

АССОЦИАТИВНЫЕ МАССИВЫ




АЛЕНА БАТИЦКАЯ



АЛЕНА БАТИЦКАЯ

 [ABatitskaya](#)

 [@alenabat](#)



ПЛАН ЗАНЯТИЯ

1. Повторение – мать учения
2. Что такое ассоциативный массив?
3. Операции с объектом
4. Доступ к свойствам
5. Где попрактиковаться?



ПОВТОРЕНИЕ – МАТЬ УЧЕНИЯ

ВСПОМНИМ ПРОШЛЫЙ МАТЕРИАЛ

Как вы думаете, что выведет следующий код?

```
1  for (let i = 0; i <= 100; i++) {  
2      if (i % 2 !== 0) {  
3          console.log(i);  
4      }  
5  }
```

ПОСМОТРИМ НА РЕЗУЛЬТАТ

При помощи цикла `for` и условия внутри него мы выведем нечётные числа от 0 до 100 (99 в нашем случае будет последним, а 1 — первым).

В цикле мы пробегаем по всем целым числам от нуля до 100 включительно с шагом в единицу, а так как на каждом шаге мы делаем проверку на нечетность числа, то в консоль попадает только часть чисел.

ЧТО ТАКОЕ АССОЦИАТИВНЫЙ МАССИВ?

АССОЦИАТИВНЫЙ МАССИВ И С ЧЕМ ЕГО ЕДЯТ

Имена и числа мы с вами научились собирать в единое целое. А что если нам нужно описать какую-то сущность из реального мира, и одного только имени не достаточно?

ЗАДАЧА

- Интернет-магазин книг.
- Нам нужно хранить информацию о книгах.
- О каждой книге нам известны её ISBN, название, автор, цена, количество на складе.
- Разработать структуру представления информации о книгах, удобную для хранения и обработки.

ИСПОЛЬЗУЕМ «ПЛОСКИЕ» МАССИВЫ

Зарезервируем под каждую книгу 5 ячеек массива и разместим в них информацию о книге:

```
1  let books = [  
2    'b1', 'История земли', 'Филип Фрай', 946, 16,  
3    'b2', 'Моя жизнь...', 'Доктор Джон Зоидберг', 731, 2,  
4    /* ... */ ];  
5  
6  for (let i = 0; i < books.length; i += 5) {  
7    console.log('ID:', books[i]); // ID b1  
8    console.log(books[i + 1]); // История земли  
9    console.log('Цена:', books[i + 3]); // Цена 946  
10 }
```

ИСПОЛЬЗУЕМ СВЯЗАННЫЕ МАССИВЫ

```
1 let bookIDs = ['b1', 'b2' /* , ... */ ];
2 let bookTitles = ['История земли', 'Моя жизнь...' /* , ... */];
3 let bookAuthors = ['Филип Фрай', 'Доктор Джон Зоидберг' /* , ... */];
4 let bookPrices = [946, 731 /* , ... */ ];
5 let bookAmounts = [16, 2 /* , ... */ ];
6
7 for (let i = 0; i < bookIDs.length; ++i) {
8     console.log('ID:', bookIDs[i]); // ID: b1
9     console.log(bookTitles[i]); // История земли
10    console.log('Цена:', bookPrices[i]); // Цена 946
11};
```

ИСПОЛЬЗУЕМ ВЛОЖЕННЫЕ МАССИВЫ

Представим каждую книгу отдельным массивом:

```
1  let books = [  
2  [ 'b1', 'История земли', 'Филип Фрай', 946, 16 ],  
3  [ 'b2', 'Моя жизнь...', 'Доктор Джон Зоидберг', 731, 2 ],  
4  /* ... */ ];  
5  
6  for (let i = 0; i < books.length; ++i) {  
7      console.log('ID:', books[i][0]); // ID: b1  
8      console.log(books[i][1]); // История земли  
9      console.log('Цена:', books[i][3]); // Цена 946  
10 }
```

ПРОАНАЛИЗИРУЕМ ВАРИАНТЫ РЕШЕНИЯ

- Для «плоского» массива при добавлении еще одного свойства книге весь уже написанный код будет работать неверно.
- Для связанных массивов постоянно нужно следить за тем, чтобы индексы не разъехались. Добавление свойства не ломает существующий код, но чтобы использовать новое свойство, придется все переписать.
- Вложенный массив имеет минимум недостатков. Нужно только следить, чтобы порядок свойств в книге-массиве не нарушался и не совсем очевидно назначение свойств.

А ЕСТЬ ЛИ ДРУГИЕ ВАРИАНТЫ?

Что, если дать элементам книги-массива названия вместо индексов?

Для этого нам подойдет другая структура данных. Ассоциативный массив (или *Объект*) — структура данных, в которой можно хранить любые данные в формате ключ-значение.

Объект можно легко представить как паспорт человека. В паспорте есть графы: Имя, Фамилия, год рождения и так далее. Это ключи. А то, что написано в этих графах — значения.

В РЕАЛЬНОЙ ЖИЗНИ



В КОДЕ

Заполним «паспорт» для персонажа с картинки:

```
1  let person = {  
2    name: "Иван", // графа Имя со значением Иван  
3    age: 30, // графа Возраст со значением 30  
4    isMarried: false // графа Женат? со значением Не женат  
5  };  
6  
7  console.log(person);  
8  console.log(person.age); // 30
```



ОПЕРАЦИИ С ОБЪЕКТОМ

СОЗДАНИЕ ОБЪЕКТОВ

Создание и заполнение объектов очень похоже на то, что мы уже видели в лекции по массивам:

1. Объект создаётся с помощью фигурных скобок (`{ }`);
2. При создании вы можете оставить его пустым и добавить данные позже, а можете и сразу заполнить какими-то значениями (`{ isMarried: false }`, `{ name: 'Иван' }`);
3. Заполнение объекта, как вы уже заметили, заключается в добавлении пар `ключ: значение`.

