Desenvolvimento de Softwares - Apostila

Conteúdos: tipos de dados, variáveis, constantes, expressões, conversão de tipos, operadores lógicos e estruturas de seleção.

Aula 1 – Introdução aos Tipos de Dados

Tipos de dados representam a natureza das informações armazenadas em um programa. Em JavaScript, os principais tipos são Number, String, Boolean, Undefined e Null. Os tipos de dados permitem que o computador saiba como tratar cada informação, realizando cálculos, comparações ou manipulação de textos. O uso correto dos tipos reduz erros e melhora a organização do código. Exemplos práticos incluem idade (Number), nome (String) e situação de aprovação (Boolean).

Exercícios:

1. O que é um tipo de dado?
2. Cite três tipos de dados.
3. Explique a diferença entre String e Number.

Aula 2 – Variáveis e Armazenamento de Dados

Variáveis são espaços reservados na memória para armazenar informações. Elas permitem que os dados sejam alterados durante a execução do programa. Nomes de variáveis devem ser claros e representar seu conteúdo. O programador utiliza variáveis para guardar valores digitados pelo usuário, resultados de cálculos e informações temporárias.

Exercícios:

4. Defina variável.
5. Crie cinco exemplos de variáveis.
6. Por que nomes significativos são importantes?

Aula 3 – Declaração e Inicialização de Variáveis

Declarar uma variável significa informar sua existência ao programa. Inicializar significa atribuir um valor inicial. Uma variável pode ser declarada sem valor e receber conteúdo posteriormente. Entender esse processo é essencial para controlar corretamente os dados utilizados pelo sistema.

Exercícios:

7. Diferencie declaração e inicialização.
8. Crie exemplos dos dois casos.
9. Explique por que variáveis podem mudar de valor.

Aula 4 – Constantes

Constantes armazenam valores que não devem ser alterados durante a execução do programa. São úteis para taxas, valores fixos, limites e configurações do sistema. O uso de constantes melhora a manutenção e a segurança do código.

Exercícios:

10. O que diferencia constante e variável?
11. Crie três exemplos de constantes.
12. Quando utilizar constantes?

Aula 5 – Operadores Aritméticos

Operadores aritméticos permitem realizar cálculos matemáticos como soma, subtração, multiplicação, divisão, potência e resto da divisão. São utilizados em praticamente todos os sistemas computacionais, desde calculadoras até sistemas financeiros.

Exercícios:

13. Resolva operações básicas.
14. Explique a função do operador %.

Aula 6 – Expressões Aritméticas

Uma expressão é formada por valores e operadores. A ordem de precedência influencia diretamente o resultado. Parênteses são utilizados para controlar a ordem das operações e tornar o código mais legível.

Exercícios:

15. Resolva expressões propostas.
16. Explique a importância dos parênteses.

Aula 7 – Operadores Relacionais

Operadores relacionais realizam comparações entre valores. O resultado será sempre verdadeiro ou falso. Essas comparações são fundamentais para a tomada de decisões em programas.

Exercícios:

17. Avalie comparações.
18. Explique o resultado de cada expressão.

Aula 8 – Conversão de Tipos

A conversão de tipos permite transformar um dado de um formato para outro. Pode ocorrer automaticamente (implícita) ou por ação do programador (explícita). Esse recurso é muito utilizado na entrada de dados e integração entre sistemas.

Exercícios:

19. Diferencie conversão implícita e explícita.
20. Converta texto em número e vice-versa.

Aula 9 – Operadores Lógicos

Operadores lógicos combinam condições para criar regras mais complexas. Os principais são AND, OR e NOT. Eles são utilizados em autenticação, permissões de acesso e validações diversas.

Exercícios:

21. Explique AND, OR e NOT.
22. Crie exemplos de uso.

Aula 10 – Avaliação de Expressões Lógicas

Expressões lógicas são utilizadas para verificar condições e determinar caminhos diferentes para execução do programa. A compreensão dessas expressões é a base para estruturas de seleção e repetição.

Exercícios:

23. Avalie expressões lógicas.
24. Crie exemplos relacionados ao ambiente escolar.

Desenvolvimento de Softwares - Apostila (Aulas 11 a 25)

Aula 11 – Estrutura de Seleção if

A estrutura if permite executar comandos apenas quando uma condição é verdadeira.
Exemplo em JavaScript:

```
if (idade >= 18) {
   console.log("Maior de idade");
}
```

O if é utilizado em validações, autenticação de usuários e diversas regras de negócio.

Exercícios:

Crie um programa que informe se um aluno foi aprovado com nota maior ou igual a 60.

