

JSON представление данных

- Введение
- Структура формата JSON
- Объект JSON
- Метод JSON.stringify()
- Метод JSON.parse()
- Вложенность в строках JSON

Введение

Что такое JSON

JSON (JavaScript Object Notation, объектная нотация JavaScript) — это стандартный **текстовый формат** для представления структурированных данных на основе синтаксиса объектов JavaScript. По сути, формат JSON представляет собою **обычную строку**.

Несмотря на то, что он очень похож на синтаксис литерала объекта JavaScript, его можно использовать независимо от JavaScript, и многие среды программирования имеют возможность читать (анализировать) и генерировать JSON.

JSON существует в виде строки — это удобно, когда вы хотите **передавать данные по сети**. Обычно используется для обмена данными в веб-приложениях. Обмен заключается в отправке некоторых данных с сервера клиенту, чтобы их можно было отобразить на веб-странице, и наоборот, от клиента на сервер, для обеспечения обратной связи между клиентом и приложением.

Еще раз. При веб-разработке JSON очень часто применяется в качестве формата для **передачи информации** от веб-сервера клиенту (веб-

браузеру) и наоборот при асинхронных запросах (запросы без перезагрузки страницы).

Для **получения доступа к данным** строку JSON необходимо обратно преобразовать в **исходный объект**. Если преобразование выполняется на стороне клиента, то в объект **JavaScript (JS)**, если на стороне сервера, то в объект **PHP** (если серверный язык – PHP).

Таким образом.

- Для передачи данных используем **JSON-строку**.
- Для работы с данными **JavaScript-объект** или **PHP-объект**.
- Для преобразования данных из строки в объект и наоборот JavaScript предоставляет специальные **методы глобального объекта JSON**.

При помощи простых правил формирования JSON-конструкций, разработчик может обеспечить простой и надежный способ хранения любого вида информации, будь то обычное число, целые строки или огромное количество различных объектов, выраженных в простом тексте.

Строка JSON может храниться в собственном файле, который в основном представляет собой просто текстовый файл с расширением **.json**.

Где используется JSON

После того как файл создан, содержащиеся в нем JSON-данные довольно легко перенаправить в другое приложение Сети через любые пути передачи данных. Это связано с тем, что JSON-строка представляет собой обычный текст.

Как выглядит этот процесс? Его можно представить в виде двух шагов.

- **На первом шаге**, сервер, по запросу пришедшему ему от клиента, формирует некоторый **набор данных**, который затем **упаковывает в строку формата JSON**. Завершается работа на сервере отправкой JSON данных в качестве результата клиенту.

- **На втором шаге**, клиент **получает в качестве ответа от сервера строку JSON** и распаковывает её, т.е. переводит в **JavaScript объект**. После этого на клиенте выполняются дальнейшие с ними действия, например, выводятся на страницу.

Это один из примеров использования формата JSON. Но его применение не ограничивается только этим сценарием, их очень много и не только в веб.

Некоторые области применения формата:

- Хранение данных.
- Конфигурация и проверка данных.
- Генерация структур данных из пользовательского ввода.
- Передача данных с клиента на сервер, с сервера на клиент и между серверами.

Примечание. Работа с JSON в JS обычно осуществляется в двух направлениях:

- Преобразование JS объекта в JSON-строку (для **передачи** по сети) называется **сериализацией**.
- Преобразование JSON-строки в объект JS (для **доступа** к данным) называется **десериализацией**. Данный процесс ещё называют **парсингом** (распарсить, значит десериализовать).

В JSON, в отличие от XML и YAML, данные организованы также как в объекте JavaScript. Но JSON – это не объект, а строка. При этом не любой объект JavaScript может быть переведён один к одному в JSON. Например, если у объекта есть методы, то они при преобразовании в строку JSON будут проигнорированы и не включены в неё.

Еще раз. Если у объекта есть **методы**, то при преобразовании в строку JSON они **будут проигнорированы**.

Методы объектов будем изучать в отдельной теме.

Структура формата JSON

Типы данных JSON

В JSON типы данных можно разделить на две категории: **простые** и **сложные**.

