

ДИСЦИПЛИНА	Интеграция информационных систем с использованием API и микросервисов
ПОДРАЗДЕЛЕНИЕ	ПИШ СВЧ-электроники
ВИД УЧЕБНОГО МАТЕРИАЛА	Методические указания к практическим занятиям
ПРЕПОДАВАТЕЛЬ	Астафьев Рустам Уралович
СЕМЕСТР	1 семестр, 2024/2025 уч. год

Практическое занятие №5. JSON и получение данных	2
Применение JSON.....	2
Структура JSON	2
Рассмотрим работу с форматом JSON на примере.	7

Практическое занятие №5. JSON и получение данных

Применение JSON

JSON — самый популярный формат обмена данными между приложениями.

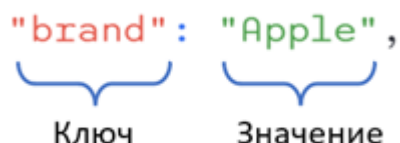
JSON (англ. JavaScript Object Notation) — текстовый формат обмена данными, основанный на JavaScript. Но при этом формат независим от JS и может использоваться в любом языке программирования.

Структура JSON

Данные, «упакованные» в формат JSON имеют следующий вид (пример):

```
1 {  
2   "brand": "Apple",  
3   "model": "iPhone 11 Pro",  
4   "isAvailable": true,  
5   "display": 5.8,  
6   "memories": [64, 256, 512],  
7   "features": {  
8     "tripleCamera": true,  
9     "faceId": true,  
10    "touchId": false,  
11    "eSIM": true  
12  }  
13 }
```

JSON состоит из пар ключ-значение (наименование параметра — значение параметра). Пары разделяются между собой запятыми, а ключ отделяется от значения через двоеточие.



"brand": "Apple",

Ключ Значение

Ключом может быть только строка, обернутая в двойные кавычки. А вот значением — почти всё что угодно:

- Строка в двойных кавычках — "I love JSON!";
- Число — 21;
- Логическое значение — true;
- Массив — [18, true, "lost", [4, 8, 15, 16, 23, 42]];

Все пары «ключ-значение» помещаются в JSON внутрь фигурных скобок.

Ниже приведен пример простых данных в JSON:

```
{  
  "name": "Alex C",  
  "age": 2,  
  "city": "Houston"  
}
```

JSON основан на JavaScript, но является независимой от языка спецификацией для данных и может использоваться почти с любым языком программирования.

JSON используется для того, чтобы получить данные от сервера. Типичная схема работы:

1. Отправляем запрос на сервер;
2. Ждём ответ;
3. Получаем JSON с набором данных;
4. Используем данные, обращаясь к ним по ключу.

Допустимые данные в JSON возможны в 2 разных форматах:

- Набор пар «ключ-значение» в фигурных скобках {...}. Это показано в примере выше.

- Упорядоченные списки пар «ключ-значение», разделенных запятой (,) и заключенных в квадратные скобки [...]. См. пример ниже:

```
[
  {
    "name": "Alex C",
    "age": 2,
    "city": "Houston"
  },
  {
    "name": "John G",
    "age": 40,
    "city": "Washington"
  },
  {
    "name": "Bala T",
    "age": 22,
    "city": "Bangalore"
  }
]
```

В данном примере во внешних квадратных скобках заключена информация о трех людях, характеризующихся одинаковым набором параметров:

Информация о первом человеке	{	"name": "Alex C",
		"age": 2,
		"city": "Houston"
	},	
Информация о втором человеке	{	"name": "John G",
		"age": 40,
		"city": "Washington"
	},	
Информация о третьем человеке	{	"name": "Bala T",
		"age": 22,
		"city": "Bangalore"
	}	
	}	

Сохранять данные JSON можно в файле с расширением .json. Давайте создадим файл employee.json с атрибутами сотрудника. Они представлены в виде ключей и значений.

```
{
  "name": "Иван Петров",
```

```
"id": "E00245",  
"role": ["Разработчик", "Исследователь"],  
"age": 33,  
"doj": "11-12-2019",  
"married": false,  
"address": {  
  "street": "32, пр-т Вернадского",  
  "city": "Москва",  
  "country": "Россия"  
},  
"referred-by": "E0012"  
}
```

В примере выше присутствуют следующие атрибуты сотрудника:

- name – имя сотрудника. Значение в строковом формате (String).

