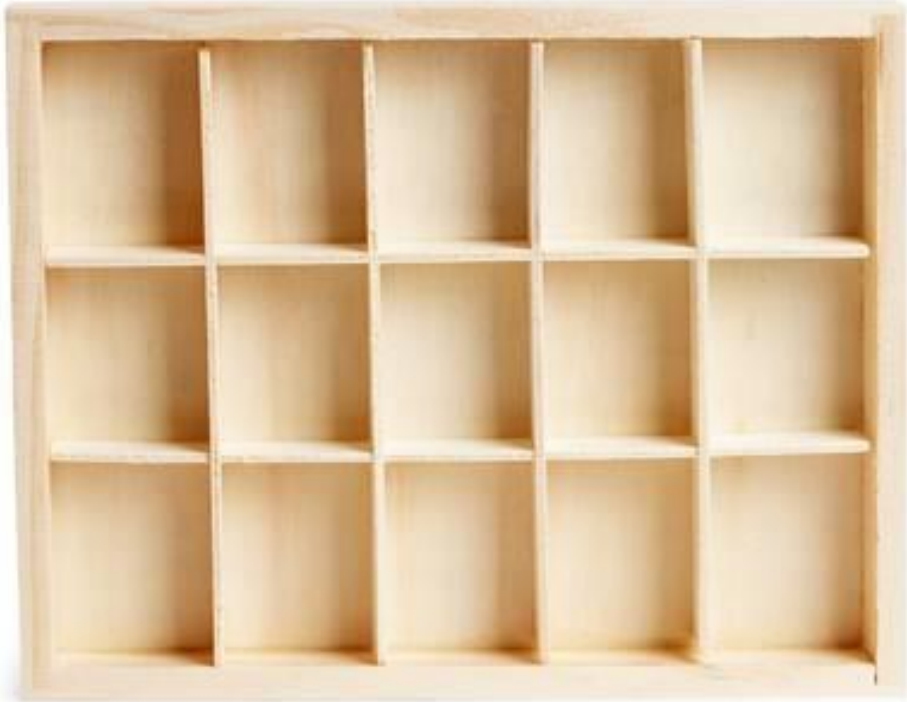


Коллекции в JavaScript

JS
COURSE
ORT DNIPRO

ORT**DNIPRO**.ORG/**JS**

Коллекции (структуры данных)



Коллекциями в языках программирования называют **структуры данных** предназначенные для хранения множества значений. Коллекции в **JavaScript** можно разделить на те которые хранят пары **ключ => значение** (массив **Array**, ассоциативные массивы **Object** и **Map**) и просто хранящие значения без индексации (множество – **Set**).

Коллекции в JavaScript

*Большинство (но не все) коллекций построено по принципу хранения пар: ключ-значение и такие коллекции называют **словари**...*

- | | |
|--|--------|
| 1. Массивы (с числовыми индексами) | Array |
| 2. Ассоциативные массивы (со строковыми индексами) | Object |
| 3. Словари (с ключом произвольного типа) | Map |
| 4. Множество (без ключей, элементы не повторяются) | Set |

*Тип данных всех коллекций – **object**, все они построение на базе объектов.*

1. Массив (**Array**)



Массив / Array

Массив (с числовыми индексами) – коллекция хранящая неограниченное количество элементов (ячеек), у каждого из которых есть порядковый номер. Типы данных хранимых в ячейках массива не ограничены, в рамках одного массива в разных ячейках могут храниться разные типы данных, в том числе и другие (вложенные) массивы.

Базовые действия с массивом

```
2
3   let arr = [10, 23, 167, 32, 77];
4   //let arr = new Array(10, 23, 167, 32, 77);
5
6   arr[2] = 787;
7
8   console.dir(arr); //All structure of object.
9
10  console.log("Array length: ", arr.length);
11  console.log("Array: ", arr[0], arr[1], arr[2], arr[3], arr[4]);
12  console.log("Out of...", arr[42]); //undefined;
13
14  arr.push(100);
15  let last = arr.pop();
16
17  arr.unshift(200);
18  let first = arr.shift();
19
20  delete arr[2]; //WARNING!
21
```

Подробнее: <https://learn.javascript.ru/array>

Полезные методы массива

```
2
3   let arr = [10, 23, 167, 32, 23, -56, 0, 77];
4
5   arr.indexOf(23);           //1;
6   arr.lastIndexOf(23);       //4;
7   arr.indexOf(99);           //-1 - It's mean NotFound;
8
9   arr.includes(32);           //true;
10  arr.includes(88);           //false;
11
12  arr.reverse();              //Previous order is lost;
13  console.log(arr);           //[77, 0, -56, 23, 32, 167, 23, 10];
14
15  let arr_2 = arr.slice(2, 5);
16  console.log("Sliced:", arr_2); //[-56, 23, 32];
17
18  arr.splice(2, 5, 'a', 'b', 'c');
19  console.log("Spliced:", arr); //[77, 0, "a", "b", "c", 10]
20
```

