



# Trabalhando com objetos

Evelise Dib

# Objetos



Objetos em JavaScript, assim como em muitas outras linguagens de programação, podem ser comparados com objetos na vida real. O conceito de objetos em JavaScript pode ser entendido com objetos tangíveis da vida real.

A diferença é que usamos para gravar alguns dados com tipos e propriedades para que possamos usá-las depois.

# Objetos

Como exemplo um vaso.

Ele tem propriedades de cores, formato, peso, material. Se usarmos isso para o javascript, obtemos um conjunto de dados com suas respectivas características

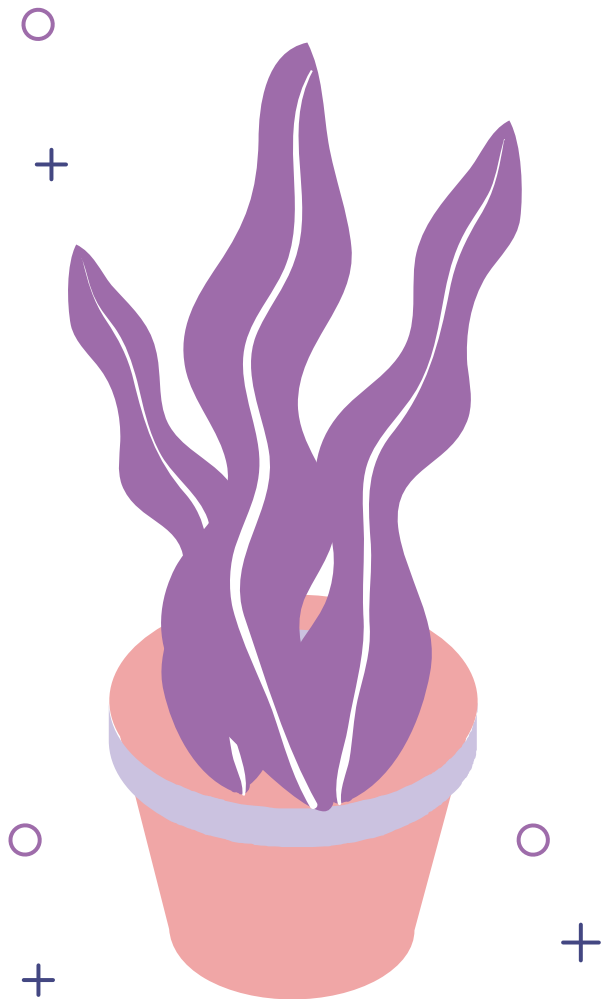
```
const vaso = {  
  "formato": "redondo",  
  "cor": "rosa",  
  "peso": "1,4kg",  
  "material": "cerâmica"  
}
```

# Objetos

Quando acessamos os valores de vaso, obtemos um conjunto de características. Para acessar esses valores, usamos o ponto e a propriedade escolhida.

```
console.log(vaso.peso);  
// 1,1kg  
console.log(vaso.material);  
// cerâmica
```

