使用 JSON

JavaScript 对象表示法 (JSON) 是一种基于文本的标准格式，用于表示基于 JavaScript 对象语法的结构化数据。它通常用于在 Web 应用程序中传输数据（例如，将一些数据从服务器发送到客户端，以便在网页上显示，反之亦然）。您会经常遇到它，因此在本文中，我们为您提供了使用 JavaScript 处理 JSON 所需的一切，包括解析 JSON 以便您可以访问其中的数据，以及创建 JSON。

先决条件：	基本的计算机知识，对 HTML 和 CSS 有基本的了解，熟悉 JavaScript 基础知识（请参阅第一步和构建块）和 OOJS 基础知识（请参阅对象简介）。
客观的：	了解如何使用存储在 JSON 中的数据，并创建您自己的 JSON 字符串。

不，真的，什么是 JSON?

JSON是一种基于文本的数据格式，遵循 JavaScript 对象语法，由 Douglas Crockford 推广。尽管它与 JavaScript 对象字面量语法非常相似，但它可以独立于 JavaScript 使用，并且许多编程环境都具有读取（解析）和生成 JSON 的能力。

JSON 以字符串的形式存在——当你想通过网络传输数据时很有用。当您要访问数据时，需要将其转换为本机 JavaScript 对象。这不是什么大问题——JavaScript 提供了一个全局JSON对象，它具有可用于在两者之间进行转换的方法。

> **注意：**将字符串转换为原生对象称为*反序列化*，而将原生对象转换为字符串以便通过网络传输称为*序列化*。

一个 JSON 字符串可以存储在它自己的文件中，它基本上只是一个扩展名为 `.json` 、MIME 类型为 `application/json` .

JSON结构

如上所述，JSON 是一个字符串，其格式非常类似于 JavaScript 对象字面量格式。您可以在 JSON 中包含与标准 JavaScript 对象中相同的基本数据类型——字符串、数字、数组、布尔值和其他对象文字。这允许您构建数据层次结构，如下所示：

```
{
  "squadName": "Super hero squad",
  "homeTown": "Metro City",
  "formed": 2016,
  "secretBase": "Super tower",
  "active": true,
  "members": [
    {
      "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"
      ]
    },
    {
      "name": "Eternal Flame",
      "age": 1000000,
      "secretIdentity": "Unknown",
      "powers": [
        "Immortality",
        "Heat Immunity",
        "Inferno",
        "Teleportation",
```

```
      "Interdimensional travel"
    ]
  }
 ]
}
```

如果我们将这个字符串加载到一个 JavaScript 程序中并将其解析为一个名为 `superHeroes` example 的变量，那么我们就可以使用我们在JavaScript对象基础知识文章中看到的相同的点/括号表示法来访问其中的数据。例如：

```
superHeroes.homeTown
superHeroes['active']
```

要访问层次结构下的数据，您必须将所需的属性名称和数组索引链接在一起。例如，要访问成员列表中列出的第二个英雄的第三个超级大国，您可以这样做：

```
superHeroes['members'][1]['powers'][2]
```

1. 首先，我们有变量名 — `superHeroes` 。
2. 在里面，我们想要访问 `members` 属性，所以我们使用 `["members"]` .
3. `members` 包含一个由对象填充的数组。我们想访问数组中的第二个对象，所以我们使用 `[1]` .
4. 在这个对象内部，我们想要访问 `powers` 属性，所以我们使用 `["powers"]` .
5. 属性内部 `powers` 是一个数组，其中包含所选英雄的超能力。我们想要第三个，所以我们使用 `[2]` .

