РАБОТА С JSON ОБЪЕКТЫ МАССИВЫ

JavaScript

Модуль 1. Урок 2.

JSON

Что такое JSON?
JSON английское название **JavaScript Object Notation**
JSON представляет собой легкий формат обмена данными.
JSON является независимым от языка

JSON представляет собой строку. В этой строке данные хранятся в формате, похожим на обычные объекты JavaScript. Но есть отличия: все строки (как ключи, так и значения) обязательно должны быть в двойных кавычках.

```
var json = {
 "name": "Иван",
 "age": 20,
 "online": true,
 "wife": null,
 "arr": [1, 2, 3, 4]
};
```

МЕТОДЫ JSON

Метод **JSON.parse** преобразует строку с JSON в обычный объект JavaScipt, с которым можно будет работать дальше обычными средствами

```
var json = '{ "name": "Иван", "age": 20, "online": true, "wife":
null, "arr": [1, 2, 3, 4], }’;

var result = JSON.parse(json);
console.log(result); //увидим объект
```

МЕТОДЫ JSON

Метод JSON.stringify преобразует объекты JavaScript в строку JSON. Посмотрим на следующих примерах:

```
var obj = {
  name: 'Иван’,
  age: 20,
  bool: true,
  arr: [1, 2, 3, 4]
};
var json = JSON.stringify(obj);
console.log(json);
```

Результат:

```
'{ "name": "Иван", "age": 20, "empty": null, "arr": [1, 2, 3, 4], }';
```

ЛИТЕРАЛЫ ТИПА OBJECT

```
var empty = {};                                 // An object with no properties
var point = { x:0, y:0 };                       // Two properties
var point2 = { x:point.x, y:point.y+1 };        // More complex values
var book = {
    "main title": "JavaScript",                 // Property names include spaces,
    'sub-title': "The Definitive Guide",        // and hyphens, so use string
                                                // literals
    "for": "all audiences",                     // for is a reserved word, so quote
    author: {                                   // The value of this property is
        firstname: "David",                     // itself an object.  Note that
        surname: "Flanagan“                    // these property names are
                                                // unquoted.
    }
};
```

КЛАСС OBJECT

- Объект представляет собой составное значение: он объединяет несколько значений (примитивные значения или другие объекты) и позволяет сохранять и извлекать эти значения по имени
- Объектом является неупорядоченный набор свойств, каждый из которых имеет имя и значение
- Имена свойств - это строки, поэтому мы можем сказать, что объекты отображают строки в значения
- Объект наследует свойства другого объекта, известного как его «прототип» (prototype)
- Объекты динамические - свойства обычно можно добавлять и удалять

СОЗДАНИЕ ОБЪЕКТОВ

Объекты могут быть созданы с помощью:

- литерал, например {x:1}
- ключевое слово new, например new Object()
- функция Object.create()

КЛАСС OBJECT

Оператор new создает и инициализирует новый объект

```
var o = new Object();      // Create an empty object: same as {}.
var a = new Array();       // Create an empty array: same as [].
var d = new Date();        // Create a Date object representing the current time
var r = new RegExp("js"); // Create a RegExp object for pattern matching.
```

OBJECT

Object.create() создает новый объект, используя первый аргумент как прототип объекта. Object.create() также принимает опциональный второй аргумент, описывающий свойства объекта.

```
var o1 = Object.create({x:1, y:2});      // o1 inherits properties x and y.
var o2 = Object.create(null);            // o2 inherits no props or methods.
var o3 = Object.create(Object.prototype); // o3 is like {} or new Object().
```

СВОЙСТВА ОБЪЕКТА

- Свойство имеет имя и значение
- Имя свойства может быть любой строкой, включая пустую строку, но ни один объект не может иметь два свойства с одним именем
- Значение может быть любым значением JavaScript
- Свойство можно определить с помощью методов getter и setter

```
var o = {
    // An ordinary data property
    data_prop: value,

    // An accessor property defined as a pair of functions
    get accessor_prop() { /* function body here */ },
    set accessor_prop(value) { /* function body here */ }
};
```

СВОЙСТВА ОБЪЕКТА

- Чтобы получить значение свойства, используйте операторы точки (.) или квадратной скобки ([])
- Новое свойство может быть добавлено к объекту
- Свойство может быть удалено из объекта с помощью оператора delete

```
var author = book.author;         // Get the "author" property of the book.
var name = author.surname         // Get the "surname" property of the author.
var title = book["main title"] // Get the "main title" property of the book.
delete book.author;               // The book object now has no author property.
delete book["main title"];        // Now it doesn't have "main title", either.
```

