### Variável

Uma variável é um local na memória de um programa de computador onde você pode armazenar dados temporariamente. As variáveis têm um nome e um tipo de dado associado que determina que tipo de informações podem ser armazenadas nelas.

HTML (`variavel.html`):

```html

<!DOCTYPE html>

<html>

<head>

<title>Variável</title>

</head>

<body>

<script>

// Declaração de variável

let idade = 25;

console.log(idade); // Saída: 25

</script>

</body>

</html>

```

### Tipo de Dado

O tipo de dado é uma classificação que determina que tipo de valor uma variável pode armazenar. Alguns exemplos comuns de tipos de dados incluem números inteiros (int), números de ponto flutuante (float), caracteres (char), booleanos (bool), entre outros.

HTML (`tipodedado.html`):

```html

<!DOCTYPE html>

<html>

<head>

<title>Tipo de Dado</title>

</head>

<body>

<script>

let nome = "Maria"; // String

let ativo = true; // Boolean

let numero = 42; // Number

</script>

</body>

</html>

```

### Vetor (Array)

Um vetor é uma estrutura de dados que permite armazenar uma coleção de elementos do mesmo tipo. Os elementos são acessados através de um índice. Vetores são usados para armazenar informações de forma organizada, como uma lista de números, nomes, etc.

HTML (`vetor.html`):

```html

<!DOCTYPE html>

<html>

<head>

<title>Vetor</title>

</head>

<body>

<script>

let frutas = ["maçã", "banana", "laranja"];

console.log(frutas); // Saída: ["maçã", "banana", "laranja"]

</script>

</body>

</html>

```

### Índice

Um índice é um número que identifica a posição de um elemento em uma estrutura de dados, como um vetor ou uma lista. Os índices geralmente começam em 0 (para a primeira posição) e aumentam sequencialmente. Por exemplo, o primeiro elemento de um vetor tem índice 0, o segundo tem índice 1, e assim por diante.

HTML (`indice.html`):

```html

<!DOCTYPE html>

<html>

<head>

<title>Índice</title>

</head>

<body>

<script>

let frutas = ["maçã", "banana", "laranja"];

console.log(frutas[1]); // Saída: "banana" (índice 1)

</script>

</body>

</html>

```

### Operadores Lógicos e Relacionais

Operadores lógicos são usados para realizar operações lógicas em valores booleanos (verdadeiro ou falso), como "E" lógico (&&), "OU" lógico (||) e "NÃO" lógico (!). Operadores relacionais são usados para comparar valores, como igual (==), diferente (!=), maior que (>), menor que (<), maior ou igual (>=) e menor ou igual (<=).

HTML (`operadores.html`):

```html

<!DOCTYPE html>

<html>

<head>

<title>Operadores Lógicos e Relacionais</title>

</head>

<body>

<script>

let idade = 25;

let maiorDeIdade = idade >= 18;

console.log(maiorDeIdade); // Saída: true

</script>

</body>

</html>

```

### IF

O "if" é uma estrutura de controle de fluxo em programação que permite executar um bloco de código se uma condição especificada for verdadeira. Ele também pode ser usado em conjunto com "else" para executar um bloco de código alternativo se a condição for falsa.

HTML (`if.html`):

```html

<!DOCTYPE html>

<html>

<head>

<title>IF</title>

</head>

<body>

<script>

let temperatura = 30;

if (temperatura > 25) {

console.log("Está quente!");

} else {

console.log("Não está tão quente.");

}

</script>

</body>

</html>

```

### FOR

O "for" é uma estrutura de repetição em programação que permite executar um bloco de código várias vezes, com base em uma condição e um contador. É frequentemente usado para iterar através de uma coleção de elementos, como um vetor, de forma controlada.

HTML (`for.html`):

```html

<!DOCTYPE html>

<html>

<head>

<title>FOR</title>

</head>

<body>

<script>

for (let i = 0; i < 5; i++) {

console.log("Número: " + i);

}

</script>

</body>

</html>

```

### WHILE

O "while" é outra estrutura de repetição em programação que permite executar um bloco de código repetidamente enquanto uma condição especificada for verdadeira. Ele é usado quando não se sabe previamente quantas vezes o código precisa ser repetido.

HTML (`while.html`):

```html

<!DOCTYPE html>

<html>

<head>

<title>WHILE</title>

</head>

<body>

<script>

let contador = 0;

while (contador < 3) {

console.log("Contador: " + contador);

contador++;

}

</script>

</body>

</html>

```

### Evento Click

Um evento é uma ocorrência específica em um programa de computador que pode acionar a execução de um código específico. Os eventos podem ser acionados pelo usuário (como clicar em um botão) ou por outras partes do programa (como receber um dado de entrada). Os eventos são frequentemente usados em programação orientada a eventos para controlar a interação do programa com o usuário.

HTML (`evento\_click.html`):

```html

<!DOCTYPE html>

<html>

<head>

<title>Evento Click</title>

</head>

<body>

<button id="meuBotao">Clique em mim</button>

<script>

let meuBotao = document.getElementById("meuBotao");

meuBotao.addEventListener("click", function() {

alert("Botão clicado!");

});

</script>

</body>

</html>

```

### Evento Keydown, Keyup e Keypress

HTML (`evento\_teclado.html`):

```html

<!DOCTYPE html>

<html>

<head>

<title>Eventos de Teclado</title>

</head>

<body>

<input type="text" id="meuCampoDeTexto" placeholder="Pressione uma tecla">

<script>

let campoDeTexto = document.getElementById("meuCampoDeTexto");

campoDeTexto.addEventListener("keydown", function(event) {

console.log("Tecla pressionada: " + event.key);

});

campoDeTexto.addEventListener("keyup", function(event) {

console.log("Tecla liberada: " + event.key);

});

campoDeTexto.addEventListener("keypress", function(event) {

console.log("Tecla mantida pressionada: " + event.key);

});

</script>

</body>

</html>

```

### Evento Mouseover e Mouseout

HTML (`evento\_mouse.html`):

```html

<!DOCTYPE html>

<html>

<head>

<title>Eventos de Mouse</title>

<style>

.minhaDiv {

width: 200px;

height: 100px;

background-color: white;

}

</style>

</head>

<body>

<div id="minhaDiv" class="minhaDiv"></div>

<script>

let elemento = document.getElementById("minhaDiv");

elemento.addEventListener("mouseover", function() {

elemento.style.backgroundColor = "lightblue";

});

elemento.addEventListener("mouseout", function() {

elemento.style.backgroundColor = "white";

});

</script>

</body>

</html>

```

### Evento Focus e Blur

HTML (`evento\_foco.html`):

```html

<!DOCTYPE html>

<html>

<head>

<title>Eventos de Foco</title>

</head>

<body>

<input type="text" id="meuCampo" placeholder="Digite algo">

<script>

let meuCampo = document.getElementById("meuCampo");

meuCampo.addEventListener("focus", function() {

console.log("Campo recebeu foco.");

});

meuCampo.addEventListener("blur", function() {

console.log("Campo perdeu foco.");

});

</script>

</body>

</html>

```

### Evento DOMContentLoaded

HTML (`evento\_dom.html`):

```html

<!DOCTYPE html>

<html>

<head>

<title>Evento DOMContentLoaded</title>

</head>

<body>

<h1>Conteúdo da página</h1>

<script>

document.addEventListener("DOMContentLoaded", function() {

alert("O documento HTML foi totalmente carregado!");

});

</script>

</body>

</html>

```