ЧТЕНИЕ И ЗАПИСЬ ДАННЫХ

Доступ к свойствам объекта осуществляется по имени свойства (иногда говорят «по ключу»).

Для обращения к свойствам используется запись «через точку»:
`объект.свойство`.

```
1 let person = {};  
2  
3 // при присвоении свойства в объекте автоматически создаётся «ящик»  
4 // с именем "name" и в него записывается содержимое 'Иван'  
5 person.name = 'Иван';  
6 person.age = 30;  
7  
8 console.log(person); // { name: "Иван", age: 30 }  
9 console.log(person.name); // "Иван"
```

ЕЩЕ НЕМНОГО ПРО ЧТЕНИЕ

Альтернативный способ обратиться к свойству объекта (для того, чтобы прочитать или записать что-то в него) — это доступ через квадратные скобки:

```
1 | let person = { name: "Иван", age: 30 };  
2 |  
3 | console.log(person['name']); // 'Иван'
```

Свойство в данном случае берется в кавычки — является строкой. Также мы можем вместо строки использовать переменную:

```
1 | let field = 'age';  
2 |  
3 | console.log(person[field]); // 30
```

И ЕЩЕ ЧУТЬ-ЧУТЬ :)

А что будет, если попробовать прочитать у объекта свойство, которое мы не задали? Все просто — в случае обращения к несуществующим полям JS возвращает нам `undefined`.

```
1 | let person = { name: "Иван", age: 30 };  
2 |  
3 | console.log(person.friends); // undefined
```

УДАЛЕНИЕ

Удаление свойства из объекта осуществляется оператором `delete`.

```
1 let person = { name: "Иван", age: 30 };  
2  
3 delete person.age;  
4  
5 console.log(person); // { name: "Иван" }
```



ДОСТУП К СВОЙСТВАМ

ПРОВЕРКА НАЛИЧИЯ СВОЙСТВА В ОБЪЕКТЕ

Убедиться, что объект содержит поле с определенным именем можно с помощью оператора `in`. Но, как правило, достаточно просто попробовать прочесть это свойство и если мы получим значение, отличное от `undefined` – значит поле определено и присутствует в данном объекте.

```
1 let person = { name: "Иван", age: 30 };
2
3 console.log('name' in person); // true
4 console.log('surname' in person); // false
5 // person.hasPets === undefined – поле в объекте отсутствует
6 console.log(person.hasPets);
```

ПОПРАКТИКУЕМСЯ?

Представим, что в нашем магазине авторы и их произведения объединены в виде единого объекта:

```
1 let author = {  
2   id: '98y98jhkjh',  
3   name: 'Тургенев, Иван Сергеевич',  
4   books: ['Ася', 'Муму', 'Отцы и дети']  
5 };
```

Наша задача — вывести имя автора, а затем все его произведения, каждое на своей строчке.

ПОСМОТРИМ НА ВАРИАНТ РЕАЛИЗАЦИИ

```
1 console.log(author.name + ':\n');
2
3 for (let i = 0; i < author.books.length; i++) {
4     console.log('- ' + author.books[i] + ';\n');
5 }
```

УСЛОЖНИМ ЗАДАЧУ

А что если в массиве будут не строки, а объекты. Например, коллекция книг одного жанра.

```
1 let books = {};  
2 books.fantasy = [  
3   { author: 'Айзек Азимов', title: 'Я, робот' },  
4   { author: 'Роберт Хайнлайн', title: 'Звездный десант' },  
5   { author: 'Герберт Уэллс', title: 'Война миров' }  
6 ]
```

Задача очень похожая — вывести автора и его произведение.

ИТАК

Какой бы вложенной не была наша структура данных — принцип один и тот же — значения полей объекта мы получаем «через точку», а при работе с массивами используем циклы или встроенные методы-помощники.

```
1 | for (let i = 0; i < books.fantasy.length; i++) {  
2 |   let book = books.fantasy[i];  
3 |   console.log('Автор: ' + book.author + ', книга: ' + book.title + '.');  
4 | }
```



ЧЕМУ МЫ НАУЧИЛИСЬ?

1. Узнали, что такое ассоциативные массивы и их особенности в JS;
2. Узнали, как описать объекты реального мира в коде;
3. Поработали еще с одной структурой данных и стали на шаг ближе к профессионалам своего дела.



ДОМАШНЕЕ ЗАДАНИЕ

Давайте посмотрим на [домашнее задание](#).

- Вопросы по домашней работе задаем в группе Facebook!
- Задачи можно сдавать по частям.
- Зачет по домашней работе проставляется после того, как приняты все **3 задачи**.

ГДЕ И КАК МОЖНО ПОИГРАТЬСЯ С АССОЦИАТИВНЫМИ МАССИВАМИ

- [Codewars](#), programming challenges;
- [Объекты как ассоциативные массивы](#), `learn.javascript.ru`;
- Консоль браузера.

ЧТО ПОЧИТАТЬ

- <https://developer.mozilla.org/ru/docs/Learn/JavaScript/Объекты/Основы>
- по традиции документация на MDN;
- <https://learn.javascript.ru/array> – `learn.javascript.ru`,
опять же по традиции.



А ПОСМОТРЕТЬ?

- [Обзор основ от Sorax;](#)
- [Краткий обзор того, что мы сегодня обсудили на английском.](#)



Спасибо за внимание! Время задавать вопросы

АЛЕНА БАТИЦКАЯ

 [ABatickaya](#)

 [@alenabat](#)