СВОЙСТВА ОБЪЕКТА

- Оператор in используется для проверки того, имеет ли объект свойство с заданным именем
- Метод hasOwnProperty() объекта проверяет, имеет ли этот объект собственное свойство с заданным именем. Он возвращает false для унаследованных свойств

```
var o = { x: 1 }
"x" in o;           // true: o has an own property "x"
"y" in o;           // false: o doesn't have a property "y"
"toString" in o;    // true: o inherits a toString property

o.hasOwnProperty("x");         // true: o has an own property x
o.hasOwnProperty("y");         // false: o doesn't have a property y
o.hasOwnProperty("toString"); // false: toString is an inherited property
```

ОБЪЕКТ КАК МАССИВ

- Цикл for / in используется для перечислимого свойства (собственного или унаследованного) указанного объекта
- Встроенные методы, которые наследуют объекты, не перечислимы
- Свойство propertyIsEnumerable() возвращает true, только если именованное свойство является собственным свойством и перечислимо.

```
var o = {x:1, y:2, z:3};                // Three enumerable own properties
o.propertyIsEnumerable("toString")      // => false: not enumerable
for(p in o)                             // Loop through the properties
    console.log(p);                     // Prints x, y, and z, but not toString

for(p in o)
    if (!o.hasOwnProperty(p)) continue;       // Skip inherited properties

for(p in o)
    if (typeof o[p] === "function") continue; // Skip methods
```

СТАНДАРТНЫЕ СВОЙСТВА И МЕТОДЫ ОБЪЕКТА

- Все объекты JavaScript (кроме явно созданных без прототипа) наследуют свойства от Object.prototype
- Object.prototype определяет некоторые полезные методы: toString (), valueOf (), toJSON ()

СТАНДАРТНЫЕ СВОЙСТВА И МЕТОДЫ ОБЪЕКТА

- Метод toString() не принимает аргументов; он возвращает строку, которая каким-то образом представляет значение объекта, на который он вызывается
- JavaScript вызывает этот метод объекта всякий раз, когда ему нужно преобразовать объект в строку
- Например, когда вы используете оператор + для конкатенации строки с объектом или когда вы передаете объект методу, который ожидает строку
- По умолчанию метод toString () оценивается по строке «[object Object]»

```
var s = { x:1, y:1 }.toString(); // "[object Object]"
```

СТАНДАРТНЫЕ СВОЙСТВА И МЕТОДЫ ОБЪЕКТА

- Метод valueOf() вызывается, когда JavaScript необходимо преобразовать объект в некоторый примитивный тип, отличный от строки - обычно число
- JavaScript вызывает этот метод автоматически, если объект используется в контексте, где требуется примитивное значение
- Некоторые из встроенных классов определяют свой собственный метод valueOf(). Например, Date.valueOf() возвращает число миллисекунд с 00:00 01 января 1970 UTC

СТАНДАРТНЫЕ СВОЙСТВА И МЕТОДЫ ОБЪЕКТА

- Object.prototype фактически не определяет метод toJSON()
- Метод JSON.stringify () ищет метод toJSON() для любого объекта, которому предлагается сериализовать
- Если этот метод существует для объекта, который будет сериализован, он вызывается, а возвращаемое значение сериализуется вместо исходного объекта

МАССИВ

- Самый простой способ создания массива - иметь литерал массива.

```
var empty = [];                  // An array with no elements
var primes = [2, 3, 5, 7, 11];   // An array with 5 numeric elements
var misc = [ 1.1, true, "a", ]; // 3 elements of various types + trailing comma

var base = 1024;
var table = [base, base+1, base+2, base+3]; // values may be expressions

var b = [[1,{x:1, y:2}], [2, {x:3, y:4}]];  // can contain object literals
                                            // or other array literals


var a = new Array();   // no arguments - empty array same as []
var a = new Array(10); // a single numeric argument - specifies a length

// the constructor arguments become the elements of the new array
var a = new Array(5, 4, 3, 2, 1, "testing, testing");
```

- Другим способом создания массива является конструктор Array()

