Estruturas

É bastante importante entender as estruturas de programação comuns na maioria das linguagens, pois entendendo o conceito de cada uma fica muito mais tranquilo de aplicar esse conhecimento em qualquer linguagem de programação que seja.

Arrays

Os arrays funcionam como listas de dados/informações, vamos dizer lista de itens, cada item dentro do array possui uma posição, ou como dizemos, um índice, e esse índice começa a ser contado de **0** até **N** que é o número de itens dentro de um array, veja um exemplo de array:

```
var itens = ['item 1', 'item 2', 'item 3']
```

Observe que usamos **[]** para o array, e dentro dele colocamos itens separados por vírgula, e cada item é um pequeno texto, e cada item possui uma posição, então esse array do exemplo tem as seguintes posições: **0**, **1** e **2**.

Os arrays podem abrigar dados de vários tipos, mas em quase todas as vezes você vai usar arrays com apenas um tipo de dado, por exemplo, uma lista de nomes → serão todos textos, uma lista de números → apenas números, e assim por diante.

Para você ter uma ideia melhor de um array veja o exemplo abaixo onde transformamos um array de nomes em uma pequena tabela:

```
var nomes = ["Lucas", "Otávio", "Gabriel", "Claudemir", "Marcela", "Lívia" ]
```

Nosso array de nomes acima representado de forma mais visual como uma tabela:

# Posição	Aa Item
0	Lucas
1	Otávio
2	Gabriel
3	Claudemir
4	Marcela
5	Lívia

Os arrays são amplamente utilizados em diversos tipos de aplicações, vamos pensar em algumas aplicações famosas e onde elas provavelmente utilizam arrays:

iFood: apresenta uma lista de restaurantes da sua região;

Mercado livre: apresenta uma lista de produtos de acordo com o que você pesquisa;

Netflix: apresenta uma lista de filmes e séries para você escolher;

Facebook: você visualiza uma lista de amigos em seu perfil;

Whatsapp: você manda mensagem para alguém que está em uma lista de contatos;

> Note que usamos a palavra **lista** pois um array funciona como tal.

Como podemos acessar um item específico dentro de um array? Através do índice do item em questão, veja por exemplo, se eu quiser acessar o nome "Gabriel" dentro do meu array de nomes:

```
var nomes = ["Lucas", "Otávio", "Gabriel", "Claudemir", "Marcela", "Lívia" ]

nomes[2]
```

nomes[2] → "Gabriel", pois "Gabriel" se encontra na posição **2** do meu array de nomes.

Os arrays possuem uma propriedade chamada `length`, que é comum ver em linguagens de programação, essa propriedade nos informa o comprimento do array, por exemplo:

```
var nomes = ["Lucas", "Otávio", "Gabriel", "Claudemir", "Marcela", "Lívia" ]

nomes.length
```

nomes.length → 5, pois nosso array possui 5 itens

Objetos

Os objetos são representações de objetos da vida real ou então algo que você queira representar em sua aplicação, podemos criar um objeto chamado "pessoa", ou então um objeto chamado "notebook", podemos criar um objeto como criamos uma variável, objetos possuem propriedades que podem ser de diversos tipos:

```
var pessoa = {
  nome: "Lucas",
  idade: 23,
  instrutor: true
}
```

Note que usamos **{}** para os objetos, e cada propriedade é separada uma da outra com o uso de vírgulas, e cada propriedade pode ser acessada através de um ponto, veja:

```
var pessoa = {
  nome: "Lucas",
  idade: 23,
  instrutor: true
}

pessoa.nome
pessoa.idade
pessoa.instrutor
```

pessoa.nome → "Lucas"

pessoa.idade → 23

pessoa.instrutor → true

Arrays de objetos

Voltando um pouco nos arrays, é muito comum vermos arrays que abrigam vários objetos, isso é muito útil em situações cotidianas e pode ter certeza que você irá se deparar muito com esse tipo de estrutura, veja um exemplo de um array de pessoas, onde cada pessoa é representada por um objeto:

```
var pessoas = [
    {
        nome: "Lucas",
        idade: 23,
        instrutor: true
    },
    {
        nome: "Otávio",
        idade: 23,
        instrutor: true
    },
    {
        nome: "Marcela",
        idade: 40,
        instrutor: false
    },
]
```

Observe que a ideia segue a mesma, cada objeto tem uma posição (indice) e podemos acessar determinado objeto com base nesse indice, por exemplo:

```
pessoas[1]
```

Nesse caso teremos como retorno o seguinte:

```
{
    nome: "Otávio",
    idade: 23,
    instrutor: true
}
```

Pois na posição 1 do nosso array de pessoas temos o objeto que contém o Otávio.