Explique a função do if.

Aula 12 – Estrutura if...else

O if...else permite escolher entre dois caminhos possíveis.

Exemplo:

```
let nota = 50;

if (nota >= 60) {
   console.log("Aprovado");
} else {
   console.log("Reprovado");
}
```

Essa estrutura é utilizada quando sempre haverá duas possibilidades.

Exercícios:

Crie um programa para verificar se uma pessoa é maior ou menor de idade.

Aula 13 – Estruturas Encadeadas

Estruturas encadeadas permitem analisar várias condições.

Exemplo:

```
if (nota >= 80) {
   console.log("Ótimo");
} else if (nota >= 60) {
   console.log("Bom");
} else {
   console.log("Reprovado");
}
```

Muito utilizadas em sistemas de classificação.

Exercícios:

Crie uma classificação para notas de 0 a 100.

Aula 14 – Operador Ternário

O operador ternário simplifica decisões simples.

Exemplo:

```
let resultado = idade >= 18 ? "Maior" : "Menor";
```

Ele substitui estruturas if simples e torna o código mais compacto.

Exercícios:

Transforme um if simples em operador ternário.

Aula 15 – Estrutura switch

O switch é utilizado quando existem diversas opções fixas.

Exemplo:

```
let dia = 2;

switch(dia){
case 1:
 console.log("Domingo");
 break;
case 2:
 console.log("Segunda");
 break;
}
```

Muito utilizado em menus e sistemas de opções.

Exercícios:

Crie um switch para os dias da semana.

Aula 16 – Exercícios de Estruturas de Seleção

Revisão prática envolvendo if, else, else if, switch e operador ternário.

Exercícios:

Resolver 10 exercícios propostos pelo professor.

Aula 17 – Introdução aos Laços de Repetição

Laços permitem repetir comandos várias vezes sem duplicar código.

Principais estruturas:
- while
- do while
- for

São fundamentais em sistemas que processam listas e grandes quantidades de dados.

Exercícios:

Explique a vantagem dos laços de repetição.

Aula 18 – Estrutura while

O while executa enquanto a condição for verdadeira.

Exemplo:

```
let contador = 1;

while(contador <= 5){
 console.log(contador);
 contador++;
}
```

Ideal quando não sabemos exatamente quantas repetições serão necessárias.

Exercícios:

Crie um contador de 1 a 10 usando while.

Aula 19 – Estrutura do while

O do while executa pelo menos uma vez.

Exemplo:

```
let numero = 1;

do{
 console.log(numero);
 numero++;
} while(numero <= 5);
```

Muito utilizado em menus de sistemas.

Exercícios:

Crie um exemplo utilizando do while.

Aula 20 – Estrutura for

O for é a estrutura mais utilizada para repetições conhecidas.

Exemplo:

```
for(let i = 1; i <= 10; i++){
 console.log(i);
}
```

Muito utilizada para percorrer vetores e listas.

Exercícios:

Crie uma contagem de 1 até 20 usando for.

Aula 21 – Contadores e Acumuladores

Contadores registram quantidades.
Acumuladores somam valores.

Exemplo:

```
let soma = 0;

for(let i=1;i<=5;i++){
 soma += i;
}
```

Resultado: 15.

Exercícios:

Some os números de 1 até 100.

Aula 22 – Repetições Crescentes

Executam valores em ordem crescente.

Exemplo:

```
for(let i=0;i<=50;i++){
 console.log(i);
}
```

Muito utilizadas em relatórios e listagens.

Exercícios:

Mostre números pares de 0 a 100.

Aula 23 – Repetições Decrescentes

Executam valores em ordem decrescente.

Exemplo:

```
for(let i=100;i>=0;i--){
 console.log(i);
}
```

Muito utilizadas em contagens regressivas.

Exercícios:

Faça uma contagem regressiva de 50 até 0.

Aula 24 – Laços Aninhados

Laços aninhados são repetições dentro de outras repetições.

Exemplo:

```
for(let i=1;i<=3;i++){
  for(let j=1;j<=3;j++){
    console.log(i,j);
  }
}
```

Utilizados em tabelas e matrizes.

Exercícios:

Crie uma tabuada utilizando laços aninhados.

Aula 25 – Comandos break e continue

break encerra o laço.
continue ignora uma iteração.

Exemplo:

```
for(let i=1;i<=10;i++){
 if(i==5) break;
}

for(let i=1;i<=10;i++){
 if(i==5) continue;
}
```

São utilizados para controle avançado de fluxo.

Exercícios:

Crie exemplos utilizando break e continue.