
FIAP

Javascript – Objetos

Objeto

Em desenvolvimento de software, um objeto é uma coleção de dados (variáveis/propriedades) e funcionalidades (métodos/funções) relacionadas, encapsuladas em um único pacote.

Os objetos são usados para modelar coisas do mundo real, fornecendo uma maneira de descrever suas características e comportamentos.

Sintaxe

Como acontece com muitas coisas em JavaScript, a criação de um objeto geralmente começa com a definição e a inicialização de uma variável. Para inicializar um objeto vazio utilizamos os caracteres de chaves {}.

```
// Objeto vazio
let newObject = {}
```

```
// Objeto com propriedades
let person = {
  name: 'John',
  age: 30,
  city: 'New York',
  isStudent: false,
  hobbies: ['music', 'movies', 'sports'],
  greet: function () {
    console.log('Hello!')
  }
}
```

Acessando propriedades de um Objeto

Em JavaScript, existem duas formas principais de acessar as propriedades de um objeto: usando a notação de ponto (.) e usando colchetes ([]).

1. Notação de Ponto (.)

   Na notação de ponto, você acessa as propriedades de um objeto diretamente usando o nome da propriedade seguido por um ponto. Esta forma é a mais comum e geralmente é preferida quando o nome da propriedade é conhecido antecipadamente.

```
// Objeto com propriedades
let person = {
  name: 'John',
  age: 30,
  city: 'New York',
  isStudent: false,
  hobbies: ['music', 'movies', 'sports'],
  greet: function () {
    console.log('Hello!')
  }
}



console.log(person.name) // John
console.log(person.age) // 30
```

Acessando propriedades de um Objeto

Em JavaScript, existem duas formas principais de acessar as propriedades de um objeto: usando a notação de ponto (.) e usando colchetes ([]).

2. Notação de Colchetes ([])

   Na notação de colchetes, você acessa as propriedades de um objeto usando uma string dentro de colchetes. Esta forma é útil quando o nome da propriedade é dinâmico ou contém Caracteres especiais que não são válidos em identificadores JavaScript.

```
// Objeto com propriedades
let person = {
  name: 'John',
  age: 30,
  city: 'New York',
  isStudent: false,
  hobbies: ['music', 'movies', 'sports'],
  greet: function () {
    console.log('Hello!')
  },
  'day-of-birth': new Date('1990-01-01'),
}


console.log(person.'day-of-birth') // Error
console.log(person['day-of-birth']) // 1990-01-01
console.log(person['name']) // John
```

Acessando propriedades de um Objeto

Quando usar cada forma?

Notação de Ponto (.)
Use quando o nome da propriedade é conhecido antecipadamente e é um identificador válido em JavaScript.

Notação de Colchetes ([])
Use quando o nome da propriedade é dinâmico, definido em tempo de execução, ou contém caracteres especiais.

Modificando Propriedades

```
1   // Modificando um Objeto
2   const person = {
3     name: 'John',
4     age: 20
5   }
6
7
8   console.log(person.age) // 20
9
10  person.age = 25
11
12  console.log(person.age) // 25
13
```

```
16  // Adicionando uma nova propriedade
17  console.log(person.country) // undefined
18
19  person.country = 'Brazil'
20
21  console.log(person.country) // Brazil
```

```
28  // Deletando uma propriedade
29  delete person.country
30
31  console.log(person.country) // undefined
```

Modificando Propriedades

```
// Criando um objeto
const person = {
  name: 'John',
  age: 20
}

// Acessando o Objeto
console.log(person)

// Modificando uma propriedade
console.log(person.age) // 20

person.age = 25

console.log(person.age) // 25


// Adicionando uma nova propriedade
console.log(person.country) // undefined

person.country = 'Brazil'

console.log(person.country) // Brazil

// Acessando o objeto
console.log(person)

// Deletando uma propriedade
delete person.country

console.log(person.country) // undefined
```

Combinando objetos - Object.assign()

O método Object.assign() é usado para copiar os valores de todas as propriedades enumeráveis de um ou mais objetos de origem para um objeto de destino. Ele retorna o objeto de destino modificado.
O metodo possui no mínimo dois parâmetros (podendo ter infinitos), onde o primeiro é o objeto destino e do segundo em diante os objetos a serem combinados com o primeiro.