К категории простых типов (примитивов) относятся:

- **string** – строки
- **number** – числа
- **boolean** – логические (булевы) значения
- **null**

К сложным типам относятся:

- **object** – объекты
- **array** – массивы

Синтаксис JSON заимствован из JavaScript, поэтому для представления значений простых и сложных типов используется тот же синтаксис, что и в JavaScript.

Простые значения

Простейший пример JSON-кода – любое значение простого типа. Обратите внимание на определение строк. **Строки должны быть в кавычках.**

example_1. Простые значения JSON

```
<script>

    // определение в JS

    var num = 25;

    var str = "string";

    var bool = true;

    var nul = null;
```

```
// определение в JSON

var numJSON = 25;

var strJSON = "\"string\"";

var boolJSON = true;

var nullJSON = null;

</script>
```

В JSON строки должны быть заключены **только в двойные кавычки**.

Важно. Использование одинарных кавычек приводит к синтаксической ошибке.

Объекты

Структура объекта JSON практически ничем не отличается от записи JS объекта.

Объект JSON представляет собой заключённый в фигурные скобки список из нуля или более пар **"ключ": значение**. В этой паре ключ отделяется от значения с помощью **знака двоеточия (:)**, а одна пара от другой - с помощью **запятой (,)**.

Имена ключей (свойств объектов) обязательно должны быть заключены в **двойные кавычки**.

Важно. Отсутствие **двойных** кавычек или **использование одинарных** кавычек в имени свойства является ошибкой.

Это важное отличие JSON от JS объекта. В JS объекте ключ (имя свойства) заключать в двойные кавычки не обязательно.

Свойства объекта могут содержать значения любого типа (простого или сложного). Это позволяет разработчику строить **иерархию данных**.

example_2. Объекты JSON

```
<script>

    // определение JS-объекта

    let personJS = {

        surname : "Бородина",

        name : "Ксения",

        patronymic : "Алексеевна",

        post : "Преподаватель"

    };

    // определение JSON-объекта

    // для записи в несколько строк используем обратные кавычки

    let personJSON = `

    {

        "surname": "Бородина",

        "name": "Ксения",

        "patronymic": "Алексеевна",

        "post": "Преподаватель"

    }`;

</script>
```

Чтобы не усложнять доступ к данным, при задании ключам имён лучше придерживаться тех же правил, что и при именовании переменных.

Важно. Значение ключа в JSON не может быть **функцией** или **датой** (объектом типа Date).

Массивы

Массив JSON представляет собой заключённый в квадратные скобки список из нуля или более значений, разделённых запятыми. **Массив может содержать значения любого типа** (простого или сложного).

example_3. Массивы JSON

```
<script>

    // определение JS-массива

    let coursesJS = [

        "Профессия веб-разработчик",

        "Fullstack разработчик на Python",

        "Серверное программирование на PHP"

    ];

    // определение JSON-массива

    // для записи в несколько строк используем обратные кавычки

    let coursesJSON = `[

        "Профессия веб-разработчик",

        "Fullstack разработчик на Python",

        "Серверное программирование на PHP"

    ]`;

</script>
```

Иерархия данных в формате JSON

Пример более сложного JSON формата, хранящего в качестве значения ключа другие **сложные типы данных** (объект и массив) приведен в example_4.

example_4. JSON может хранить вложенные структуры

```
<script>

    // определение сложного JS-объекта
```

```
let personJS = {  
    surname : "Бородина",  
    name : "Ксения",  
    patronymic : "Алексеевна",  
    post : "Преподаватель",  
    category : "Соответствие занимаемой должности",  
    experience : {"total" : 21.3, "college" : 14.5},  
    courses : ["Профессия веб-разработчик", "Fullstack  
разработчик на Python"]  
};  
  
// определение сложного формата JSON-строки  
let personJSON = `{  
    "surname" : "Бородина",  
    "name" : "Ксения",  
    "patronymic" : "Алексеевна",  
    "post" : "Преподаватель",  
    "category" : "Соответствие занимаемой должности",  
    "experience" : {"total" : 21.3, "college" : 14.5},  
    "courses" : ["Профессия веб-разработчик", "Fullstack  
разработчик на Python"]  
}`;  
  
</script>
```

Каждый из **типов данных**, которые передаются в JSON как значения, будут поддерживать свой **собственный синтаксис**. При этом:

- вложенные объекты берутся в **фигурные** скобки;
- массивы берутся в **прямоугольные** скобки.