Além disso podemos unir as duas formas de acessar informações, por exemplo se quisermos acessar a idade do Otávio, como fazemos? Veja:

```
pessoas[1].idade
```

Nesse caso acima teremos **23** como retorno.

Quando entrarmos nos estudos de JavaScript vamos aprender a manipular arrays e objetos com métodos correspondentes a essas estruturas, por exemplo, como adiciono itens em um array? Como apago itens de um array? Isso e muito mais, mas por enquanto é importante entender o que é e para que serve um array.

Loops

Arrays e objetos se tornam ainda mais interessantes quando usamos loops, os loops são usados para percorrer estruturas de dados e fazer algo com essa ideia, para você visualizar melhor lembre do array usado nos exemplos anteriores:

```
var pessoas = [
    {
        nome: "Lucas",
        idade: 23,
        instrutor: true
    },
    {
        nome: "Otávio",
        idade: 23,
        instrutor: true
    },
    {
        nome: "Marcela",
        idade: 40,
        instrutor: false
    },
]
```

Como podemos visualizar o nome de cada pessoa sem fazer isso de uma forma muito manual? Usando um loop que percorra esse array, entenda que os loops são comuns em várias linguagens de programação, muda apenas a sintaxe (maneira de escrever), mas a ideia é a mesma, percorrer estruturas, entendido isso vamos ver um exemplo de loop sendo usado em nosso array de exemplo:

```
var pessoas = [
    {
        nome: "Lucas",
        idade: 23,
        instrutor: true
    },
    {
        nome: "Otávio",
        idade: 23,
        instrutor: true
    },
    {
        nome: "Marcela",
        idade: 40,
        instrutor: false
    },
]

for (var i = 0; i < pessoas.length; i++) {
    escreva(pessoas[i].nome);
}
```

Vejamos esse loop chamado **for**, esse loop é muito utilizado em linguagens como o JavaScript que você irá aprender, porém entenda que ele é um loop comum em várias linguagens de programação, então vamos entender a ideia dele.

Esse loop recebe entre parênteses alguns argumentos para funcionar corretamente, que no caso é a criação de uma variável **i**, que no caso poderia ter qualquer nome, mas o **i** vem de **iteração/ iterar**, depois dissemos que enquanto o **i** for menor que o comprimento do array pessoas ele continua, e a cada iteração ele aumenta em 1, no caso determinamos isso no **i++**, que quer dizer: **i = i + 1**, e a cada iteração nós queremos ver o nome de uma pessoa de dentro do nosso array.

Veja um outro exemplo de loop for:

```
var numeros = [1, 2, 3, 4, 5]

for (var i = 1; i <= numeros.length; i++) {
    escreva(i * 10);
}
```

Teremos agora um resultado assim:

```
10
20
30
40
50
```

Pois pegamos cada item do array e multiplicamos por 10.

Veja abaixo mais um exemplo de uma manipulação simples de estruturas usando arrays de objetos e loops:

```
var carros = [
    {
        marca: "Chevrolet",
        modelo: "Chevette"
    },
    {
        marca: "Honda",
        modelo: "Civic"
    },
    {
        marca: "Volkswagen",
        modelo: "Jetta"
    },
]

var qtdeCarros = carros.length;

escreva('Temos ' + qtdeCarros + ' carros em estoque: ');
for (var i = 0; i < carros.length; i++) {
  escreva(carros[i].marca + ' ' + carros[i].modelo)
}
```

Como resultado temos:

```
Temos 3 carros em estoque:
Chevrolet Chevette
Honda Civic
Volkswagen Jetta
```

O que vimos aqui é o básico dessas estruturas, veremos em breve isso sendo usado com JavaScript e fazendo coisas mais reais onde você terá uma compreensão melhor de tudo que estamos falando aqui, o importante nesse momento é começar a associar as ideias da base da programação, fique tranquilo se ainda não entendeu tudo 100%, aprender programar é um processo que exige paciência.