Оно указано в двойных кавычках.

- id – уникальный идентификатор сотрудника. Опять же, в строковом формате.

- role – роли, которые сотрудник выполняет в организации. Таких ролей может быть несколько, поэтому лучше перечислять эти данные в формате массива (Array). Если ролей несколько, то каждая из них помещается в собственные кавычки, а все роли – в квадратные скобки (массив).

- age – текущий возраст сотрудника. Это числовое значение (Number), без кавычек

- doj – дата найма сотрудника. Поскольку это дата, ее добавляют в двойных кавычках и обрабатывают как строку.

- married – замужем/женат ли сотрудник? Ответом может быть да/нет (то есть true или false), так что это логический формат (Boolean).

- `address` – адрес сотрудника. Может состоять из нескольких частей: улица, город, страна, индекс и т.д. Такое поле лучше представлять в виде встроеного JSON-описания (набором пар «ключ-значение»).

- `referred-by` – идентификатор сотрудника, который порекомендовал этого человека на должность в организацию. Если сотрудник пришел по рекомендации, то атрибут имеет значение. В остальных случаях поле остается пустым, т.е. вместо идентификатора в кавычках пишется `null` без кавычек.

Теперь давайте создадим набор данных по сотрудникам в формате JSON. Если мы хотим добавить несколько записей о разных сотрудниках, то необходимо прописать их в квадратных скобках [...].

```
[
  {
    "name": "Иван Петров",
    "id": "E00245",
    "role": ["Разработчик", "Исследователь"],
    "age": 33,
    "doj": "11-12-2019",
    "married": false,
    "address": {
      "street": "32, пр-т Вернадского",
      "city": "Москва",
      "country": "Россия"
    },
    "referred-by": "E0012"
  },
  {
    "name": "Евгений Гусев",
```

```
"id": "E01245",  
"role": ["Аналитик"],  
"age": 43,  
"doj": "17-10-2023",  
"married": true,  
"address": {  
  "street": "21, ул. Академика Королёва",  
  "city": "Москва",  
  "country": "Россия"  
},  
"referred-by": null  
}  
]
```

Обратите внимание на значение атрибута `referred-by` для сотрудника Евгения Гусева. Оно пустое. То есть никто из сотрудников не давал ему рекомендаций.

Рассмотрим работу с форматом JSON на примере.

Для данного примера рассмотрим материалы с гитхаба по ссылке к занятию. Для обучения работе с JSON необходимо открыть в браузере файлы практического занятия. Файл `style.css` содержит простой CSS для стилизации нашей страницы, в то время как `index.html` содержит очень простой HTML-код и секцию `<script>`, которую мы будем развивать в примере.

Добавим внутрь тегов `<script>` строки:

```
var header = document.querySelector('header');  
var section = document.querySelector('section');
```

Эти строки захватывают ссылки на элементы `<header>` и `<section>` и сохраняют их в переменных. Данные JSON доступны по ссылке в интернете:

`https://mdn.github.io/learning-area/javascript/oojs/json/superheroes.json`

Чтобы получить JSON, мы будем использовать JS-компонент, называемый XMLHttpRequest (часто называемый XHR). Это очень полезный объект JavaScript, который позволяет нам делать сетевые запросы для извлечения ресурсов с сервера через JavaScript (например, изображения, текст, JSON, даже фрагменты HTML), что означает, что мы можем обновлять небольшие разделы контента без необходимости перезагрузки всей страницы.

