complexas, como Array, Função, Classe são **Objetos.**
- Classes em JS provêm uma maneira mais simples e clara de **criar objetos lidar com heranças**
- Podemos armazenar qualquer tipo de dado nas classes.
- Usamos a **notação ponto,** de objetos para usarmos métodos definidos dentro das classes.

Instanciando a classe com `new Class`

- Estamos criando um objeto em memória, que passa a ser um agregado de valores e propriedades dentro do JS

```
class Matematica {
  soma(valorA, valorB) {
    return valorA + valorB;
  }
  subtracao(valorA, valorB) {
    return valorA - valorB;
  }
}

var instanciaMatematica = new Matematica();
```

```
var resultado = instanciaMatematica.soma(4, 7);

console.log(resultado);
```