> **注意：在我们的**JSONTest.html 示例中，我们已经使上面看到的 JSON 在一个变量中可用（参见源代码 ）。尝试加载它，然后通过浏览器的 JavaScript 控制台访问变量中的数据。

数组作为 JSON

上面我们提到 JSON 文本基本上看起来像字符串中的 JavaScript 对象。我们还可以将数组与 JSON 相互转换。下面也是有效的 JSON，例如：

```
[
  {
    "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"
    ]
  }
]
```

以上是完全有效的 JSON。例如，您只需要从数组索引开始访问数组项（在其解析版本中） `[0]["powers"][0]` 。

其他注意事项

- JSON 纯粹是一个具有指定数据格式的字符串——它只包含属性，没有方法。
- JSON 要求在字符串和属性名称周围使用双引号。除了围绕整个 JSON 字符串外，单引号无效。
- 即使是单个错位的逗号或冒号也会导致 JSON 文件出错，并且无法正常工作。您应该小心验证您尝试使用的任何数据（尽管计算机生成的 JSON 不太可能包含错误，只要生成器程序正常工作）。您可以使用 JSONLint 等应用程序验证 JSON 。
- JSON 实际上可以采用任何有效包含在 JSON 中的数据类型的形式，而不仅仅是数组或对象。因此，例如，单个字符串或数字将是有效的 JSON。

- 与 JavaScript 代码中对象属性可能不带引号不同，在 JSON 中只能将带引号的字符串用作属性。

主动学习：处理 JSON 示例

因此，让我们通过一个示例来展示我们如何在网站上使用一些 JSON 格式的数据。

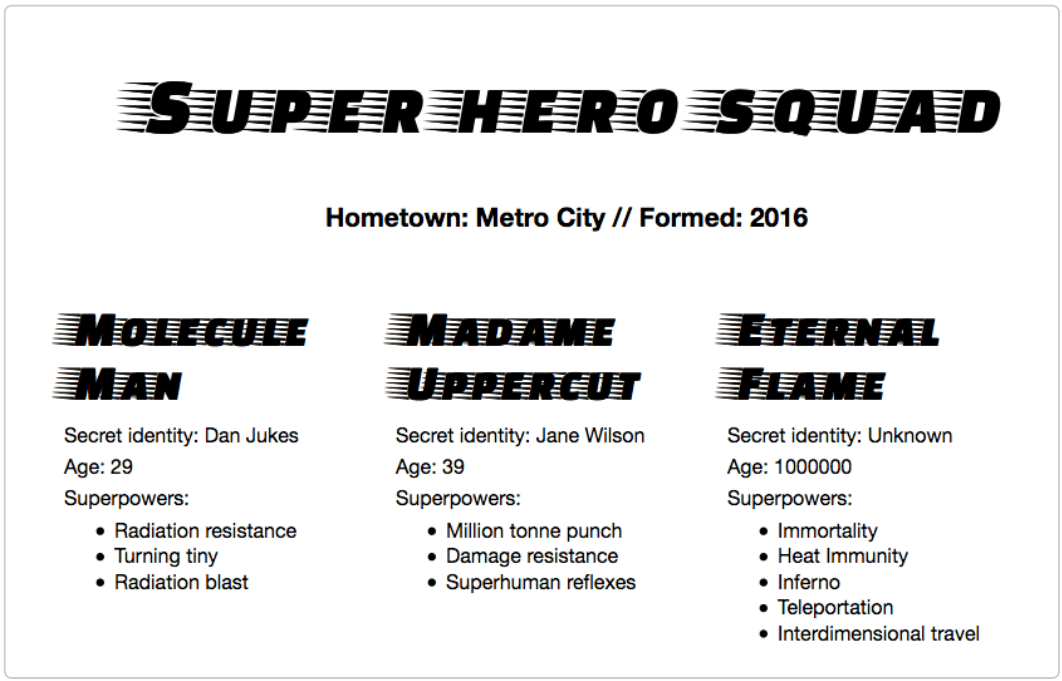
入门

首先，制作heroes.html 和style.css 文件的本地副本。后者包含一些简单的 CSS 来设置页面样式，而前者包含一些非常简单的主体 HTML，以及一个包含 `<script>` 我们将在本练习中编写的 JavaScript 代码的元素：

```
<header>

</header>

<section>

</section>

<script>

</script>
```

我们已经在我们的 GitHub 上提供了我们的 JSON 数据，网址为 https://mdn.github.io/learning-area/javascript/oojs/json/superheroes.json 。