МАССИВ

- Оператор [] используется для доступа к элементу массива
- Произвольное выражение, которое имеет неотрицательное целочисленное значение, должно быть внутри скобок
- Этот синтаксис используется как для чтения, так и для записи значения элемента массива
- Массив автоматически сохраняет значение свойства length

```
var a = ["world"];      // Start with a one-element array
var value = a[0];       // Read element 0
a[1] = 3.14;            // Write element 1
i = 2;
a[i] = 3;               // Write element 2
a[i + 1] = "hello";     // Write element 3
a[a[i]] = a[0];         // Read elements 0 and 2, write element 3
```

МАССИВ

- Массив - это упорядоченный набор значений
- Каждое значение называется элементом, и каждый элемент имеет числовое положение в массиве, называемое его индексом
- Массивы JavaScript являются нетипизированными: элемент массива может быть любого типа, а разные элементы одного и того же массива могут быть разных типов
- Массивы JavaScript динамичны: они растут или сокращаются по мере необходимости
- Массивы наследуют свойства от Array.prototype, который определяет богатый набор методов манипуляции массивами

МАССИВ

- Массивы - это специализированный вид объекта
- Квадратные скобки, используемые для доступа к элементам массива, работают так же, как квадратные скобки, используемые для доступа к свойствам объекта
- JavaScript преобразует индекс числового массива, который вы указываете, в строку - индекс 1 становится строкой «1» - затем использует эту строку в качестве имени свойства

```
o = {};          // Create a plain object
o[1] = "one";    // Index it with an integer
o[“two”] = ”two";  // Index it with a string
```

МАССИВ

- Все индексы являются именами свойств, но только имена свойств, которые являются целыми числами от 0 до 232-2, являются индексами
- Отрицательные или не целочисленные индексы преобразуются в строку, и эта строка используется как имя свойства
- Когда строка, которая является неотрицательным целым числом или числом с плавающей запятой, используется в качестве индекса, она ведет себя как индекс массива, а не свойство объекта

```
a[-1.23] = true;  // This creates a property named "-1.23"
a["1000"] = 0;    // This the 1001st element of the array
a[1.000]          // Array index 1.  Same as a[1]
```

ДОБАВЛЕНИЕ ЭЛЕМЕНТОВ

- Чтобы добавить новый элемент в массив, просто присвойте ему значение

```
a = new Array(5);    // No elements, but a.length is 5.
a = [];              // Create an array with no elements and length = 0.
a[1000] = 0;         // Assignment adds one element but sets length to 1001.
```

ДЛИНА МАССИВА

- Свойство length указывает количество элементов в массиве. Его значение больше, чем самый высокий индекс в массиве
- Когда массив разрежен, свойство length больше, чем количество элементов, и все, что мы можем сказать об этом, состоит в том, что длина гарантируется быть больше, чем индекс каждого элемента массива
- Если для свойства length задано неотрицательное целое число n, меньшее его текущего значения, любые элементы массива, индекс которых больше или равен n, удаляются из массива

```
[].length               // => 0: the array has no elements
['a','b','c'].length    // => 3: highest index is 2, length is 3

a = [1,2,3,4,5];        // Start with a 5-element array.
a.length = 3;           // a is now [1,2,3].
a.length = 0;           // Delete all elements.  a is [].
a.length = 5;           // Length is 5, but no elements, like new Array(5)
```

МЕТОДЫ МАССИВА

- Метод **Array.join()** преобразует все элементы массива в строки и объединяет их, возвращая полученную строку
- Сепаратор элементов можно указать как необязательный параметр строки. По умолчанию элементы в полученной строке разделяются запятой

```
var a = [1, 2, 3];      // Create a new array with these three elements
a.join();               // => "1,2,3"
a.join(" ");            // => "1 2 3"
a.join("");             // => "123"
var b = new Array(10); // An array of length 10 with no elements
b.join('-')             // => '---------': a string of 9 hyphens
```

МЕТОДЫ МАССИВА

- Метод Array.reverse () меняет порядок элементов массива и возвращает массив в обратном порядке
- reverse() не создает новый массив, а вместо этого перестраивает их в уже существующем массиве

```
var a = [1,2,3];
a.reverse().join()  // => "3,2,1" and a is now [3,2,1]
```