Подробнее: <https://learn.javascript.ru/array-methods>

Псевдомассивы на примере строкового типа

```
2
3   let str = 'Joan Peter Michelle Laura Stiven';
4
5   console.log('String length:', str.length);
6   console.log(str[0], str[1], str[2], str[3]);
7
8   let arr = str.split(' '); //Create Array;
9   console.dir(arr);    //[ "Joan", "Peter", "Michelle", "Laura", "Stiven" ];
10
11  let new_str = arr.join(', ');
12  console.log(new_str); // 'Joan, Peter, Michelle, Laura, Stiven'
13
```

Псевдомассивами называют структуры у которых есть возможность обратиться к элементами при помощи синтаксиса [...], а также возможность узнать количество элементов (**.length**), но, при этом, не являющиеся массивами и не обладающие функциональностью массивов. В частности строки не позволяют менять символы строки.

Подробнее: <https://learn.javascript.ru/string>

Оператор ...

(**spread** оператор, оператор деструктуризации)

Оператор ... (spread оператор)

```
2
3   let arr_1 = [1,2,3];
4   let arr_2 = [4,5,6, ...arr_1];
5
6   console.log(arr_1, arr_2); //[1, 2, 3] > [4, 5, 6, 1, 2, 3];
7
8   let maximun = Math.max(...arr_2);
9   console.log("Maximum", maximun); //6
10
11  let arr_copy = [...arr_2]; //One level copy of arr_2
12  console.log(arr_copy); //[4, 5, 6, 1, 2, 3];
13
```

Оператор ... (**spread** оператор) находясь по правую сторону от оператор присвоения (или при передаче параметров функции) позволяет подставить всё содержимое массива или любого другого итерируемого (перебираемого), объекта.

Подробнее: <https://learn.javascript.ru/string>

Деструктуризация массива

```
2
3   let arr = ['Alfa', 'Beta', 'Gamma', 'Delta', 'Epsilon'];
4
5   let [a, b] = arr;
6
7   console.log(a, b); //Alfa Beta;
8
9   let [c, d, ...e] = arr;
10
11  console.log(c, d, e); //Alfa Beta ["Gamma", "Delta", "Epsilon"];
12
```

Деструктуризация массива – способ извлечь элементы массива для присваивания их значений отдельным переменным.

Подробнее: <https://learn.javascript.ru/destructuring-assignment>

Немного практики #1

Задача: написать скрипт
определяющая по номеру билета
его «счастливость», т.е. если
сумма первых 3 десятичных
цифр равна **сумме** 3 последних
десятичных цифр.



2. Ассоциативный массив (Object)

Базовые действия с объектом (ассоциативным массивом)

```
2
3  ✓ let parcel = {
4      title: "Gift",
5      width: 200,
6      height: 300,
7      length: 100,
8      price: 199
9  }
10
11  parcel.price          = 119;
12  parcel.fragile        = true;
13  parcel['city code']   = '49000';
14
15  console.dir(parcel);
16
17  let {title, price, ...others} = parcel;
18
19  console.log(title, price); //Gift 199
20  console.log(others); /* { width: 200, height: 300,
21                        length: 100, fragile: true,
22                        city code: "49000" } */
23
```

Ассоциативный массив это также коллекция вида ключ-значение, но в отличие от массивов **ключом выступает** не число, а **строка**. В JavaScript в качестве ассоциативных массивов выступают **объекты (object** - одноимённый тип данных). Можно сказать также, что объекты в JavaScript построены на базе концепции ассоциативных массивов. Объекты также могут быть подвержены **деструктуризации**. Понятие длины (**length**) и последовательности элементов в ассоциативных массивах не применяется.