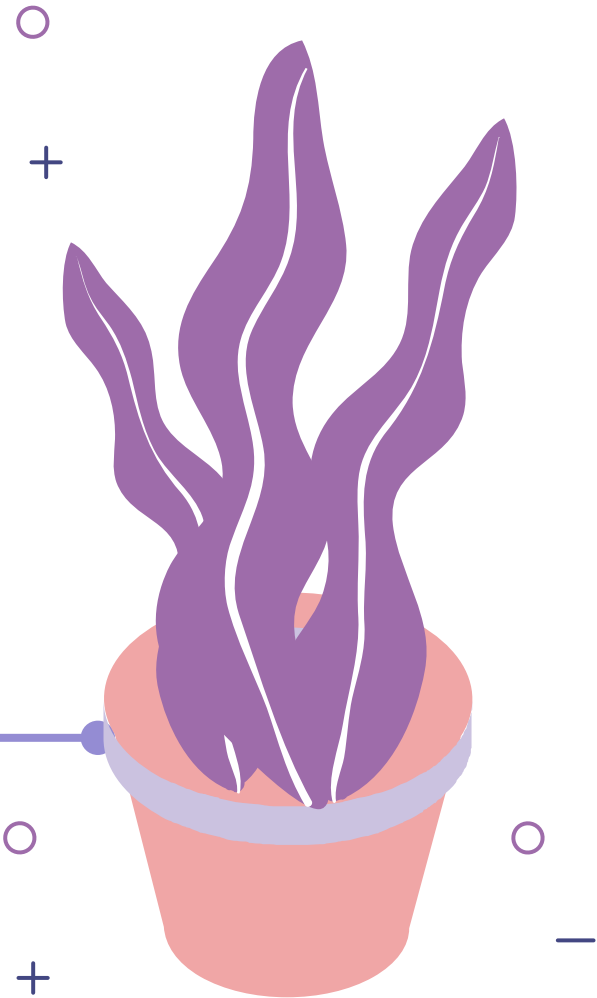
Quando acessamos uma propriedade do objeto que não existe o valor retornado será *undefined* pois não existe.



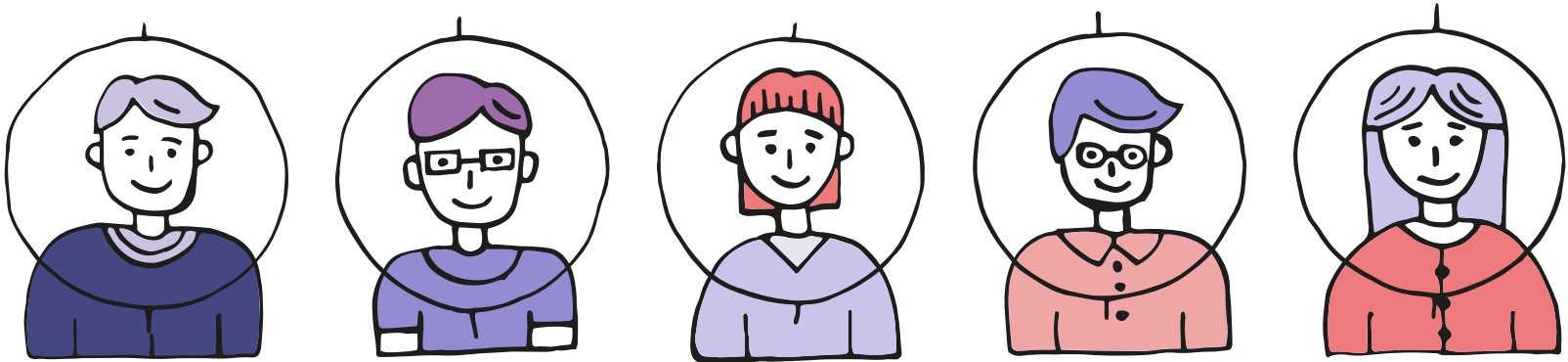
# Alterando valores

Também podemos alterar os valores que foram definidos nesse objeto, acessando as propriedades através do colchete.

```
vaso["peso"] = "2,2kg";  
console.log(vaso.peso);  
// "2,2kg"
```



# E quando temos uma lista?



# Chamamos de Array


Array é conjunto de dados. Usamos para armazenar mais de um valor em uma única variável.

Quando falamos em lista (array), cada item daquela lista tem um número ligado a ele (chamamos de índice) *index*.