МЕТОДЫ МАССИВА

- Array.sort() сортирует элементы массива на месте и возвращает отсортированный массив
- Когда sort() вызывается без аргументов, он сортирует элементы массива в алфавитном порядке
- Чтобы отсортировать массив в некотором порядке, отличном от алфавитного, функция сравнения должна быть передана как аргумент sort ()

```
var a = [33, 4, 1111, 222];
a.sort();                    // Alphabetical order:  1111, 222, 33, 4
a.sort(function(a,b) {       // Numerical order: 4, 33, 222, 1111
    return a-b;     // Returns < 0, 0, or > 0, depending on order
});
a.sort(function(a,b) {return b-a});    // Reverse numerical order
```

МЕТОДЫ МАССИВА

- Метод Array.concat () создает и возвращает новый массив, содержащий элементы исходного массива, на который был вызван concat (), за которым следуют каждый из аргументов concat ()
- concat () не изменяет массив, на который он вызывается

```
var a = [1,2,3];
a.concat(4, 5)          // Returns [1,2,3,4,5]
a.concat([4,5]);        // Returns [1,2,3,4,5]
a.concat([4,5],[6,7])   // Returns [1,2,3,4,5,6,7]
a.concat(4, [5,[6,7]])  // Returns [1,2,3,4,5,[6,7]]
```

МЕТОДЫ МАССИВА

```
var a = [1,2,3,4,5];
a.slice(0,3);    // Returns [1,2,3]
a.slice(3);      // Returns [4,5]
a.slice(1,-1);   // Returns [2,3,4]
a.slice(-3,-2);  // Returns [3]
```

array.slice(start,end)

start - *required*. Целое число, указывающее, где начать выбор. Используйте отрицательные числа для выбора из конца массива

end - *optional*. Целое число, определяющее, где завершить выбор.

МЕТОДЫ МАССИВА

array.splice(index, howmany, item1, ..., itemX)

index - Требуется. Целое число, указывающее, в какую позицию добавлять / удалять элементы. Используйте отрицательные значения для указания позиции с конца массива

howmany - обязательно. Количество элементов для удаления. Если установлено значение 0, никакие элементы не будут удалены

item1, ..., itemX - Необязательно. Новый элемент (ы), который нужно добавить в массив

```
var a = [1,2,3,4,5,6,7,8];
a.splice(4);     // Returns [5,6,7,8]; a is [1,2,3,4]
a.splice(1,2);   // Returns [2,3]; a is [1,4]
a.splice(1,1);   // Returns [4]; a is [1]

var a = [1,2,3,4,5];
a.splice(2,0,'a','b');  // Returns []; a is [1,2,'a','b',3,4,5]
a.splice(2,2,[1,2],3);  // Returns ['a','b']; a is [1,2,[1,2],3,3,4,5]
```

МЕТОДЫ МАССИВА

- Методы push() и pop() позволяют работать с массивами, как если бы они были стеками

```
var stack = [];          // stack: []
stack.push(1,2);         // stack: [1,2]      Returns 2
stack.pop();             // stack: [1]        Returns 2
stack.push(3);           // stack: [1,3]      Returns 2
stack.pop();             // stack: [1]        Returns 3
stack.push([4,5]);       // stack: [1,[4,5]]  Returns 2
stack.pop()              // stack: [1]        Returns [4,5]
stack.pop();             // stack: []         Returns 1
```

МЕТОДЫ МАССИВА

- Методы unshift () и shift() ведут себя подобно push() и pop(), за исключением того, что они вставляют и удаляют элементы из начала массива, а не из конца

```
var a = [];               // a:[]
a.unshift(1);             // a:[1]          Returns: 1
a.unshift(22);            // a:[22,1]       Returns: 2
a.shift();                // a:[1]          Returns: 22
a.unshift(3,[4,5]);       // a:[3,[4,5],1]  Returns: 3
a.shift();                // a:[[4,5],1]    Returns: 3
a.shift();                // a:[1]          Returns: [4,5]
a.shift();                // a:[]           Returns: 1
```

МЕТОДЫ МАССИВА

- toString() преобразует каждый из своих элементов в строку (при необходимости вызывает методы toString () своих элементов) и выводит список этих строк, разделенный запятыми

```
[1,2,3].toString()          // Yields '1,2,3'
["a", "b", "c"].toString()  // Yields 'a,b,c'
[1, [2,'c']].toString()     // Yields '1,2,c'
```

- toLocaleString() - это локализованная версия toString(). Он преобразует каждый элемент массива в строку, вызывая метод toLocaleString () элемента, а затем он объединяет результирующие строки, используя специфичную для локали (и реализацию) строку разделителя