

```
length - 1;      return
b.push(a[c]);    }
((\r|\n|\r)/gm, " ")
inp_array.length;
c.push(inp_array)
].sort, inp_array);
keyWord(a, ""); -1 <
keyWord(a, ""); -1 < b && a
("); use_array(a,
function use_array(a,
array(a, b) { for (
var c = -1, d
Sort(a) {

```

Funções em JavaScript

Bem-vindo à apresentação sobre funções em JavaScript! Descubra o poder e utilidade das funções e aprenda como usá-las para criar código eficiente e modular.



by Iuri Santos

O que são funções em JavaScript

As funções em JavaScript são blocos de código reutilizáveis que podem ser chamados para executar uma tarefa específica. Elas permitem agrupar instruções relacionadas e fornecer uma estrutura organizada ao seu código.

JS user.js src/models

```
import { Schema, model } from 'mongoose'  
import { isEmail } from 'validator'  
import { hash } from 'bcrypt'|
```

```
> const UserSchema = Schema({ ...  
})
```

```
UserSchema.pre('save', function async() {  
  if (user.isModified('password')) {  
    this.password = await hash(this.password)  
  }  
})
```

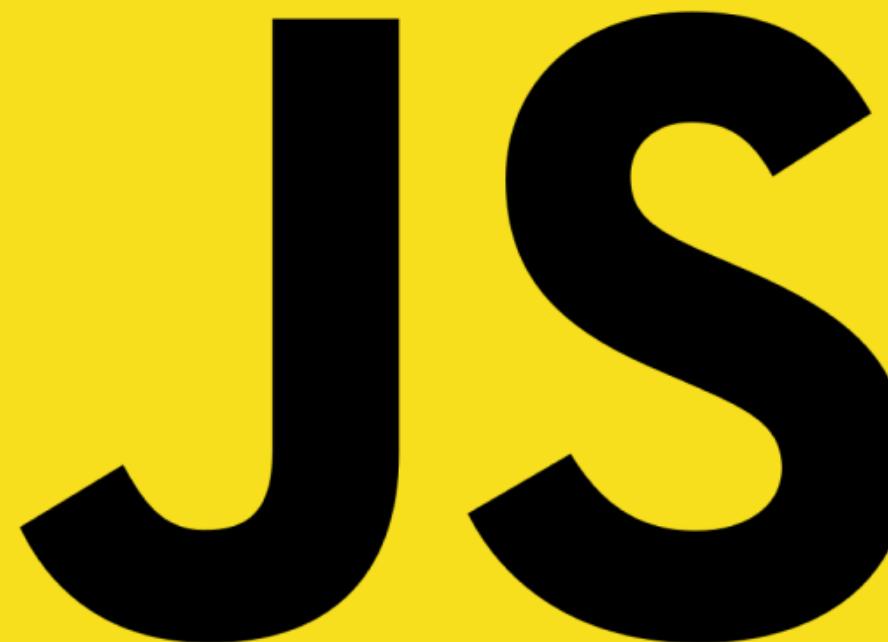
```
User = model('User', UserSchema)
```

```
export default User;
```

```
// Sample Codes  
// email : {  
//   type : String,  
//   required : true,  
//   trim : true,  
//   validate(value) {  
//     if(!validator.isEmail(value)) {  
//       throw new Error('Email is not valid')  
//     }  
//   }  
// }|
```

Importância e utilidade das funções

As funções desempenham um papel fundamental na programação em JavaScript, tornando o código mais legível, reutilizável e fácil de depurar. Elas ajudam a quebrar o código em partes menores, o que facilita a manutenção e a colaboração em projetos complexos.



Sintaxe básica das funções

Parâmetros e argumentos

As funções podem receber parâmetros, que são valores fornecidos quando a função é chamada. Dentro da função, podemos usar esses parâmetros para realizar operações específicas.

1

2

3

Declaração de função

Para criar uma função, usamos a palavra-chave `function` seguida pelo nome da função e um par de parênteses. Por exemplo: `function minhaFuncao() { }`

Retorno de valores

Uma função pode retornar um valor usando a palavra-chave `return`. Isso permite que o resultado seja usado em outras partes do código.



Sintaxe básica:

```
function nomeDaFuncao(parametro1, parametro2, ...) {  
    // Bloco de código da função  
}
```

Exemplos de aplicação

Calcule a média

Crie uma função que recebe um array de números e retorna a média deles.

Verifique se é par

Implemente uma função que verifica se um número é par e retorna true ou false.



Função: Calcule a média

```
function calcularMedia(numero1, numero2) {  
    const soma = numero1 + numero2;  
    const media = soma / 2;  
    return media;  
}  
  
const resultado = calcularMedia(5, 7); // resultado conterá 6
```

Função: Verifique se é Par

```
function isPar(numero) {  
    return numero % 2 === 0;  
}  
  
// Exemplos de uso:  
console.log(isPar(4)); // true  
console.log(isPar(7)); // false  
console.log(isPar(0)); // true
```

Funções sem parâmetros

As funções também podem ser usadas sem parâmetros. Elas são úteis quando queremos que o código execute uma ação específica sem a necessidade de receber informações externas.

```
function saudacao() {  
    console.log('Olá, mundo!');  
}  
  
saudacao();
```

Exercício:

Crie a mesma função de média só que com os parâmetros inseridos pelo usuário