Usamos os [ ] para começar a usar uma lista no javascript



```
const devas = ['Isa', 'Monica', 'Cintia', 'Eve'];
```



```
const Evelise = 'Eve';  
const Isabella = 'Isalla';  
const Monica = 'Monica';  
const Cintia = 'Cintia';
```

# Usando o índice do array

Diferente de um objeto, quando trabalhamos com a lista, precisamos informar qual item da lista desejamos visualizar. Para isso, utilizamos o nosso índice para informar a posição.

```
const devas = ['Isa', 'Monica', 'Cintia', 'Eve'];
```

```
console.log(devas[0]);  
//Isa  
console.log(devas[1]);  
//Monica  
console.log(devas[2]);  
//Cintia  
console.log(devas[3]);  
//Evelise
```





# Tamanho de uma lista

Quando trabalhamos com uma lista, podemos verificar quantos itens existem dentro de uma determinada lista.



```
console.log(devas.length);  
//4
```



# Acessando um array via loop

Quando trabalhamos com um array, podemos fazer o que chamamos de loop para percorrer todos os valores de uma lista e acessar suas propriedades

```
const devas = ['Evelise', 'Isa', 'Monica', 'Cintia', 'Ju'];  
  
for(let i = 0; i < devas.length; i++){  
  console.log(devas[i]);  
}
```



# Acessando um array via loop

Alem do for assim, temos duas outras formas de percorrer um array de maneira mais fácil.

For in (usado para pegar o índice de cada item)

```
const devas = ['Evelise', 'Isa', 'Monica', 'Cintia', 'Ju'];  
  
for(let index in devas){  
  console.log(index);  
}
```

For of (usado para pegar o valor de cada item)

```
const devas = ['Evelise', 'Isa', 'Monica', 'Cintia', 'Ju'];  
  
for(let deva of devas){  
  console.log(deva);  
}
```



# E se misturar objeto com array?




# Acessando um array via loop

Podemos colocar objetos para atribuir mais de uma característica aos valores de cada item em uma lista.

Para acessar as propriedades, sempre precisamos passar o índice primeiro, depois a propriedade. Caso estejamos usando um loop, não é necessário, pois o `for of` faz isso para nós “por trás dos panos”



```
console.log(devas[0].nome);  
//Cintia
```



```
const devas = [  
  {  
    nome: 'Cintia',  
    funcao: 'Deva and Mentora'  
  },  
  {  
    nome: 'Isa',  
    funcao: 'Deva and Mentora'  
  },  
  {  
    nome: 'Monica',  
    funcao: 'Deva and Mentora'  
  },  
  {  
    nome: 'Ju',  
    funcao: 'Deva and Mentora'  
  }  
]  
  
for(let deva of devas){  
  console.log(deva.nome);  
  console.log(deva.funcao);  
}
```

# Objetos com custom name

Ainda existe um caso que podemos usar um objeto com as chaves nomeadas para trabalhar no formato de lista

Nesse caso, usamos o for in para pegar o “nome” atribuído aquele objeto e acessar as propriedades.

```
const devas = {  
  "Cintia":{  
    funcao: 'Deva and Mentora',  
    idade: 'x',  
  },  
  "Isa":{  
    funcao: 'Deva and Mentora',  
    idade: 'x',  
  },  
  "Monica":{  
    funcao: 'Deva and Mentora',  
    idade: 'x',  
  },  
  "Ju":{  
    funcao: 'Deva and Mentora',  
    idade: 'x',  
  }  
}  
  
for(let nome in devas){  
  console.log(nome);  
  console.log(devas[nome].funcao);  
}
```

# Arrays e objetos dentro do outro

Você pode ter outros arrays dentro do objeto e dentro de outro array (OMG WHAT?)

Sim, e podemos fazer um loop dentro do loop para isso!

```
const devasInArray = [
  {
    "nome": "Isa",
    "funcao": "deva e mentora",
    "idade": "x",
    "animaisFavoritos": ['Gato', 'Cachorro', 'Peixe']
  },
  {
    "nome": "Cintia",
    "funcao": "deva e mentora",
    "idade": "y",
    "animaisFavoritos": ['Gato', 'Sapo']
  }
]

for(let deva of devasInArray){
  console.log(deva.nome);
  console.log(deva.funcao);
  console.log(deva.idade);
  for(let animal of deva.animaisFavoritos){
    console.log(animal);
  }
  console.log('-----');
}
```

# Para estudar

Caso queira se aprofundar mais, busque por essas funções para trabalhar de forma mais fácil com listas no javascript!

- Map
- Filter
- Reduce
- Sort