Примечание. JSON не допускает использование внутри своей структуры комментариев.

Пример JSON строки, с которыми, скорее всего, вы будете иметь дело в реальных приложениях представлен в следующем примере.

example_5. Пример сложного формата JSON

`<script>`

```
// определение сложного формата JSON-строки

let personJSON = `
{
    "surname" : "Бородина",
    "name" : "Ксения",
    "patronymic" : "Алексеевна",
    "post" : "Преподаватель",
    "category" : "Соответствие занимаемой должности",
    "experience" : {"total" : 21.3, "college" : 14.5},
    "courses" : ["Профессия веб-разработчик", "Fullstack
разработчик на Python"],
    "education" : [
        {
            "institution" : "Санкт-Петербургский
государственный университет (СПбГУ)",
            "qualification" : "Информационные системы и
технологии",
            "specialty" : "Безопасность киберфизических
систем",
            "year_receipts" : 1993,
```

```

        "year_release" : 1998
    },
    {
        "institution" : "Московский государственный
        университет имени М.В.Ломоносова",
        "qualification" : "Факультет вычислительной
        математики и кибернетики",
        "specialty" : "Прикладная математика и
        информатика",
        "year_receipts" : 1990,
        "year_release" : 2001
    }
]

} `;

// вывод строки в браузер

document.write(personJSON);

</script>

```

Запись в формате JSON на нескольких строках делает его более **читабельным**, особенно при работе с большим количеством данных. JSON игнорирует пробелы между его элементами, поэтому вы можете свободно распределять пары ключ-значение, чтобы сделать данные **проще для восприятия**.

Сравните, к примеру, запись объекта JSON из example_5 в одну строку:

```

{ "surname" : "Бородина", "name" : "Ксения", "patronymic" :
"Алексеевна", "post" : "Преподаватель", "category" :
"Соответствие занимаемой должности", "experience" : {"total" :
21.3, "college" : 14.5}, "courses" : ["Профессия веб-
разработчик", "Fullstack разработчик на Python"], "education" :
[ { "institution" : "Санкт-Петербургский государственный

```

```
университет (СПбГУ)", "qualification" : "Информационные системы  
и технологии", "specialty" : "Безопасность киберфизических  
систем", "year_receipts" : 1993, "year_release" : 1998 }, {  
"institution" : "Московский государственный университет имени  
М.В.Ломоносова", "qualification" : "Факультет вычислительной  
математики и кибернетики", "specialty" : "Прикладная математика  
и информатика", "year_receipts" : 1990, "year_release" : 2001 }  
] }
```

Вот **типичные ошибки** при составлении строк формата JSON:

```
let json = `{  
    name : "John",    // имя свойства без кавычек  
    "surname" : 'Smith', // одинарные кавычки в значении  
    'isAdmin' : false, // одинарные кавычки в ключе  
    "birthday" : new Date(2022, 08, 16) // конструктор "new" не  
    допускается, только значения  
}`;
```

Объект JSON

Для работы с форматом JSON в JavaScript есть **глобальный объект JSON**. У объекта JSON есть два метода: **stringify()** и **parse()**. Кроме этих двух методов он не содержит больше никакой дополнительной функциональности:

- **JSON.stringify** для преобразования JS-объектов в JSON.
- **JSON.parse** для преобразования JSON обратно в JS-объект.

Метод JSON.stringify()

Функция **JSON.stringify()** преобразует объект JS в строку JSON.

```
let json = JSON.stringify(value, [replacer, space]);
```

Параметры

- **value** – значение для кодирования.
- **replacer** – массив свойств для кодирования или функция соответствия, которая будет вызываться для каждой пары key, value.
- **space** – дополнительное пространство (отступы), используемое для форматирования.

Возвращаемое значение

JSON-строка.

