**Objetos**

O objeto é uma entidade independente com algumas propriedades e tipos. As propriedades normalmente têm chave e valor, a chave sendo o nome e o valor o valor de fato.

As propriedades podem ser acessadas com ‘.’ ou com ‘[“*nome da chave*”]’. O uso dos colchetes pode ser útil quando não sabemos o nome das chaves e também quando temos que percorrer uma lista de chaves.

Apesar de um objeto poder ser declarado com const ainda é possível fazer alterações nele, porque a palavra reservada const não permite uma atribuição nova completa, não é possível atribuir um novo objeto, mas as propriedades podem ser **alteradas.**

É possível deletar um conjunto chave/valor utilizando o operador **delete**

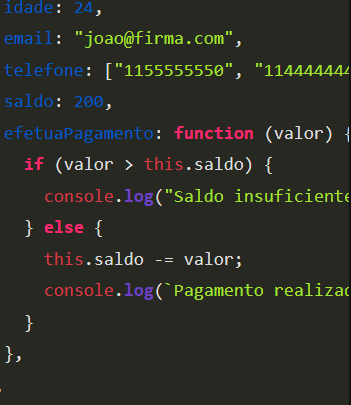
**delete** objPersonagem.aliado

**delete** objPersonagem["status"]

console.**log**(objPersonagem.aliado) //undefined

console.**log**(objPersonagem.status) //undefined

Também é possível adicionar funções como propriedades de objetos, nesse caso é chamado de método da função. Nela podemos descrever todo o comportamento e ela não precisa de um nome, já que é a propriedade já é nomeada. Existe a palavra reservada chamada ***this***, essa palavra se refere literalmente ao que estamos manipulando no momento.



O delete apaga tanto a chave quanto o valor. O valor de retorno do operador **delete** é um booleano (true ou false) porém, mesmo que tentemos apagar uma propriedade que não existe ele vai retornar *true,* como se realmente tivesse apagado.

***Objeto literal ->*** é criado com uma notação literal, ou seja, criado da forma mais usual colocando as chaves atribuídas à uma variável. Um objeto literal sempre aponta para um mesmo local na memória, mesmo criando cópias dele.

**Object.create() ->** o objeto também pode ser criado com através desse método. É um bom método para fazer uma cópia sem usar o mesmo espaço de memória, o parâmetro recebido por esse método é um objeto já criado.

Método **for...in ->** permite a iteração sobre propriedades de um objeto como em um array.

**\*\***

*Caso a sintaxe de espalhamento (spread operator) seja usada em objetos que tenham chaves/propriedades com o mesmo nome, o JS vai sobrescrever o valor destas propriedades à medida que encontra novas valores com o mesmo nome de chave.*

Apesar de prático, o uso de sintaxe de espalhamento pode gerar bastante processamento, então deve ser usado com cuidado em caso de loops ou funções recursivas

**JSON ->** *java-script object notation (notação de objeto java-script)*

Na notação JSON, para escrever um objeto, não há variáveis e as chaves das propriedades têm “” (aspas (duplas)). Além disso arrays e objetos aninhados podem existir e é **proibido** ter *trailing comma* (vírgula no último item de um bloco).

Outros detalhes são que:

- Funções não são permitidas;

- Comentários não são permitidos;

- Suporta apenas tipos primitivos (str, number, boolean, null)

É um formato muito utilizado quanto temos um conjunto de configurações ou padrão que queremos colocar no código e que iremos querer acessar em outros lugares. Essa notação ainda permite transformar um objeto em string e transmitir eles para um site ou algo do tipo

**require() ->** é uma função do node para puxar código de algum outro lugar

No node podemos usar um objeto global JSON e com ele podemos usar um método para transformar o objeto em string que é o *.****stringify***

Uma das razões de usar *json* é que é muito mais simples transmitir dados em texto

O caminho inverso, transformar string em objeto, pode ser feito com o método .**parse()** (usado no JSON)

Para objetos é possível usar um método para saber se esse objeto tem uma certa propriedade. O método é ***hasOwnProperty(****“nome da propriedade”****).***