Формат JSON, метод toJSON

JSON (JavaScript Object Notation) – это общий формат для представления значений и объектов. Его описание задокументировано в стандарте RFC 4627. Первоначально он был создан для JavaScript, но многие другие языки также имеют библиотеки, которые могут работать с ним. Таким образом, JSON легко использовать для обмена данными, когда клиент использует JavaScript, а сервер написан на Ruby/PHP/Java или любом другом языке.

JavaScript предоставляет **методы**:

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

JSON поддерживает следующие типы данных:

- Объекты { ... }
- Массивы [...]
- Примитивы:
- строки,
- числа,
- логические значения true/false,
- null.

Строки используют **двойные кавычки**. Никаких одинарных кавычек или обратных кавычек в JSON. **Строковые ключи** объекта также заключаются в **двойные кавычки**. В отличие от структур JavaScript, в JSON не разрешены запятые после последних элементов массивов и объектов

Например:

```
// число в JSON остаётся числом
alert( JSON.stringify(1) ) // 1

// строка в JSON по-прежнему остаётся строкой, но в двойных кавычках
alert( JSON.stringify('test') ) // "test"

alert( JSON.stringify(true) ); // true

alert( JSON.stringify([1, 2, 3]) ); // [1,2,3]
```

JSON является независимой от языка спецификацией для данных, поэтому JSON.stringify **пропускает** некоторые специфические свойства объектов JavaScript. А именно:

- Свойства-функции (методы).
- Символьные свойства.
- Свойства, содержащие undefined.

```
let user = {
  sayHi() { // будет пропущено
    alert("Hello");
  },
  [Symbol("id")]: 123, // также будет пропущено
  something: undefined // как и это - пропущено
};
alert( JSON.stringify(user) ); // {} (пустой объект)
```

Вложенные объекты поддерживаются и конвертируются автоматически. Важное ограничение: не должно быть **циклических ссылок**. JSON не поддерживает **комментарии**.

```
let room = {
  number: 23
};
let meetup = {
  title: "Conference",
  participants: ["john", "ann"]
};
meetup.place = room;       // meetup ссылается на room
room.occupiedBy = meetup; // room ссылается на meetup
JSON.stringify(meetup); // Ошибка: Преобразование цикличной структуры в JSON
```

Полный синтаксис **JSON.stringify**:

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

value - Значение для кодирования.
replacer - Массив свойств для кодирования или функция соответствия function(key, value).
space - Дополнительное пространство (отступы), используемое для форматирования.
Полученная строка json называется JSON-форматированным или сериализованным объектом.

Пример1:

```
let room = {
  number: 23
};
let meetup = {
  title: "Conference",
  participants: [{name: "John"}, {name: "Alice"}],
  place: room // meetup ссылается на room
};

room.occupiedBy = meetup; // room ссылается на meetup

alert( JSON.stringify(meetup, ['title', 'participants', 'place', 'name', 'number']) );
/*{
  "title":"Conference",
  "participants":[{"name":"John"},{"name":"Alice"}],
  "place":{"number":23}
}*/
```

Пример2:

```
let room = {
  number: 23
};
let meetup = {
  title: "Conference",
  participants: [{name: "John"}, {name: "Alice"}],
  place: room // meetup ссылается на room
};

room.occupiedBy = meetup; // room ссылается на meetup

alert( JSON.stringify(meetup, function replacer(key, value) {
  alert(`${key}: ${value}`);
  return (key == 'occupiedBy') ? undefined : value;
}));

/* пары ключ:значение, которые приходят в replacer:
:            [object Object]
title:       Conference
participants: [object Object],[object Object]
0:           [object Object]
name:        John
1:           [object Object]
name:        Alice
place:       [object Object]
number:      23    */
```

Доп. информацию см https:/learn.javascript.ru/json

Как и toString для преобразования строк, объект может предоставлять метод toJSON (пользовательский) для преобразования в JSON. JSON.stringify автоматически вызывает его, если он есть. Например:

```
let room = {
  number: 23,
  toJSON() {
    return this.number;
  }
};
let meetup = {
  title: "Conference",
  room
```

```
};
alert( JSON.stringify(room) ); // 23
alert( JSON.stringify(meetup) );
/*  {
   "title":"Conference",
   "room": 23
 }  */
```

JSON.parse - чтобы декодировать JSON-строку (десериализация). Синтаксис:

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

str -JSON для преобразования в объект.
reviver - Необязательная функция, которая будет вызываться для каждой пары (ключ, значение) и может преобразовывать значение.
Например:

```
let user = '{ "name": "John", "age": 35, "isAdmin": false, "friends": [0,1,2,3] }';
user = JSON.parse(user);
alert( user.friends[1] ); // 1
```

Еще пример:

```
let str = '{"title":"Conference","date":"2017-11-30T12:00:00.000Z"}';

let meetup = JSON.parse(str, function(key, value) {
  if (key == 'date') return new Date(value);
  return value;
});

alert( meetup.date.getDate() ); // 30 - теперь работает!
```