В большинстве случаев **JSON.stringify** используется только с **первым** аргументом. Но если нам нужно настроить процесс преобразования, то можно использовать второй аргумент JSON.stringify (сделаем в теме, посвященной функциям).

example_6. Преобразование объекта JS в строку JSON

```
<script>
```

```
// -- пользователь выбрал товар на сайте одежды
// заказ сформирован в виде JS-объекта
// в объекте допускаются строки в одинарных кавычках

let order = {
  id : 1,
  title : 'Утепленная стеганая куртка',
  brand : 'MARCO DI RADII',
  colour : 'Синий',
  country : 'Китай',
  hood : 'Есть',
  size : 52,
  price : 25.590,
  image : 'steganaya-marco.jpg'
```

```
};

// преобразуем объект в JSON

let orderJSON = JSON.stringify(order);

// вывод в консоль объекта заказа

console.log(order);

// вывод в консоль заказа в виде строки JSON

console.log(orderJSON);

// заказ упакован в JSON, можно отправлять для обработки на сервер

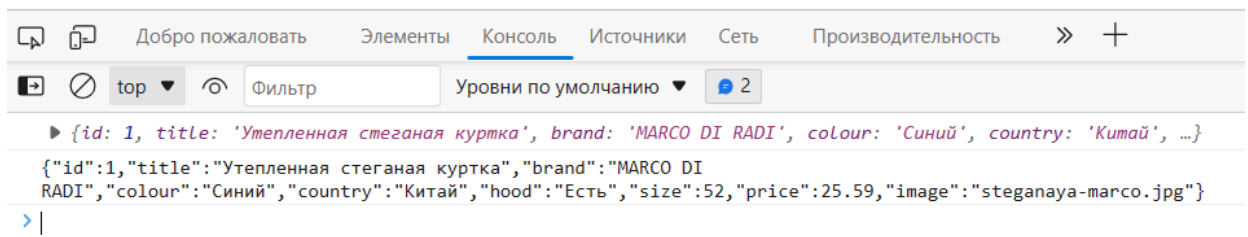
// дальнейшие действия зависят от логики работы приложения

</script>
```

Полученная строка JSON называется **JSON-форматированным** или **сериализованным объектом**. Строки имеют малый вес и поэтому очень полезны при обмене данными между клиентом и сервером или для размещения в хранилище данных.

Обратите внимание, что объект в формате JSON имеет несколько важных отличий от объектного литерала JS:

- Строки используют **двойные кавычки**. Никаких одинарных кавычек или обратных кавычек в JSON. Так 'Утепленная стеганая куртка' становится "Утепленная стеганая куртка".
- Имена свойств объекта также заключаются в **двойные кавычки**. Это обязательно. Так size:52 становится "size":52.
- Конечные **запятые запрещены**.
- **Ведущие нули запрещены**, перед десятичной запятой обязательно должна быть хотя бы одна цифра.



JSON.stringify поддерживает **все типы данных JSON**.

example_7. Преобразование разных типов данных

```
<script>

    // число в JSON остаётся числом

    console.log (JSON.stringify(25)); // 25

    // строка в JSON по-прежнему остаётся строкой, но в двойных
    кавычках

    console.log (JSON.stringify('test')) // "test"

    console.log (JSON.stringify(true)); // true

    console.log (JSON.stringify(null)); // null

    console.log (JSON.stringify([1, 2, 3])); // [1,2,3]

    // {"a":1,"b":"2","c":"три"}

    console.log (JSON.stringify({a:1, b:'2', c:'три'}));

</script>
```

JSON является независимой от языка спецификацией для данных, поэтому JSON.stringify не обрабатывает некоторые специфические свойства объектов JavaScript.

А именно:

- свойства-функции (**методы**)
- свойства, содержащие **undefined**
- **символьные** ключи и значения

example_8. Сериализуются не все поля объекта

```
<script>

    // определение JS-объекта

    let person = {

        surname : "Бородина",

        name : "Ксения",

        post : "Преподаватель",

        note : undefined,

        fnMsg () {return "Добрый день, " + this.name}

    };

    // сериализуем объект

    console.log(JSON.stringify(person));

</script>
```

Протестируем второй аргумент метода **stringify**. Второй аргумент **либо массив** свойств для кодирования, **либо функция** соответствия.