我们将把 JSON 加载到我们的脚本中，并使用一些巧妙的 DOM 操作来显示它，如下所示：



顶层函数

顶级函数如下所示：

```
async function populate() {

  const requestURL = 'https://mdn.github.io/learning-
area/javascript/oojs/json/superheroes.json';
  const request = new Request(requestURL);

  const response = await fetch(request);
  const superHeroes = await response.json();

  populateHeader(superHeroes);
  populateHeroes(superHeroes);

}
```

要获取 JSON，我们使用一个名为Fetch的 API 。该 API 允许我们发出网络请求以通过 JavaScript 从服务器检索资源（例如图像、文本、JSON，甚至 HTML 片段），这意味着我们可以更新一小部分内容而无需重新加载整个页面。

在我们的函数中，前四行使用 Fetch API 从服务器获取 JSON：

- 我们声明 `requestURL` 变量来存储 GitHub URL
- 我们使用 URL 来初始化一个新 `Request` 对象。
- 我们使用该 `fetch()` 函数发出网络请求，并返回一个 `Response` 对象
- `json()` 我们使用对象的函数将响应检索为 JSON `Response` 。

> **注意：** API `fetch()` 是**异步的**。我们将在下一个模块中学习很多关于异步函数的知识，但现在，我们只说我们需要在使用 fetch API 的函数名称之前添加关键字，并在调用之前 `async` 添加关键字 `await` 任何异步函数。

毕竟， `superHeroes` 变量将包含基于 JSON 的 JavaScript 对象。然后我们将该对象传递给两个函数调用——第一个 `<header>` 用正确的数据填充，而第二个为团队中的每个英雄创建一个信息卡，并将其插入到 `<section>` .

填充标题

现在我们已经检索了 JSON 数据并将其转换为 JavaScript 对象，让我们通过编写上面引用的两个函数来使用它。首先，在之前的代码下面添加如下函数定义：

```
function populateHeader(obj) {
  const header = document.querySelector('header');
  const myH1 = document.createElement('h1');
  myH1.textContent = obj.squadName;
  header.appendChild(myH1);

  const myPara = document.createElement('p');
  myPara.textContent = `Hometown: ${obj.homeTown} // Formed:
${obj.formed}`;
  header.appendChild(myPara);
}
```

在这里，我们首先使用 创建一个h1元素 `createElement()` ， 将其设置 `textContent` 为等于 `squadName` 对象的属性，然后使用将其附加到标题 `appendChild()` 。然后我们对段落执行非常相似的操作：创建它，设置它

的文本内容并将它附加到标题。唯一的区别是它的文本被设置为包含对象的和属性的模板文字。 homeTown formed

创建英雄信息卡

接下来，在代码底部添加以下函数，创建并显示超级英雄卡片：

```
function populateHeroes(obj) {
  const section = document.querySelector('section');
  const heroes = obj.members;

  for (const hero of heroes) {
    const myArticle = document.createElement('article');
    const myH2 = document.createElement('h2');
    const myPara1 = document.createElement('p');
    const myPara2 = document.createElement('p');
    const myPara3 = document.createElement('p');
    const myList = document.createElement('ul');

    myH2.textContent = hero.name;
    myPara1.textContent = `Secret identity:
${hero.secretIdentity}`;
    myPara2.textContent = `Age: ${hero.age}`;
    myPara3.textContent = 'Superpowers:';

    const superPowers = hero.powers;
    for (const power of superPowers) {
      const listItem = document.createElement('li');
      listItem.textContent = power;
      myList.appendChild(listItem);
    }

    myArticle.appendChild(myH2);
    myArticle.appendChild(myPara1);
    myArticle.appendChild(myPara2);
    myArticle.appendChild(myPara3);
    myArticle.appendChild(myList);

    section.appendChild(myArticle);
  }
}
```

