> 此页面由社区从英文翻译而来。了解更多并加入 MDN Web Docs 社区。

使用 JSON

JavaScript 对象表示法（JSON）是用于将结构化数据表示为 JavaScript 对象的标准格式，通常用于在网站上表示和传输数据（例如从服务器向客户端发送一些数据，因此可以将其显示在网页

前提：	具备基本的计算机知识、对 HTML 和 CSS 的基本了解、熟悉 JavaScript 基础知识（阅读 JavaScript 第一步和创建代码块）和 JS 面向对象基础（阅读对象介绍 (en-US)）。
目标：	理解 JSON 的数据储存工作原理，创建你的 JSON 对象。

什么是 JSON?

JSON 是一种按照 JavaScript 对象语法的数据格式，这是道格拉斯·克罗克福特 推广的。虽然它是基于 JavaScript 语法，但它独立于 JavaScript，这也是为什么许多程序环境能够读取（解读）和生成 JSON。

JSON 可以作为一个对象或者字符串存在，前者用于解读 JSON 中的数据，后者用于通过网络传输 JSON 数据。这不是一个大事件——JavaScript 提供一个全局的 可访问的 JSON 对象来对这两种数据进行转换。

> **备注：** 将字符串转换为原生对象称为*反序列化*（deserialization），而将原生对象转换为可以通过网络传输的字符串称为*序列化*（serialization）。

一个 JSON 对象可以被储存在它自己的文件中，这基本上就是一个文本文件，扩展名为 `.json`，还有 `application/json` MIME 类型。

JSON 结构

如上所述，JSON 是一个字符串，其格式非常类似于 JavaScript 对象字面量的格式。你可以在 JSON 中包含与标准 JavaScript 对象相同的基本数据类型——字符串、数字、数组、布尔值和其他对象字面量。这使你可以构建一个数据层次结构，如下所示：

JSON

```
{
  "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"
```

```
      ]
    }
  ]
}
```

如果我们把字符串加载到 JavaScript 程序中，并将其解析到一个名为 `superHeroes` 的变量，那么我们就可以使用在 JavaScript 对象基础文章中相同的点/括号表示法来访问其中的数据。例如：

JS

```
superHeroes.hometown;
superHeroes["active"];
```

为了访问层次结构中更深层次的数据，必须将所需的属性名和数组索引链接在一起。例如，访问 members 数组第二个英雄的第三个超能力，可以这样做：

JS

```
superHeroes["members"][1]["powers"][2];
```

1. 首先我们有变量名 `superHeroes`，储存对象。
2. 在对象中我们想访问 `members` 属性，所以我们使用 `["members"]`。
3. `members` 包含有对象数组，我们想要访问第二个元素，所以我们使用 `[1]`。
4. 在对象内，我们想访问 `powers` 属性，所以我们使用 `["powers"]`。
5. `powers` 属性是一个包含英雄技能的数组。我们想要第三个，所以我们使用 `[2]`。

> **备注：** 我们已经在 JSONText.html 实例中让 JSON 对象进入变量中使其可访问（见源代码）。尝试加载它并且在你的浏览器上访问对象数据。

JSON 数组

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

JSON

```
[
  {
```

```
    "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"
    ]
  }
]
```

上面是完全合法的 JSON。你只需要通过数组索引就可以访问数组元素，如 `[0]["powers"][0]`。

其他注意事项

- JSON 是一种纯数据格式，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。单引号无效。
- 甚至一个错位的逗号或分号就可以导致 JSON 文件出错。你应该小心的检查你想使用的数据（虽然计算机生成的 JSON 很少出错，只要生成程序正常工作）。你可以通过像 JSONLint 这样的应用程序来验证 JSON。
- JSON 实际上可以是任何可以有效包含在 JSON 中的数据类型的形式。比如，单个字符串或者数字就是有效的 JSON 对象。
- 与 JavaScript 代码中对象属性可以不加引号不同，JSON 中只有带引号的字符串可以用作属性。

动手练习：一个 JSON 示例

好了，让我们通过运行这个示例来展示我们如何利用 JSON 数据。

开始吧

首先，拷贝我们的 heroes.html 和 style.css 文件。后者包含了用于页面的简单的 CSS，前者包含了简单的 HTML body，以及一个 `<script>` 元素，其中包含我们将在练习中编写的