В примере example_9 передадим массив свойств для кодирования, определяющий **список ключей объекта**, которые будут сериализованы в JSON формат.

example_9. Сериализация с дополнительным параметром

```
<script>

// определение JS-объекта

let person = {

    surname : "Бородина",

    name : "Ксения",

    patronymic : "Алексеевна",

    category : "Соответствие занимаемой должности",


```

```
        post : "Преподаватель",
        level_edu : "Высшее профессиональное",
        experience_total : 21.3,
        experience_college : 14.5,
        note : "",
    };

    // сериализуем объект
    // массив определяет сериализуемые ключи объекта
    console.log(JSON.stringify(person, ["surname", "name",
    "category", "experience_college"]));
</script>
```

Метод JSON.parse()

Чтобы декодировать JSON-строку нужен метод **JSON.parse()**. Метод преобразует строку JSON в соответствующее значение JavaScript.

Особенности преобразования:

Если во время преобразования в строке JSON встретится значение **undefined**, то оно будет опущено (не будет включено в результат).

Примечание. Может возникнуть вопрос, если метод **JSON.stringify** игнорирует свойства со значением **undefined**, то откуда оно появилось в строке JSON. Все просто – строка JSON может быть собрана **вручную**.

```
let value = JSON.parse(str, [reviver]);
```

Параметры

- **str** - JSON для преобразования в объект.

- **reviver** - необязательная функция, которая будет вызываться для каждой пары (ключ, значение) и может преобразовывать значение.

example_10. Десериализация строки в объект

```
<script>
```

```
// -- пользователь запросил информацию о товаре на сайте
одежды

// информация о товаре поступила в виде строки JSON

let orderJSON = '{ "id" : 1, "title" : "Утепленная стеганая
куртка", "brand" : "MARCO DI RADII", "colour" : "Синий",
"country" : "Китай", "hood" : "Есть", "size" : "52", "price"
: 25.590, "image" : "steganaya-marco.jpg" }';

// вывод в консоль

console.log(orderJSON);

// для доступа к данным преобразуем JSON в JS-объект

let order = JSON.parse(orderJSON);

// вывод в консоль звказа в виде JS-объекта

console.log(order);

// тестируем доступ

console.log(order.title);
```

```
</script>
```

Вложенность в строках JSON

Методы объекта JSON позволяют конвертировать сложные структуры. JSON может быть настолько сложным, насколько это необходимо, **объекты и массивы могут включать другие объекты и массивы**. Но они должны быть в том же JSON-формате.

Важно. Вложенные сложные типы поддерживаются и конвертируются автоматически.

Если пользователь выбрал какой-либо товар на сайте, то информацию о заказе удобно хранить в виде **объекта**. Но если пользователь выбрал не один товар, а больше, то информация может храниться в виде **массива из двух или более объектов**.

Чем активнее пользователь, тем сложнее может быть структура заказа. Но в любом случае, внешним контейнером заказа будет либо объект, либо массив, включающие в себя другие массивы и объекты, описывающие однотипные товары.

example_11. Сериализация массива в JSON

```
<script>
```

```
// -- пользователь выбрал два товара на сайте одежды
// заказ сформирован в виде JS-массива

let order = [
  {
    id : 1,
    title : "Утепленная стеганая куртка",
    brand : "MARCO DI RADII",
    colour : "Синий",
    country : "Китай",
    hood : "Есть",
    size : 52,
    price : 25.590,
    image : "steganaya-marco.jpg"
  },
```

```
{  
  
    id : 2,  
    title : "Пальто",  
    brand : "Allsaints",  
    colour : "Коричневый",  
    country : "Китай",  
    hood : "Отсутствует",  
    size : 50,  
    price : 53.990,  
    image : "palto-allsaints.jpg"  
  
}  
  
];  
  
// преобразуем JS-массив в JSON  
  
let orderJSON = JSON.stringify(order);  
  
// вывод в консоль массива заказа  
  
console.log(order);  
  
// вывод в консоль заказа в виде JSON  
  
console.log(orderJSON);  
  
// заказ упакован в JSON, можно отправлять для обработки на сервер  
  
// дальнейшие действия зависят от логики работы приложения  
  
</script>
```

