Работа с JSON

Введение в JSON

JSON (сокращение от JavaScript Object Notation) — это формат передачи данных. Как можно понять из названия, JSON произошел из JavaScript, но он также доступен для использования во многих других языках, включая Python, Ruby, PHP и Java. В англоязычным странах название формата в основном произносят как имя Джéйсон (Jason), а в русскоязычных — преимущественно с ударением на «о»: Джисóн.

Сам по себе JSON использует расширение .json. Когда же он определяется в других файловых форматах, как .html, он появляется в кавычках как JSON-строка или может быть объектом, назначенным на переменную. Такой формат легко передавать между сервером и клиентской частью (например, браузером).

Легкочитаемый и компактный, JSON представляет собой хорошую альтернативу XML и требует куда меньше форматирования контента. Давайте разберёмся с данными, которые вы можете использовать в JSON, а также с основной структурой и синтаксисом этого формата.

В программах на JavaScript формат JSON обычно используют в следующих случаях:

1. Хранение данных.
2. Генерирование структур данных из пользовательского ввода.
3. Обмен данными между сервером и клиентом.
4. Настройка и проверка данных.

Синтаксис и структура

Объект JSON хранит данные в формате «ключ-значение» и обычно рендерится в фигурных скобках. Когда вы работаете с JSON, вы скорее всего видите JSON-объекты в файле .json, но они также могут быть представлены JSON-объектом или строком уже в контексте самой программы.

Вот так выглядит JSON-объект:

```
{
    "first_name": "Sammy",
    "last_name": "Shark",
    "location": "Ocean",
```

```
    "online": true,

    "followers": 987

}
```

Хоть это и короткий пример, и JSON мог быть гораздо больше, он показывает, что этот формат представляет данные в виде блока ключей и значений, заключённого в фигурные скобки. Большая часть данных, используемых в JSON, заключаются в JSON-объекты.

Пары «ключ-значение» разделены двоеточием: например, "key" : "value". Между собой пары разделены запятыми, так что середина JSON выглядит так: "key" : "value", "key" : "value", "key" : "value". В нашем примере выше первая пара ключевых значений — это "first_name" : "Sammy".

```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 27,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": [
      "Catherine",
      "Thomas",
      "Trevor"
  ],
  "spouse": null
}
```

На данном скриншоте вы видите пример JSON-модели в связке ключа и значения.

Ключи в JSON находятся с левой стороны от двоеточия. Их нужно заключать в кавычки, как это сделано в случае с "key". Именем ключа может выступать любая строка. В каждом объекте ключи должны быть уникальными. Ключевая строка может содержать пробелы, как в "first

name", но это может усложнить получение доступа к ним в процессе разработки, так что лучшим вариантом будет использовать нижнее подчеркивание: "first_name".

JSON-значения находятся справа от двоеточия и должны принадлежать к одному из шести типов данных: быть строкой, числом, объектом, массивом, булевым значением или null.

В некоторых случаях значения также могут состоять из сложных типов данных, таких как JSON-объект или массив.

Каждый тип данных, который передаётся как значения в JSON, поддерживает свой синтаксис, поэтому строки пишут в кавычках, а цифры нет.

Хотя файлах .json пары ключ-значение обычно разносятся на несколько строк, JSON также может быть написан в одну строку:

```
{ "first_name": "Sammy", "last_name": "Shark", "online": true, }
```

Такой подход распространён в файлах других форматов или при работе с JSON-строкой.

Работа с JSON в многострочном формате зачастую делает его более удобочитаемым, особенно когда вы пытаетесь справиться с большим набором данных. Поскольку JSON игнорирует пробелы между своими элементами, вы можете использовать пробелы как дополнительный разделитель, который упростит восприятие данных человеком:

```
{
    "first_name": "Sammy",
    "last_name": "Shark",
    "online": true
}
```

Очень важно помнить, что при всём визуальном сходстве, объекты JSON по формату отличаются от объектов JavaScript. И хотя вы можете использовать функции внутри объектов JavaScript, вы не можете использовать их как значения в JSON.

Самое важное свойство формата JSON — он «понятен» самым разным языкам программирования, поэтому без труда передаётся между приложениями. Объекты JavaScript могут работать только напрямую через JavaScript.

Пока что мы видели JSON в самых простых случаях, но он может стать иерархическим и сложным, включая в себя вложенные объекты и массивы. Сейчас мы рассмотрим более сложный пример JSON.

Итоги

Давайте подведём промежуточные итоги. Для чего нам необходим JSON? Мы знаем, что данные на странице могут формироваться с помощью HTML-разметки, и любые правки, которые мы вносим, мы добавляем в разметку. Но это в статических сайтах, а в динамических все данные размещаются на сервере, и нам необходим удобный формат: именно за эту часть и отвечает JSON. Мы получаем данные с сервера в удобном для нас формате, чтобы в дальнейшем использовать их для формирования страницы.

Работаем с комплексными типами в JSON

В дополнение к вложенным массивам, JSON может содержать другие вложенные объекты. Такие объекты и массивы будут передаваться как значения, назначенные ключам, и будут представлять собой связку ключ-значение.