Combinando objetos - Object.assign()

FIAP

```
const person = {
  name: 'John Doe',
  age: 25
}

const address = {
  city: 'New York',
  country: 'USA'
}

const socialInfos = {
  facebook: 'https://facebook.com/johndoe',
  twitter: 'https://twitter.com/johndoe'
}

// Merge two objects
const result = Object.assign(person, address, socialInfos)
console.log(result)
```

Combinando objetos - Spread Operator (...)

FIAP

O operador de spread (...) permite que um objeto iterável, como um array ou objeto, seja expandido em locais onde zero ou mais argumentos (para funções) ou elementos (para arrays) são esperados.

- Facilita a cópia e a concatenação de arrays e objetos.
- Útil para passar argumentos para funções de forma mais concisa.
- Permite criar cópias modificadas de objetos sem alterar o original.

```
1   const array1 = [1, 2, 3]
2   const array2 = [...array1, 4, 5]
3
4   console.log(array2) // Output: [1, 2, 3, 4, 5]
```

```
1   const obj1 = { a: 1, b: 2 }
2   const obj2 = { ...obj1, c: 3 }
3
4   console.log(obj2) // Output: { a: 1, b: 2, c: 3 }
5
```

Combinando objetos

O Object.assign() é útil para copiar as propriedades de objetos existentes para um novo objeto, enquanto o spread operator é útil para expandir e copiar arrays e objetos de forma mais flexível em várias situações.

Métodos para manipulação Objetos

1. Object.keys()
   Retorna um array contendo os nomes das propriedades próprias e enumeráveis de um objeto.

2. Object.values()
   Retorna um array contendo os valores das propriedades próprias e enumeráveis de um objeto.

3. Object.entries()
   Retorna um array contendo pares [chave, valor] de todas as propriedades próprias e enumeráveis de um objeto.

Métodos para manipulação Objetos

```
1   const person = {
2     name: 'John Doe',
3     age: 25,
4     location: 'Lagos'
5   }
6
7   const keys = Object.keys(person) // ['name', 'age', 'location']
8
9   const values = Object.values(person) // ['John Doe', 25, 'Lagos']
10
11  const entries = Object.entries(person) // [['name', 'John Doe'], ['age', 25], ['location', 'Lagos']]
12
```

JSON (JavaScript Object Notation)

JSON é uma sintaxe para serialização de objetos, matrizes, números, strings, booleanos, e null. Baseia-se em sintaxe Javascript, mas é distinta desta: alguns Javascript não são JSON, e alguns JSON não são Javascript.

O JSON é amplamente utilizado para a troca de dados na internet devido à sua simplicidade, leveza e facilidade de compreensão tanto para humanos quanto para máquinas

Como um formato de dados independente de linguagem, o JSON tornou-se uma escolha popular para a comunicação entre sistemas distribuídos, sendo utilizado em diversas aplicações, como APIs RESTful, sistemas de armazenamento de dados em nuvem e comunicação entre cliente e servidor em aplicativos web e móveis

JSON (JavaScript Object Notation)

Além disso, o JSON é suportado por uma ampla gama de linguagens de programação além de JavaScript, incluindo Python, Java, C#, Ruby, entre outras.

Isso significa que os dados formatados em JSON podem ser facilmente manipulados e interpretados em diferentes ambientes e tecnologias, promovendo a interoperabilidade e a integração entre sistemas heterogêneos.

Essa capacidade de interoperabilidade torna o JSON uma escolha versátil e poderosa para desenvolvedores que precisam trocar dados entre diferentes sistemas e plataformas de maneira eficiente e confiável.

JSON (JavaScript Object Notation)

Sintaxe

- Chave-valor separados por dois pontos (:).
- Pares chave-valor separados por vírgula (,).
- Os valores podem ser strings, números, booleanos, arrays, outros objetos ou null.