example_12. Парсинг строки JSON в массив JS

```
<script>
```

```
// -- пользователь запросил информацию о двух товарах на
сайте одежды
```

```
// информация о товаре поступила в виде JSON-массива
```

```
let orderJSON = `[
```

```
{
```

```
  "id" : 1,
```

```
  "title" : "Утепленная стеганая куртка",
```

```
  "brand" : "MARCO DI RADII",
```

```
  "colour" : "Синий",
```

```
  "country" : "Китай",
```

```
  "hood" : "Есть",
```

```
  "size" : 52,
```

```
  "price" : 25.590,
```

```
  "image" : "steganaya-marco.jpg"
```

```
},
```

```
{
```

```
  "id" : 2,
```

```
  "title" : "Пальто",
```

```
  "brand" : "Allsaints",
```

```
  "colour" : "Коричневый",
```

```
  "country" : "Китай",
```

```
  "hood" : "Отсутствует",
```

```
  "size" : 50,
```

```
  "price" : 53.990,
```

```
        "image" : "palto-allsaints.jpg"

    }

]`;

// вывод в консоль

console.log(orderJSON);

// для доступа к данным преобразуем JSON в JS-массив

let order = JSON.parse(orderJSON);

// вывод в консоль заказа в виде JS-массива

console.log(order);

// тестируем доступ

console.log(order[0].title);

console.log(order[1].title);

</script>
```

Так, если на сайте возможно заказать телевизор, свитер худи и спортивную добавку (почему бы и нет ;), то формат JSON может состоять из объекта, содержащего три ключа (например: tv, hoody, sportpit), значением которых будут массивы, описывающие однотипные товары. И даже если каждый из товаров заказан в единственном экземпляре, хранить его удобно в массиве.

Формированием такого заказа и займемся в одной из самостоятельных работ.

Обработка вложенных структур JSON

JSON может быть достаточно сложной структурой. Создавая структуру для хранения чего бы то ни было, вы должны представлять, насколько удобно эту структуру будет разбирать для обработки и вывода в браузер. Выполняются такие действия, как правило, с помощью управляющих конструкций языка JavaScript.

К основным операторам и управляющим конструкциям JS относятся:

- **условный** оператор
- **тернарный** оператор
- оператор **switch**
- операторы **цикла**

Важно. Смысл хранить сложные структуры есть только тогда, когда есть возможность обрабатывать эти структуры, например с помощью **условных конструкций** и **циклов**.

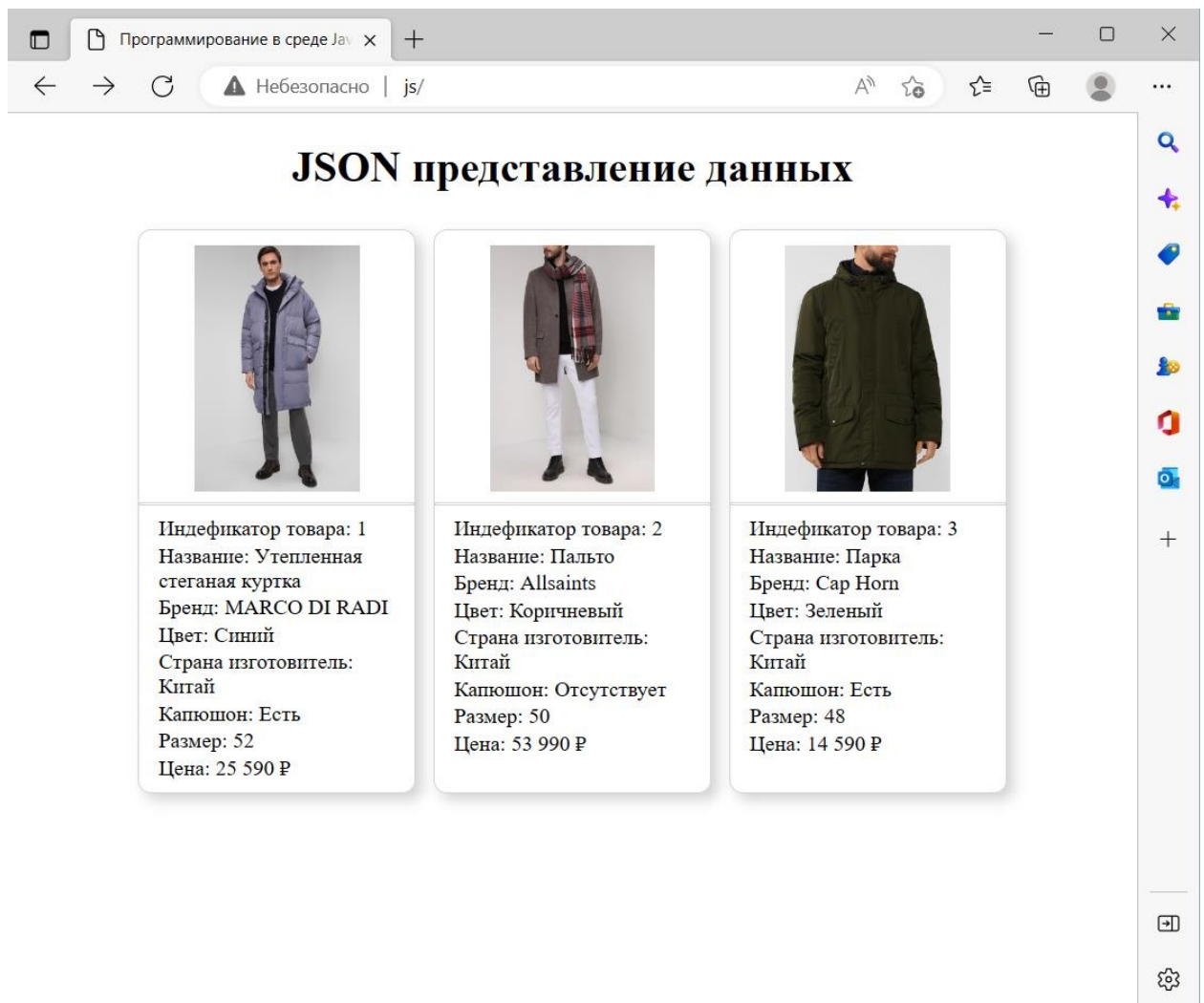
Обработкой сложных структур данных займемся в теме: **Управляющие конструкции**.