Подробнее: <https://learn.javascript.ru/object>

Подробнее: <https://learn.javascript.ru/destructuring-assignment>

Объект - ссылочная структура

```
2
3 let person_1 = { name: 'Jhon', age: 35 };
4 let person_2 = person_1;
5
6 person_1.name = "Helen";
7 person_1.age = 27;
8
9 console.log(person_1); // {name: "Helen", age: 27} ?!?!?!
10 console.log(person_2); // {name: "Helen", age: 27}
11
12 //let person_3 = Object.assign({}, person_1);
13 let person_3 = {...person_1};
14
15 person_1.name = "Bill";
16 person_1.age = 51;
17
18 console.log(person_1); // {name: "Bill", age: 51}
19 console.log(person_3); // {name: "Helen", age: 27}
20
```

В переменных хранятся не сами объекты а ссылки на области памяти где они расположены, поэтому при «копировании» переменной присваивается ссылка на объект. И обе переменные позволяют работать с одним и тем же объектом. Если необходимо создать копию объекта, то помочь может оператор ... или же метод **Object.assign(...)**.

Подробнее: <https://learn.javascript.ru/object>

Подробнее: <https://learn.javascript.ru/destructuring-assignment>

Прототипы объектов

У объекта может быть объект-предок, в **JavaScript** его называют **прототипом**. Если требуемое свойство (или метод) не найден в объекте, то оно ищется у **прототипа**.

Прототип это объект который «дополняет» своими свойствами и методами другой (дочерний) объект. Установить кто у объекта будет **прототипом** можно при помощи свойства **__proto__**.

Благодаря **прототипам** в **JavaScript** можно организовать объекты в «**цепочки**» так, чтобы свойство, не найденное в одном объекте, автоматически искалось бы в другом (родительском).

Подробнее о прототипах мы поговорим в контексте «Объектно-ориентированного программирования» в JavaScript.

Подробнее: <https://learn.javascript.ru/prototypes>

Тип **Symbol**

Тип Symbol

```
1
2
3   let citySymbol = Symbol('city');
4
5   let key = 'city';
6
7   let ob = {
8     name: 'Ivan',
9     age: 23,
10    [citySymbol]: 'Dnipro'
11  }
12
13  //-----//
14
15  let a = Symbol('Jhon');
16  let b = Symbol('Jhon');
17
18  console.log(a == b); //false
19
```

Тип данных **Symbol** (символ) был добавлен в **JavaScript**, чтобы стать дополнительным идентификатором (в дополнение к строковым) свойств и методов в объектах, при этом не создавая конфликтов с уже имеющимися строковыми идентификаторами.

Каждый символ уникален, сравнения двух символов, даже с одним и тем же именем всегда даёт **false**. Поля заданные символами в объекте не перебираются циклом **for-in**.

Немного практики #2

Задача: Вводится дата в формате '**YYYY-MM-DD**' (например '**2019-05-20**') необходимо преобразовать её в формат '**20 травня 2019 р.**'

JSON | JavaScript Object Notation

JSON (JavaScript Object Notation)

JSON - текстовый формат обмена данными, удобный для чтения и написания как человеком, так и компьютером. Основан на синтаксисе (правилах записи) массивов в **JavaScript**. Формат поддерживается практически во всех современных языках программирования.

```
[ { "name": "Jane",  "age": 23 },  
  { "name": "Max",   "age": 16 },  
  { "name": "Maria", "age": 34 },  
  { "name": "Alex",  "age": 20 },  
  { "name": "Cate",  "age": 45 } ]
```

<http://www.json.org/json-ru.html>

Для работы с форматом **JSON** у нас есть два метода: **JSON.stringify(data)** – который преобразует структуру данных в строковое представление, и метод **JSON.parse(str)** который делает обратное действие.

Подробнее: <https://learn.javascript.ru/json>

WebAPI построенные на обмене данными в формате JSON

Разработчикам доступно огромное количество сервисов которые предоставляющие доступ к данным в формате **JSON**. Такого рода сервисы носят название **WebAPI**.



```
[{"ccy": "USD", "base_ccy": "UAH", "buy": "28.05000", "sale": "28.25000"},  
{"ccy": "EUR", "base_ccy": "UAH", "buy": "31.95000", "sale": "32.45000"},  
{"ccy": "RUR", "base_ccy": "UAH", "buy": "0.41500", "sale": "0.43500"},  
{"ccy": "BTC", "base_ccy": "USD", "buy": "6143.7724", "sale": "6790.4852"}]
```

<https://api.privatbank.ua/>

WebAPI/JSON Національного Банку України



Національний
банк України

<https://bank.gov.ua/ua/open-data/api-dev>

3. Множество (Set)

Множество / Set

```
2
3   let set = new Set();
4
5   set.add("Jhon");
6   set.add("Helen");
7   set.add("Jhon");
8   set.add("Maria");
9   set.add("Jane");
10  set.add("Bill");
11
12  console.log(set); // {"Jhon", "Helen", "Maria", "Jane", "Bill"}
13
14  console.log( set.has('Maria') ); //true;
15  console.log( set.has('Samuel') ); //false
16
17  set.delete("Jane");
18  console.log(set); //{"Jhon", "Helen", "Maria", "Bill"}
19
```

Set – коллекция без ключей (создаётся при помощи ключевого слова **new**), позволяет хранить любые типы данных. Элемент множества встречаться в нём не более чем один раз. Есть возможность узнать есть ли элемент во множестве (метод **.has(...)**), а также узнать размер множества (свойство **.size**).

