Condições em JavaScript

Tipos de Condição:

Condição Simples, quando tem apenas a estrutura do caso verdadeiro.

```
if (condição) {
        True
}
```

Condição Composta

```
if (condição){
        True
} else{
        False
}
```

** Comando novo na aula **

```
console.log('O console funcionou corretamente.')
// Esse comando mostra uma mensagem no console JavaScript com Node
```

Condição aninhada é uma condição dentro da outra.

```
if (cond1) {

} else {
        if (cond2){

        } else {

        }
}
```

** Comando novo na aula **

```
var agora = new Date()
var hora = agora.getHours()
// Esse comando vai pegar e armazenar a hora atual do sistema
```

Condição múltipla

```
switch (expressão) {
        case valor1:
                bloco
                break
        case valor2:
                bloco
                break
        case valor3:
                bloco
```

```
            break
        default:
            bloco
            break
    }
```

A condição Switch vai testar todos os cases na ordem que são colocados, se nenhum deles for correspondente, ele executa o default, que funciona como o else do if.

Dentro de cada bloco dentro do switch, é necessário colocar um comando break.

```
var agora = new Date()
var diaSem = agora.getDay() // Pega o dia da semana
/*
    Em JavaSript, o retorno do dia da semana é numérico
    0 - Domingo
    1 - Segunda
    2 - Terça
    3 - Quarta
    4 - Quinta
    5 - Sexta
    6 - Sábado
*/

switch(diaSem) {
    case 0:
        console.log('Domingo')
        break
    case 1:
        console.log('Segunda')
        break
    case 2:
        console.log('Terça')
        break
    case 3:
        console.log('Quarta')
        break
    case 4:
        console.log('Quinta')
        break
    case 5:
        console.log('Sexta')
        break
    case 6:
        console.log('Sábado')
```

```
        break
    default:
        console.log('To bem não rogerin')
        break

}
```

Repetições

```
while (condição) {
        comandos
}
```

Estrutura de repetição com teste lógico no início.

```
do {
        comando
} while (condição)
```

Estrutura de repetição com teste lógico no final. Primeiro executa o código, e se a condição for verdadeira, executa de novo.

```
for(inicio; teste; incremento) {
        comando
}
```

```
for (c =1; c<=10; c++) {

}
```

Estrutura de Repetição com variável de controle.

Variáveis Compostas

Variáveis simples - só conseguem armazenar um valor por vez.
Variáveis compostas - são capazes de armazenar vários valores em uma mesma estrutura.

A variável composta em JavaScript é o array (vetor), e é composto de elementos, conteúdo para cada elemento e índices que iniciam de 0 para identificar os elementos.

```
let num = [5, 8, 4]
```

Para adicionar um novo valor nesse vetor de 3 espaços, com índices 0, 1, 2, usamos:

```
num[3] = 6
```

O vetor agora é [5, 8, 4, 6]
O comando para adicionar um valor explicitamente na última posição, criando um novo espaço, é: num.push[7]
O vetor agora é [5, 8, 4, 6, 7]
Para sabermos o comprimento de um array utilizamos: num.length
Para organizar os elementos do array na ordem crescente utilizamos: num.sort()

O comando num.indexOf(7) vai retornar a posição do valor especificado dentro dos parênteses, nesse caso, o 7, que está na posição 4. Se o valor especificado dentro dos parênteses não existir dentro do vetor, o JavaScript retorna -1.

Funções

As funções têm chamadas, e podem ter parâmetros, têm ações e podem possuir um retorno. Elas são ações executadas assim que são chamadas ou em decorrência de algum evento. Uma função pode receber parâmetros e retornar um resultado.

Próximos Passos

Fazer curso de HTML5/CSS.
Estudar muito sobre functions.
Objetos.
Modularização.
Expressões regulares (RegEx).
JSON.
AJAX.
NodeJS.

Introdução e fundamentos sobre objetos.

Declarar um objeto é semelhante a declarar um array, mas, ao invés de usar colchetes, usamos chaves.

```
let amigo = {nome:'José', sexo:'M', peso:85.4, engordar(p){}}
```

amigo

José	M	85.4	[function]
nome	sexo	peso	engordar()