Introdução a objetos em JavaScript

Samuel Amaro

1 de maio de 2021

Trabalhando com JSON

JavaScript Object Notation (JSON) - JavaScript Objeto Notação é um formato baseado em texto padrão para representar dados estruturados com base na sintaxe do objeto JavaScript. É comumente usado para transmitir dados em aplicativos da Web (por exemplo, enviar alguns dados do servidor para o cliente, para que possam ser exibidos em uma página da Web ou vice-versa). Você se deparará com isso com bastante frequência, portanto, neste artigo, oferecemos tudo o que você precisa para trabalhar com o JSON usando JavaScript, incluindo a análise do JSON para que você possa acessar os dados dentro dele e criar o JSON.

Objetivo: Para entender como trabalhar com dados armazenados em JSON e criar seus próprios objetos JSON.

Não, sério, o que é o JSON?

JSON é um formato de dados baseado em texto seguindo a sintaxe do objeto JavaScript, que foi popularizada por Douglas Crockford. Mesmo que se assemelhe à sintaxe literal do objeto JavaScript, ele pode ser usado independentemente do JavaScript, e muitos ambientes de programação possuem a capacidade de ler (analisar) e gerar JSON.

O JSON existe como uma string — útil quando você deseja transmitir dados por uma rede. Ele precisa ser convertido em um objeto JavaScript nativo quando você quiser acessar os dados. Isso não é um grande problema — o JavaScript fornece um objeto JSON global que possui métodos disponíveis para conversão entre os dois.

Nota: Converter uma string em um objeto nativo é chamado de

análise, enquanto a conversão de um objeto nativo em uma string para que possa ser transmitida pela rede é chamada de stringification.

Um objeto JSON pode ser armazenado em seu próprio arquivo, que é basicamente apenas um arquivo de texto com uma extensão de .json, e um MIME type (en-US) de application/json.

Estrutura JSON

Conforme descrito acima, um JSON é uma string cujo formato se parece muito com o formato literal do objeto JavaScript. Você pode incluir os mesmos tipos de dados básicos dentro do JSON, como em um objeto JavaScript padrão — strings, números, matrizes, booleanos e outros literais de objeto. Isso permite que você construa uma hierarquia de dados, assim:

```
1
2  {
3    "squadName": "Super hero squad",
4    "homeTown": "Metro City",
5    "formed": 2016,
6    "secretBase": "Super tower",
7    "active": true,
8    "members": [
9      {
10       "name": "Molecule Man",
11       "age": 29,
12       "secretIdentity": "Dan Jukes",
13       "powers": [
14         "Radiation resistance",
15         "Turning tiny",
16         "Radiation blast"
17       ]
18     },
19     {
20       "name": "Madame Uppercut",
21       "age": 39,
22       "secretIdentity": "Jane Wilson",
23       "powers": [
24         "Million tonne punch",
25         "Damage resistance",
26         "Superhuman reflexes"
27       ]
28     },
29     {
30       "name": "Eternal Flame",
31       "age": 1000000,
32       "secretIdentity": "Unknown",
33       "powers": [
34         "Immortality",
35         "Heat Immunity",
36         "Inferno",
37         "Teleportation",
38         "Interdimensional travel"
```

```
      ]
    }
  ]
}
```

Listing 1: Estrutura JSON

Se nós carregássemos esse objeto em um programa JavaScript, analisado em uma variável chamada **superHeroes** por exemplo, poderíamos então acessar os dados dentro dele usando a mesma notação dot/bracket que observamos no artigo básico do objeto JavaScript. Por exemplo:

```

  //acessando uma propriedade de menbro de uma classe ou instancia
    de objeto
  superHeroes.homeTown //return "Metro City"
  //mesma coisa mas fazendo de outra maneira
  superHeroes['active'] //return true
```

Para acessar os dados mais abaixo na hierarquia, basta encadear os nomes de propriedades e os índices de array necessários juntos. Por exemplo, para acessar o terceiro superpoder do segundo herói listado na lista de membros, você faria isso:

```

  //acessando os dados mais abaixo na hierarquia, usando a mesma
   sintaxe
  //de como acessar menbros de classe, ou de intancia de objetos

  superHeroes['members'][1]['powers'][2] //return "Superhuman
   reflexes"
```

1. Primeiro temos o nome da variável — `superHeroes`.
2. Por dentro, queremos acessar a propriedade dos `members`, então usamos `["members"]`.
3. `members` contém uma matriz preenchida por objetos. Queremos acessar o segundo objeto dentro da matriz, então usamos [1].
4. Dentro deste objeto, queremos acessar a propriedade `powers`, então usamos `["powers"]`.
5. Dentro da propriedade `powers` está um array contendo os superpoderes do herói selecionado. Nós queremos o terceiro, então nós usamos `[2]`

Matrizes como JSON

Acima, mencionamos que o texto JSON basicamente parece com um objeto JavaScript, e isso é correto correto. A razão pela qual dissemos "principalmente

certo"é que uma matriz também é válida como JSON, por exemplo:

```
[
  {
    "name": "Molecule Man",
    "age": 29,
    "secretIdentity": "Dan Jukes",
    "powers": [
      "Radiation resistance",
      "Turning tiny",
      "Radiation blast"
    ]
  },
  {
    "name": "Madame Uppercut",
    "age": 39,
    "secretIdentity": "Jane Wilson",
    "powers": [
      "Million tonne punch",
      "Damage resistance",
      "Superhuman reflexes"
    ]
  }
]
```

O código acima é um JSON perfeitamente válido. Você teria que acessar itens de matriz (em sua versão analisada) iniciando com um índice de matriz, por exemplo `[0]["powers"][0]`.

Outras Notas

- O JSON é puramente um formato de dados - contém apenas propriedades, sem métodos.
- JSON requer aspas duplas para serem usadas em torno de strings e nomes de propriedades. Aspas simples não são válidas.
- Mesmo uma única vírgula ou cólon perdidos podem fazer com que um arquivo JSON dê errado e não funcione. Você deve ter o cuidado de validar todos os dados que você está tentando usar (embora o JSON gerado por computador seja menos provável de incluir erros, desde que o programa gerador esteja funcionando corretamente).
- O JSON pode realmente assumir uma forma de qualquer tipo de dados que seja válido para inclusão dentro do JSON, não apenas matrizes ou. Por exemplo, uma string única ou número seria um objeto JSON válido.
- Ao contrário do código JavaScript no qual as propriedades do objeto podem estar sem aspas, em JSON, somente as strings entre aspas podem ser usadas como propriedades.