Избавление от дубликатов при помощи Set

```
2
3 let arr = ['Jane', 'Jhon', 'Maria', 'Alice', 'Jane', 'Peter', 'Alice', 'Donald'];
4
5 console.log(arr); //["Jane", "Jhon", "Maria", "Alice", "Jane", "Peter", "Alice", "Donald"]
6 console.log("Array length:", arr.length); //8
7
8 let set = new Set(arr);
9
10 console.log(set); // {"Jane", "Jhon", "Maria", "Alice", "Peter", "Donald"}
11 console.log(set.size); //6
12
13 let new_arr = Array.from(set);
14
15 console.log(new_arr); //["Jane", "Jhon", "Maria", "Alice", "Peter", "Donald"]
16 console.log("New array length:", new_arr.length); //6
17
```

4. Ассоциативный массив с ключами произвольного типа (**Map**)

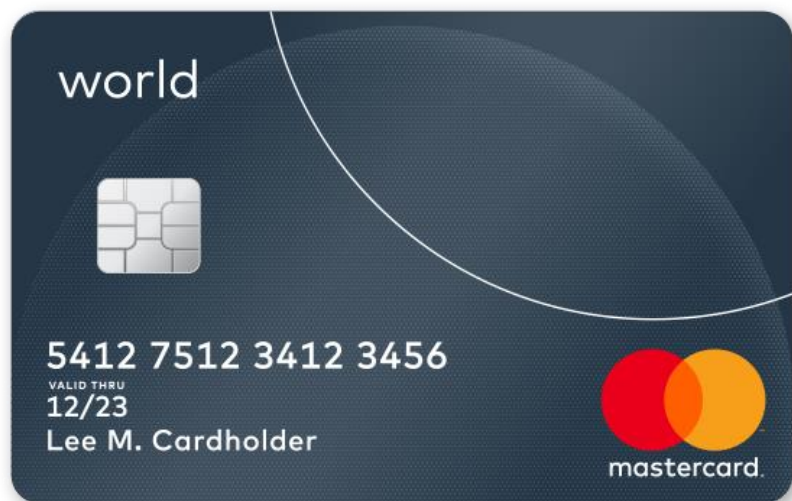
Словарь / Map

```
2
3   let ob = {};
4
5   ob[123]    = "Hello";
6   ob['123']  = "World";
7
8   console.log("Object", ob);
9   // Object {123: "World"}
10
11  let map = new Map();
12
13  map.set(123, 'Hello');
14  map.set('123', 'World');
15
16  console.log("Map", map);
17  //Map(2) {123 => "Hello", "123" => "World"}
18
19
```

Как и **object** коллекция **Map** – ассоциативный массив, но ключами к нему могут выступать любые типы данных, и в отличие от **object** они не будут приведены к строке. Также в отличие от **object** в **Map** есть понятие длины, и она доступна через свойство **.size** Основная польза **Map**, в том, что ключами могут выступать объекты.

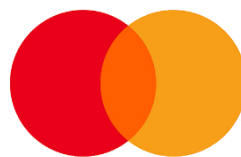
**К следующему занятию будет
полезно почитать о...**

Узнайте как работает **Алгоритм Луна**



VISA

4916 5526 5398 1949



mastercard.

5357 6872 3409 1447

Алгоритм Луна проверяет контрольную сумму числа, применяется для проверки корректности номера банковских карт.