Задание 23

Пользователь запросил на сайте одежды некоторую информацию. Ответ пришел в виде **JSON-строки**.

Используйте материал раздаточной директории. Подключите файл **product.js**, имитирующий ответ сервера, используя метод **JSON.parse** распарсите строку в **JS-объект**. Выведите информацию о товарах в браузер.

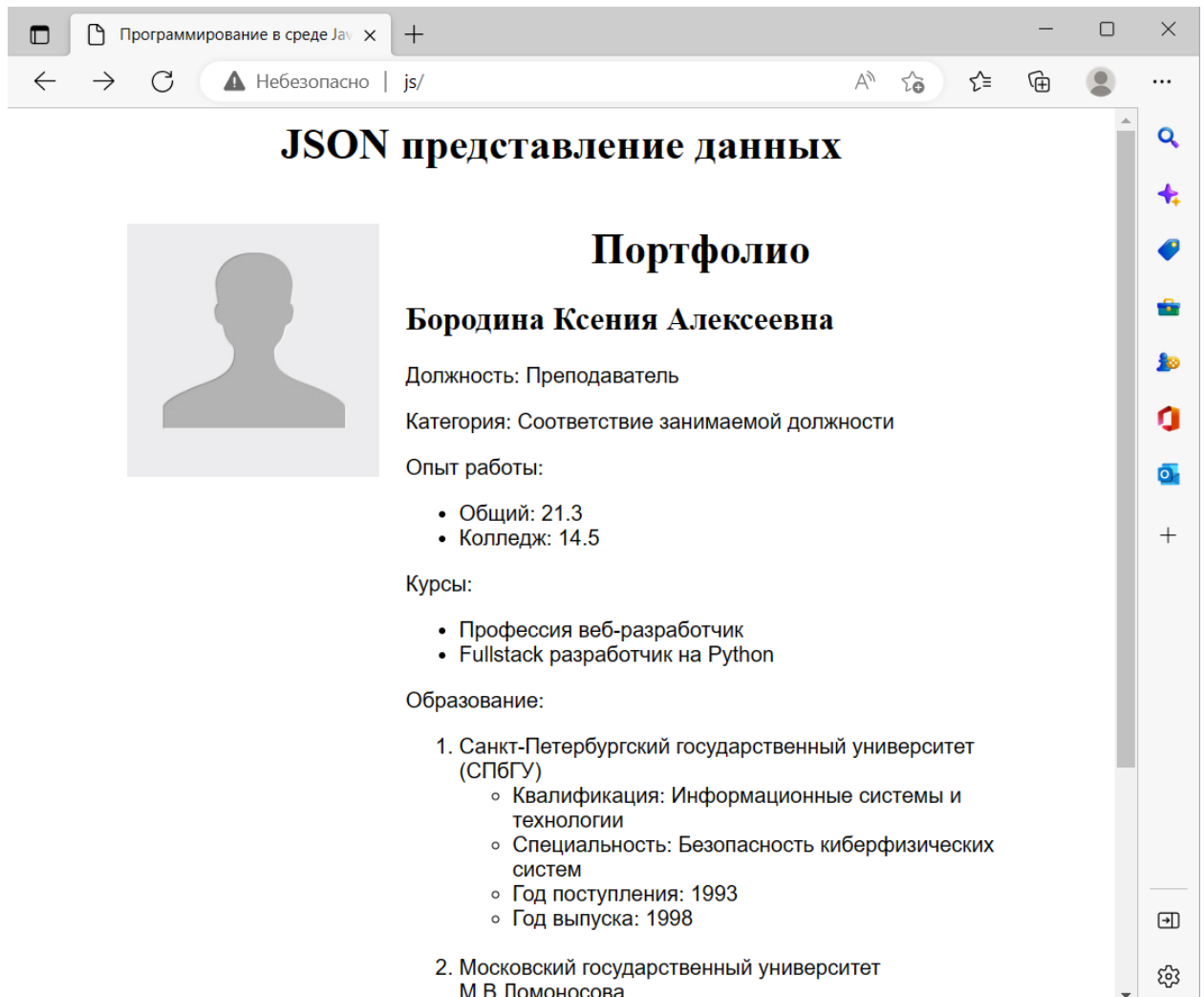
Примерный формат вывода представлен на рисунке.