JavaScript 代码：

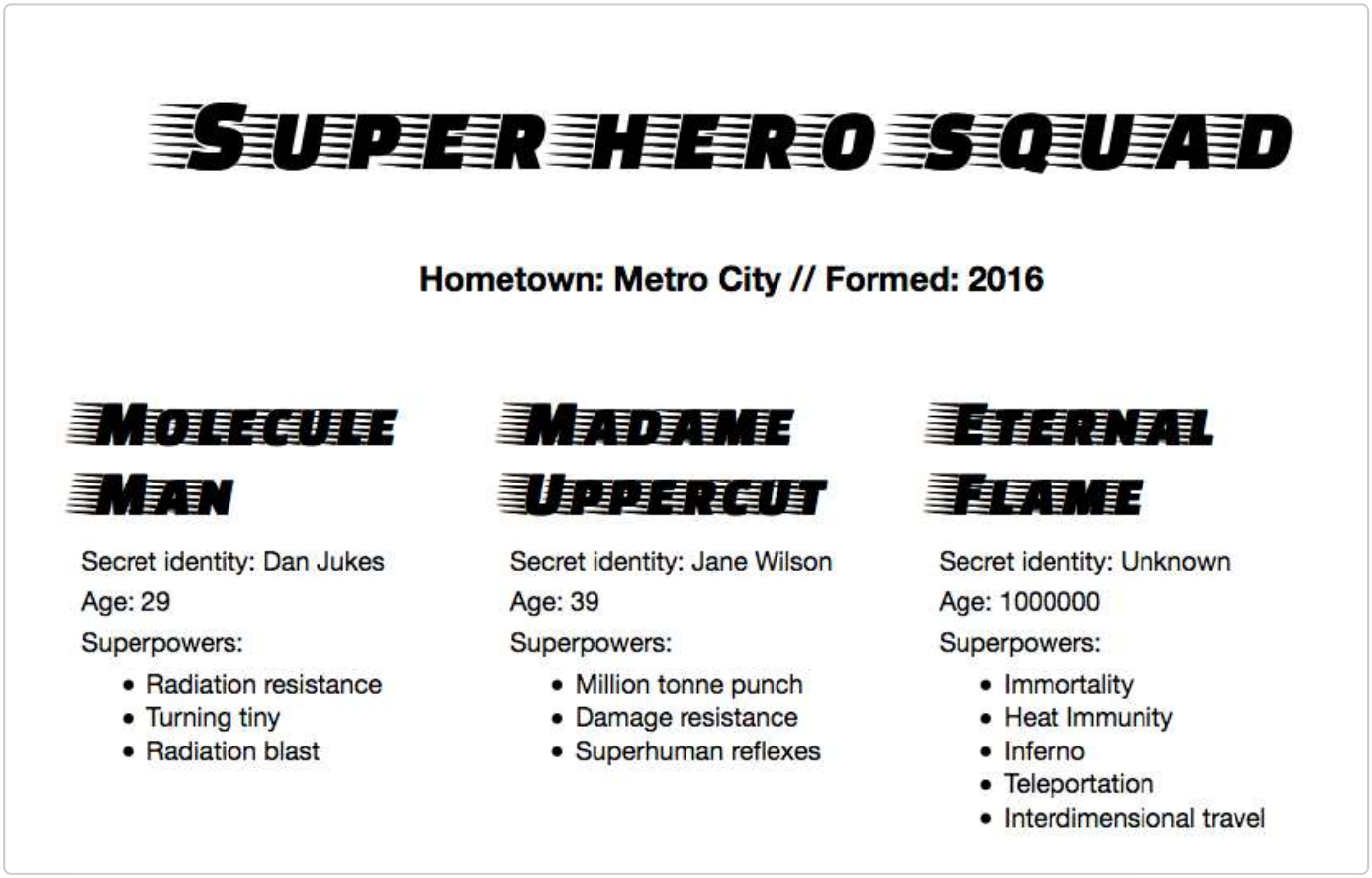
```HTML
<header>
...
</header>

<section>
...
</section>

<script>
...
</script>
```

我们已经把 JSON 数据放在了 GitHub 上面：https://mdn.github.io/learning-area/javascript/oojs/json/superheroes.json

我们准备把它加载到我们的页面中，然后使用漂亮的 DOM 操作来展示它，就像这样：



顶层函数

顶层函数（top-level function）的代码如下所示：

```
JS
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);
}
```

为了获取 JSON 数据，我们使用了名为 Fetch 的 API。 该 API 允许我们通过 JavaScript 进行网络请求，从服务器检索资源（如图像、文本、JSON，甚至 HTML 片段），这意味着我们可以仅更新页面的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 数据：

- 我们声明了 `requestURL` 变量以存储 GitHub 的 URL
- 我们使用该 URL 初始化一个新的 `Request` 对象。
- 我们使用 `fetch()` 函数进行网络请求，它返回一个 `Response` 对象
- 我们使用 `Response` 对象的 `json()` 函数将响应作为 JSON 获取。

> **备注：** `fetch()` API 是**异步**的。我们将在下一个模块中详细了解有关异步函数的知识，但现在我们只需知道需要在使用 fetch API 的函数名称之前添加 `async` 关键字，并在任何异步函数的调用之前添加 `await` 关键字。