https://uk.wikipedia.org/wiki/Алгоритм_Луна

Домашнее задание
/сделать

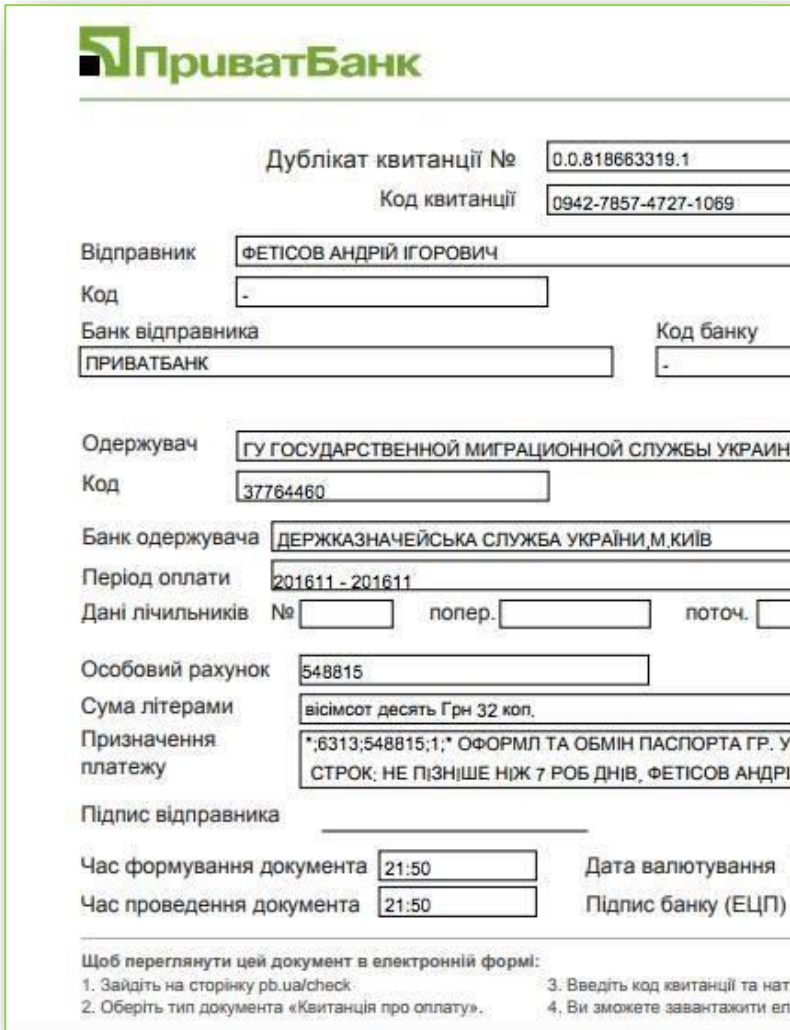
Домашнее задание #B.1

«Азбука пилотов» (или официально **фонетический алфавит ИКАО**) - стандартизированный способ прочтения букв алфавита английского языка в авиации. Каждая буква кодируется словом, которое при плохой связи позволяет с высокой вероятностью распознать букву которая передаётся. Ваша задача, написать скрипт, который будет переводить буквенно-цифровую комбинацию в набор слов из «азбуки пилотов».

Например: пользователь вводит комбинацию буквенно-цифровую, (буквы только латинские) (**например: KL1386**), а скрипт выдает «расшифровку» в соответствии с алфавитом (**например: Kilo Lima One Three Eight Six**). Регистр вводимой комбинации не должен влиять на результат (т.е. большие и маленькие буквы дают один и тот же результат).



Домашнее задание #B.2



ПриватБанк

Дублікат квитанції № 0.0.818663319.1
Код квитанції 0942-7857-4727-1069

Відправник ФЕТИСОВ АНДРІЙ ІГОРОВИЧ
Код -
Банк відправника ПРИВАТБАНК Код банку -

Одержувач ГУ ГОСУДАРСТВЕННОЙ МИГРАЦИОННОЙ СЛУЖБЫ УКРАИНЫ
Код 37764460
Банк одержувача ДЕРЖКАЗНАЧЕЙСЬКА СЛУЖБА УКРАЇНИ, М. КИЇВ

Період оплати 201611 - 201611
Дані лічильників № попер. поточ.

Особовий рахунок 548815
Сума літерами вісімсот десять грн 32 коп.
Призначення платежу *6313;548815;1;* ОФОРМЛ ТА ОБМІН ПАСПОРТА ГР. У
СТРОК: НЕ ПІЗНІШЕ НІЖ 7 РОБ ДНІВ, ФЕТИСОВ АНДРІ

Підпис відправника

Час формування документа 21:50 Дата валютування
Час проведення документа 21:50 Підпис банку (ЕЦП)

Щоб переглянути цей документ в електронній формі:
1. Зайдіть на сторінку rb.ua/check. 3. Введіть код квитанції та нат
2. Оберіть тип документа «Квитанція про оплату». 4. Ви зможете завантажити ел

Написать скрипт который будет словами записывать сумму заданную числом которое ввёл пользователь в пределах от 1 до 999 (включительно). Например **643** => «**шестьсот сорок три гривны**» (не забывая добавлять слово **гривен**, **гривна** и т.д. в зависимости от необходимого склонения).

Если задача решилась быстро и просто, то – расширяем диапазон от 1 до **999 999 999** **гривен**.