1. Начнём с того, что мы собираемся сохранить URL-адрес JSON, который мы хотим получить в переменной. Добавьте нижеследующий код JavaScript в секцию `<script>`:

```
var requestURL = 'https://mdn.github.io/learning-area/javascript/oojs/json/superheroes.json';
```

2. Чтобы создать запрос, нам нужно создать новый экземпляр объекта запроса из конструктора XMLHttpRequest, используя ключевое слово `new`. Добавьте следующую ниже свою последнюю строку:

```
var request = new XMLHttpRequest();
```

3. Теперь нам нужно открыть новый запрос, используя метод `open()`. Добавьте следующую строку:

```
request.open('GET', requestURL);
```

Указываем два обязательных для этого примера параметра:

- Метод HTTP, который следует использовать при выполнении сетевого запроса. В этом случае GET самый подходящий, так как мы просто извлекаем некоторые простые данные.

- URL-адрес для запроса - это URL-адрес файла JSON, который мы сохранили ранее.

4. Затем добавьте следующие две строки: здесь мы устанавливаем `responseType` в `JSON`, так что `XHR` знает, что сервер будет возвращать `JSON` и, что это должно быть преобразовано в объект, понятный интерпретатору `JavaScript`. Затем мы отправляем запрос методом `send()`:

```
request.responseType = 'json';  
request.send();
```

5. Добавьте следующий код ниже вашего предыдущего кода:

```
request.onload = function() {  
    var bestEmp = request.response;  
    populateHeader(bestEmp);  
    showHeroes(bestEmp);  
}
```

Здесь мы сохраняем ответ на наш запрос (доступный в свойстве `response`) в переменной `bestEmp`; эта переменная теперь будет содержать объект `JavaScript`, основанный на `JSON`! Затем мы передаём этот объект двум вызовам функций - первый из них заполнит `<header>` правильными данными, а второй создаст информационную карту для каждого работника в команде и вставляет её в `<section>`.

Мы свернули код в обработчик событий, который запускается, когда событие загрузки запускается в объекте запроса - это связано с тем, что событие загрузки запускается, когда ответ успешно возвращается; поступая таким образом, это гарантия того, что `request.response` определённо будет доступен, когда мы начнём работу с ним.

Заполнение заголовка `<header>`

Теперь мы извлекли данные `JSON` и превратили его в объект `JavaScript`, давайте воспользуемся им, написав две функции, на которые мы ссылались выше. Прежде всего, добавьте следующее определение функции ниже предыдущего кода:

```
function populateHeader(jsonObj) {
```

```
var header = document.querySelector('header');
var myH1 = document.createElement('h1');
myH1.textContent = jsonObj['squadName'];
header.appendChild(myH1);
```

```
var myPara = document.createElement('p');
myPara.textContent = 'Hometown: ' + jsonObj['homeTown'] + ' //
Formed: ' + jsonObj['formed'];
header.appendChild(myPara);
}
```

Заполнение «карточек» супергероев

Добавьте следующую функцию внизу кода, которая создаёт и отображает карты супергероев

```
function showHeroes(jsonObj) {
  var section = document.querySelector('section');
  var heroes = jsonObj['members'];
  for (var i = 0; i < heroes.length; i++) {
    var myArticle = document.createElement('article');
    var myH2 = document.createElement('h2');
    var myPara1 = document.createElement('p');
    var myPara2 = document.createElement('p');
    var myPara3 = document.createElement('p');
    var myList = document.createElement('ul');
    myH2.textContent = heroes[i].name;
    myPara1.textContent = 'Secret identity: ' + heroes[i].secretIdentity;
    myPara2.textContent = 'Age: ' + heroes[i].age;
    myPara3.textContent = 'Superpowers:';
    var superPowers = heroes[i].powers;
    for (var j = 0; j < superPowers.length; j++) {
```

```

    var listItem = document.createElement('li');
    listItem.textContent = superPowers[j];
    myList.appendChild(listItem);
  }
  myArticle.appendChild(myH2);
  myArticle.appendChild(myPara1);
  myArticle.appendChild(myPara2);
  myArticle.appendChild(myPara3);
  myArticle.appendChild(myList);
  section.appendChild(myArticle);
}
}

```

Для начала сохраним свойство `members` объекта JSON в новой переменной. Этот массив содержит несколько объектов, которые содержат информацию для каждого героя.