首先，我们将 `members` JavaScript 对象的属性存储在一个新变量中。该数组包含多个对象，其中包含每个英雄的信息。

接下来，我们使用for...of 循环遍历数组中的每个对象。对于每一个，我们：

1. 创建几个新元素： an `<article>` 、 an `<h2>` 、 three `<p>` s 和 a `<ul>` 。
2. 将 设置 `<h2>` 为包含当前英雄的 `name` 。
3. `secretIdentity` 用他们的、和一行写着“Superpowers：”的一行填充三个段落 `age` ，以介绍列表中的信息。
4. 将该属性存储 `powers` 在另一个名为的新常量中 `superPowers` ——它包含一个列出当前英雄超能力的数组。
5. 使用另一个 `for...of` 循环遍历当前英雄的超能力——我们为每个创建一个 `<li>` 元素，将超能力放入其中， `listItem` 然后使用. `<ul> myList appendChild()`
6. 我们做的最后一件事是在( )内附加 `<h2>` 、 `<p>` s 和，然后在. 附加内容的顺序很重要，因为这是它们在 HTML 中的显示顺序。 `<ul> <article> myArticle <article> <section>`

> **注意：**如果您在运行该示例时遇到问题，请尝试参考我们的 heroes-finished.html 源代码（也可以查看实时运行 。）

> **注意：**如果您无法理解我们用来访问 JavaScript 对象的点/方括号表示法，可以在另一个选项卡或文本编辑器中打开 superheroes.json 文件， 并在查看我们的内容时参考它 JavaScript。您还应该参考我们的JavaScript 对象基础知识文章，了解有关点和括号表示法的更多信息。

调用顶层函数

最后，我们需要调用我们的顶层 `populate()` 函数：

```
populate();
```

在对象和文本之间转换

上面的示例在访问 JavaScript 对象方面很简单，因为我们使用 `response.json()`.

但有时我们并没有那么幸运——有时我们收到一个原始的 JSON 字符串，我们需要自己将其转换为一个对象。而当我们想要通过网络发送一个 JavaScript 对象时，我们需要在发送之前将其转换为 JSON（字符串）。幸运的是，这两个问题在 web 开发中非常普遍，以至于浏览器中提供了一个内置的JSON对象，它包含以下两个方法：

- `stringify()`：接受一个对象作为参数，并返回等效的 JSON 字符串。

您可以在我们的heroes-finished-json-parse.html 示例（查看源代码）中看到第一个在运行——这与我们之前构建的示例完全相同，除了：

- 我们通过调用响应 `text()` 的方法以文本而不是 JSON 的形式检索响应
- 然后我们使用 `parse()` 将文本转换为 JavaScript 对象。

代码的关键片段在这里：

```
async function populate() {

  const requestURL = 'https://mdn.github.io/learning-
area/javascript/oojs/json/superheroes.json';
  const request = new Request(requestURL);

  const response = await fetch(request);
  const superHeroesText = await response.text();

  const superHeroes = JSON.parse(superHeroesText);
  populateHeader(superHeroes);
```

```
  populateHeroes(superHeroes);

}
```

正如您可能猜到的那样， `stringify()` 以相反的方式工作。尝试在浏览器的 JavaScript 控制台中逐行输入以下行以查看其运行情况：

```
let myObj = { name: "Chris", age: 38 };
myObj
let myString = JSON.stringify(myObj);
myString
```

在这里，我们创建了一个 JavaScript 对象，然后检查它包含的内容，然后使用将其转换为 JSON 字符串 `stringify()` ——将返回值保存在一个新变量中——然后再次检查它。

测试你的技能！

您已读完本文，但您还记得最重要的信息吗？在继续之前，您可以找到一些进一步的测试来验证您是否保留了此信息 — 请参阅测试您的技能：JSON。

概括

在本文中，我们为您提供了在程序中使用 JSON 的简单指南，包括如何创建和解析 JSON，以及如何访问锁定在其中的数据。在下一篇文章中，我们将开始研究面向对象的 JavaScript。

也可以看看

- JSON 参考
- 获取 API 概述
- 使用获取
- HTTP 请求方法
- 带有 ECMA 标准链接的官方 JSON 网站

此页面最后修改于 2023 年 2 月 24 日由MDN 贡献者提供。