Задание 24

В файле **index.html** раздаточного материала вам предложена строка формата **JSON**. **Вручную** преобразуйте **JSON** строку в **JavaScript** объект. Выведите данные объекта в браузер.

Примерный формат вывода представлен на рисунке.



Задание 25

В файле **car.js** раздаточного материала вам представлены массивы.

Создайте строку **JSON формата**. Строка должна содержать **объект с тремя ключами** (свойствами):

- **toyota**
- **mitsubishi**
- **bmw**

Значениями каждого ключа являются **массивы**, содержащие **объекты**, описывающие представленные автомобили.


Используя метод **JSON.parse** преобразуйте строку в **JS-объект**. Если строка JSON собрана правильно, **преобразование в объект** пройдет без ошибок.

Выведите автомобили марки **Toyota** в браузер. **Примерный формат** вывода представлен на рисунке.


Программирование в среде Java x +

← → ↻ ⚠ Небезопасно | js/ 🔍 ⭐ ⚙ 👤 ...

JSON представление данных



ID	1
Модель	Toyota Land Cruiser
Тип ДВС	Дизельный
Мощность л.с.	249
Расположение цилиндров	V-образный
Рабочий объем	4461
Число цилиндров	8



ID	2
Модель	Toyota Land Cruiser Prado
Тип ДВС	Бензиновый
Мощность л.с.	163
Расположение цилиндров	Рядный
Рабочий объем	2694
Число цилиндров	4

Задание 26

Некий пользователь (пусть это будет условная **Татьяна**) решил совершить **Новогодний online-шопинг**. В результате корзина личного кабинета пользователя пополнилась следующими товарами:

1. Телевизор **Samsung**

- тип: LED
- диагональ: 55'
- разрешение: 4K UHD
- цена: 56 940

2. Блуза **Хэйзи**

- артикул: 24028BL
- цвет: белый
- ткань: виск
- размер: M
- цена: 3 200

3. **Martini Rosso**

- объем: 0.5
- алкоголь: 13%
- вкус: Prosecco Rose
- цена: 1620

Для отправки **online заказа** на сервер сформируйте строку **JSON**. Для проверки правильности составления строки используйте метод **JSON.parse**, преобразуйте **строку формата JSON** в объект **JS**.

Выведите **объект JS** в отладочную **консоль**.