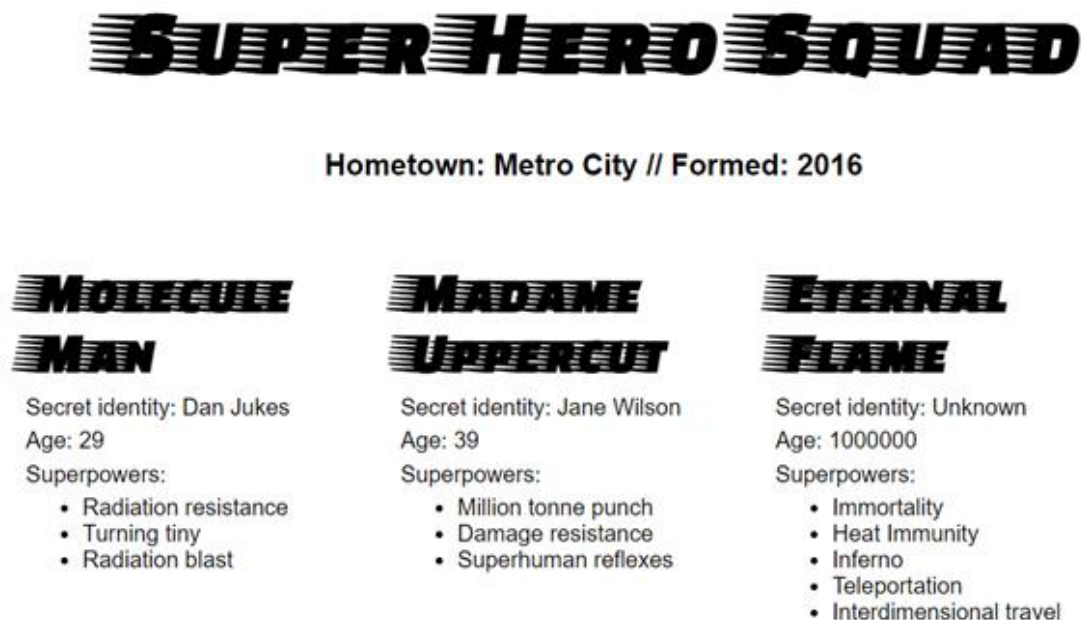
Затем мы используем цикл `for` для циклического прохождения каждого объекта в массиве. Для каждого из них мы:

1. Создаём несколько новых элементов: `<article>`, `<h2>`, три `<p>` и ``.
2. Устанавливаем `<h2>`, чтобы содержать `name` текущего героя.
3. Заполняем три абзаца своей `secretIdentity`, `age` и строкой, в которой говорится: «Суперспособности:», чтобы ввести информацию в список.
4. Сохраняем свойство `powers` в другой новой переменной под названием `superPowers` - где содержится массив, в котором перечислены сверхспособности текущего героя.
5. Используем другой цикл `for`, чтобы прокрутить сверхспособности текущего героя, для каждого из них мы создаём элемент

, помещаем в него сверхспособности, а затем помещаем listItem внутри элемента (myList) с помощью appendChild().

6. Последнее, что мы делаем, это добавляем <h2>, <p> и внутри <article> (myArticle), а затем добавляем <article> в <section>. Важное значение имеет порядок, в котором добавляются элементы, так как это порядок, который они будут отображать внутри HTML.

Если Вы всё сделали правильно, то результат будет следующим:



Полезные ссылки и источники

1. <https://github.com/astafiev-rustam/integration-of-information-systems/tree/Practice-1-5>