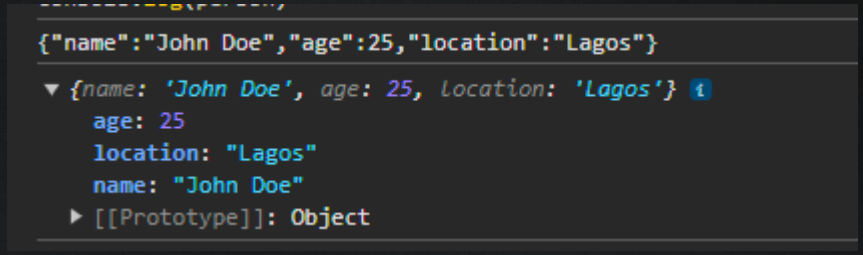
```json
{
  "name": "Jonh Doe",
  "age": 25,
  "location": "Lagos"
}
```

```javascript
const person = {
  name: 'John Doe',
  age: 25,
  location: 'Lagos'
}
```

JSON.stringify()

O método JSON.stringify() é usado para converter um objeto JavaScript em uma string JSON. Ele recebe um objeto como argumento e retorna uma representação JSON desse objeto. Geralmente utilizamos esse metodo para trafegar o JSON para o servidor ou salvar como um arquivo de texto.

```
1   const person = {
2     name: 'John Doe',
3     age: 25,
4     location: 'Lagos'
5   }
6
7   // Stringify
8   console.log(JSON.stringify(person))
9
10  // Objeto
11  console.log(person)
12
```



JSON.parse()

O método JSON.parse() é usado para analisar uma string JSON e converter seu conteúdo em um objeto JavaScript correspondente.

```
1   const json = '{"name":"John Doe","age":25,"location":"Lagos"}'
2
3   const obj = JSON.parse(json)
4
5   // Objeto Javascript
6   console.log(obj)
7
8   // JSON string
9   console.log(json)
10
```

```
> json.name
< undefined
> obj.name
< 'John Doe'
>
```

```
▼ {name: 'John Doe', age: 25, location: 'Lagos'} i                    VM95:6
    age: 25
    location: "Lagos"
    name: "John Doe"
  ▶ [[Prototype]]: Object
{"name":"John Doe","age":25,"location":"Lagos"}                       VM95:9
```

Uso e Aplicações

- Comunicação Cliente-Servidor: Usado para enviar e receber dados entre um cliente (geralmente um navegador da web) e um servidor, especialmente em APIs RESTful.
- Armazenamento de Dados: Útil para armazenar e recuperar dados em sistemas de armazenamento de dados em nuvem, como bancos de dados NoSQL.
- Serialização e Deserialização: Facilita a conversão de objetos JavaScript em strings JSON para armazenamento ou transmissão e vice-versa para recuperação e manipulação de dados.
- Configuração e Persistência de Estado: Pode ser usado para salvar e carregar o estado de uma aplicação, por exemplo, para salvar as preferências do usuário ou o estado de uma sessão.

Esses métodos são fundamentais para a troca de dados estruturados entre sistemas distribuídos, garantindo a interoperabilidade e a compatibilidade entre diferentes tecnologias e plataformas.

Trabalhando com JSON em Python

Exemplo de uso de JSON em Python

```python
import json

json_string = '{"name": "Alice", "age": 30, "city": "New York"}'

# convert json string to python dictionary, equivalent to JSON.parse() in JavaScript
data = json.loads(json_string)
print(data)

# convert python dictionary to json string, equivalent to JSON.stringify() in JavaScript
json_string = json.dumps(data, indent=4)
print(json_string)
```

Trabalhando com JSON em Java

Exemplo de uso de JSON em Java usando e biblioteca GSON
https://central.sonatype.com/artifact/com.google.code.gson/gson/2.8.6?smo=true

```java
String jsonString = "{\"name\":\"Alice\",\"age\":30,\"city\":\"New York\"}";

// create a new Gson object
Gson gson = new Gson();

// define the type of the data
Type type = new TypeToken<Map<String, Object>>() {}.getType();

// transform json string into data
Map<String, Object> data = gson.fromJson(jsonString, type);

// print data
System.out.println(data);

// transform data indo json string
String json = gson.toJson(data);

// print json string
System.out.println(json);
```

Exercícios

1. Crie um formulário no HTML com Altura e Peso e um botão de calcular. Ao clicar em calcular, usando Javascript, calcule o IMC (peso/altura²). No script utilize um objeto para salvar as informações de entrada e de saída. Exiba o resultado em kg/m

2. Continuando o exercício anterior, crie uma lista de histórico dos cálculos realizados anteriormente, e a cada nova entrada preencha a lista com um valor novo. Exiba o histórico no HTML com todos os dados de entrada e de saída

Dúvidas, críticas ou sugestões?


FIAP