在这一切之后， `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——第一个函数用正确的数据填充 `<header>`，而第二个函数为团队中的每个英雄创建一个信息卡，并将其插入到 `<section>` 中。

填充 header

现在我们已经获得我们的 JSON 数据，让我们利用它来写两个我们使用的函数。首先，添加下面的代码于之前的代码下方：

```
JS
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);
}
```

这里我们首先使用 `createElement()` 创建一个 `h1` 元素，然后将其 `textContent` 设置为对象的 `squadName` 属性，接着使用 `appendChild()` 将其添加到页眉中。然后，我们使用类似的操作来创建一个段落：创建并设置其文本内容，再将其附加到页眉。唯一的区别在于，它的文本设置为一个包含对象的 `homeTown` 和 `formed` 属性的模板字面量。

创建英雄信息卡片

接下来，添加如下的函数到代码底部，这个函数创建和展示了超级英雄的卡片：

```
JS
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，我们保存了 JSON 的 `members` 属性作为一个变量。这个数组含有多个带有英雄信息的对象。

接下来，我们使用一个循环来，遍历每个元素。对于每一个元素，我们：

1. 创建几个元素：一个 `<article>`、一个 `<h2>`、三个 `<p>` 和一个 `<ul>`。
2. 设置 `<h2>` 为当前英雄的 `name`。
3. 使用他们的 `secretIdentity`、 `age`，以及“Superpowers:”介绍信息列表来填充三个段落。
4. 保存 `powers` 属性于另一个变量 `superPowers` ——包含英雄的超能力的列表。
5. 使用另一个循环来遍历当前的英雄的超能力，对于每一个元素我们创建 `<li>` 元素，并放入超能力，然后使用 `appendChild()` 把 `listItem` 放入 `<ul>` 元素（`myList`）中。
6. 最后一件事情是将 `<h2>`、 `<p>` 和 `<ul>` 追加到 `<article>`（`myArticle`）中。然后将 `<article>` 追加到 `<section>`。追加的顺序很重要，因为它们将被展示在 HTML 中。

> **备注：** 如有疑难，试试引用我们的 heroes-finished.html 代码（也可以查看运行实例）。

> **备注：** 如果你对访问 JSON 对象的点/括号表示法有困扰。获得文件 superheroes.json ，在你的编辑器中打开并参考我们的 JS 代码将会有所帮助。你还应该参考我们的 JavaScript 对象基础文章，了解关于点和括号表示法的更多信息。

调用顶层函数

最后，我们需要调用顶层函数 `populate()`：

JS

```
populate();
```

对象和文本间的转换

上面的示例在访问 JavaScript 对象方面很简单，因为我们直接使用 `response.json()` 将网络响应转换为了 JavaScript 对象。

但是有时候我们没有那么幸运，我们接收到一些 字符串作为 JSON 数据，然后我们想要将它转换为对象。当我们想要发送 JSON 数据作为信息，我们将需要转换它为字符串，我们经常需要正确的转换数据，幸运的是，这两个问题在 web 环境中是那么普遍以至于浏览器拥有一个内建的 JSON，包含以下两个方法。

- `parse()`：以文本字符串形式接受 JSON 对象作为参数，并返回相应的对象。
- `stringify()`：接收一个对象作为参数，返回一个对应的 JSON 字符串。

你可以看看我们 heroes-finished-json-parse.html 示例的第一个操作（查看源代码 ），这做了一件与我们之前一模一样的事情，除了：

- 我们通过调用 `text()` 方法将响应作为文本获取，而不是 JSON
- 然后我们使用 `parse()` 将文本转换为 JavaScript 对象。

关键片段如下：

JS

```
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);
}
```

正如你所想， `stringify()` 做相反的事情。尝试将下面的代码输入你的浏览器 JS 控制台来看看会发生什么：

JS

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

这儿我们创建了一个 JavaScript 对象，接着检查了它包含了什么，然后用 `stringify()` 将它转换成 JSON 字符串，最后保存返回值作为变量并再一次检查。

技能测试！

你已经到达本文的末尾，但你是否记住了最重要的信息呢？在继续之前，你可以进行一些进一步的测试，以验证你是否记住了这些信息——请参阅技能测试： JSON (en-US)。

总结

在这个文章中，我们给了你一个简单的示例来在自己的程序中使用 JSON，包括创建和处理 JSON，还有如何访问 JSON 内的数据。在下一篇文章中我们将开始关注 JS 中的面向对象内容。

参见

- JSON 对象
- Fetch API
- 使用 Fetch

- HTTP 请求方法
- JSON 官网（含 ECMA 标准链接）

Help improve MDN

Was this page helpful to you?

Yes No

Learn how to contribute.

This page was last modified on 2023年11月30日 by MDN contributors.