Вложенные объекты

Для каждого из четырёх пользователей ("sammy", "jesse", "drew", "jamie") есть вложенный передающий значения JSON-объект, со своими собственными вложенными ключами "username" и "location". Первый вложенный JSON:

```
{
    "sammy": {
        "username": "SammyShark",
        "location": "Indian Ocean",
        "online": true,
        "followers": 987
    },
    "jesse": {
        "username": "JesseOctopus",
        "location": "Pacific Ocean",
        "online": false,
        "followers": 432
    },
```

```
    "drew": {
        "username": "DrewSquid",
        "location": "Atlantic Ocean",
        "online": false,
        "followers": 321
    },
    "jamie": {
        "username": "JamieMantisShrimp",
        "location": "Pacific Ocean",
        "online": true,
        "followers": 654
    }
}
```

В примере выше фигурные скобки везде используются для формирования вложенного JSON-объекта с ассоциированными именами пользователей и данными локаций для каждого из них. Как и с любым другим значением при использовании объектов, двоеточие используется для разделения элементов.

Вложенные массивы

В формате JSON данные также могут быть вложены с помощью JavaScript-массивов, которые передаются как значения. JavaScript использует для оформления массива квадратные скобки: [ ]. По сути, массивы — это упорядоченные коллекции и могут включать в себя значения совершенно разных типов данных. Пример:

```
{
    "cars": {
        "Nissan": [
            {"model":"Sentra", "doors":4},
            {"model":"Maxima", "doors":4},
            {"model":"Skyline", "doors":2}
        ],
        "Ford": [
            {"model":"Taurus", "doors":4},
            {"model":"Escort", "doors":4}
        ]
    }
}
```

Мы можем использовать массив при работе с большим количеством данных, которые легко сгруппировать. Например, если у вас есть несколько разных сайтов и профайлов в социальных сетях, ассоциированных с одним пользователем.

Пример:

```
{
    "first_name": "Sammy",
    "last_name": "Shark",
    "location": "Ocean",
    "websites": [{
            "description": "work",
            "URL": "https://www.digitalocean.com/"
        },
        {
            "desciption": "tutorials",
            "URL": "https://www.digitalocean.com/community/tutorials"
        }
    ],
    "social_media": [{
```

```
            "description": "twitter",
            "link": "https://twitter.com/digitalocean"
        },
        {
            "description": "facebook",
            "link": "https://www.facebook.com/DigitalOceanCloudHosting"
        },
        {
            "description": "github",
            "link": "https://github.com/digitalocean"
        }
    ]
}
```

Ключи "websites" и "social_media" используют массив для вложения информации о сайтах пользователя и профайлов в социальных сетях. Мы узнаём, что это массивы, по квадратным скобкам.

Использование вложенности в нашем JSON формате позволяет нам работать с наиболее сложными и иерархичными данными. https://jsonplaceholder.typicode.com/comments пример как выглядит иерархичные данные JSON и на сколько они могут быть объемными

Функции в JSON

При работе с JSON очень полезно иметь возможность быстро преобразовать строку в объект и наоборот. В этом разделе мы рассмотрим два метода JSON.

Функция JSON.stringify()

Функция JSON.stringify() преобразовывает объекты JSON в строки.

Строки позволяют упростить обмен данными между сервером и клиентом. К примеру, вы можете собирать настройки пользователей на стороне клиента, а затем передавать их на сервер. После этого вы сможете преобразовать строку в объект с помощью метода JSON.parse().

Рассмотрим объект, присвоенный переменной obj. Попробуйте преобразовать его в строку. Для этого нужно передать функции JSON.stringify() переменную obj. Присвойте эту строку переменной s.

```
const obj = {"first_name" : "John", "last_name" : "Smith", "location" : "London"}

const s = JSON.stringify(obj)
```

Теперь объект стал строкой и является значением переменной **s**:

```
`{"first_name" : "John", "last_name" : "Smith", "location" : "London"}`
```

Функция JSON.parse()

Строки удобны при обмене данными, но потом их нужно снова преобразовать в объекты. Для этого используется функция JSON.parse().

Примечание: Чтобы преобразовать текст в объект, используйте функцию **eval()**.

Теперь попробуйте преобразовать значение функции **s** в объект и присвоить его новой переменной:

```
const data = JSON.parse(s);
```

Теперь у вас есть объект **data**, идентичный объекту **obj**.

Рассмотрим ещё один пример. Функцию JSON.parse() можно использовать в контексте файла HTML:

```
<!DOCTYPE html>

<html>

<body>

   <p id="user"></p>

   <script>

        const userInfo = '{"first_name" : "John", "last_name" : "Smith", "location" : "London"}';

       const obj = JSON.parse(s);
```

```
        document.getElementById("user").innerHTML =
            "Name: " + obj.first_name + " " + obj.last_name + "<br>" +
            "Location: " + obj.location;
    </script>
</body>
</html>
```

Результат:

Name: John Smith

Location: London

В HTML-файле JSON-строка **userInfo** преобразуется в объект, который извлекается в финальный рендеринг с помощью точечной нотации.

Заключение

JSON — простой формат, который позволяет легко делиться данными, хранить их и работать с ними. Как формат JSON переживает растущую поддержку API, включая и Twitter API.

Поскольку вы вряд ли будете создавать собственные файлы .json, но будете получать их из других источников, очень важно меньше думать о JSON-структуре и больше о том, как лучше применять этот формат в ваших программах.