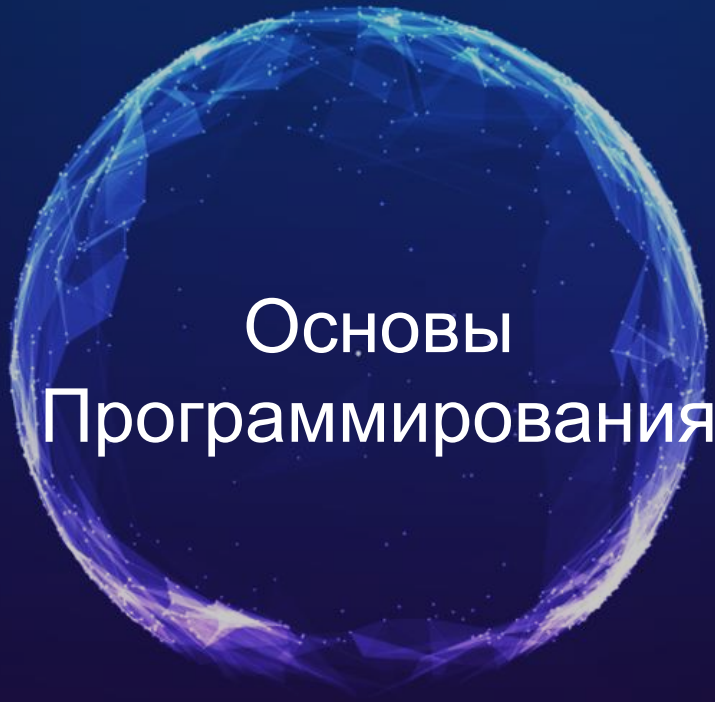

Основы Программирования

Занятие №8
Ассоциативный массив. ООП.

IT Education Academy

WWW.ITEA.UA

План занятия

- Ассоциативный массив
- Обьекты и классы
- Конструкторы, поля, методы
- Наследование
- Библиотеки

Повторение материала

Ассоциативный массив

Массив - структура данных, позволяющая хранить несколько значений, идентифицируемых по индексу - позиции элемента в массиве.

Ассоциативный массив - структура данных, позволяющая хранить несколько значений, идентифицируемых по ключу.

Т.е. в качестве «индексов» используются не только простые числа, как в массиве, но и значения ЛЮБЫХ типов.

Ассоциативный массив

С помощью таких массивов удобно хранить данные.

Например, в задаче перевода месяца в число (Июль -> 6 и тд) вместо цикла можно было бы иметь ассоциативный массив, где ключем бы был месяц, а значением число. Тогда операция поиска соответствия была бы сведена к обращению к массиву.

```
var assocArray = …;
var day = “Июль”;
var dayNum = assocArray[day]; // Обращение к ассоциативному массиву
console.log(dayNum); // 6
```

AM в JavaScript

Для создания таких массивов используется синтаксис:
var map = {key1: value1, key2: value2, …, keyN: valueN}
Где ключ и значение - значения любых типов данных.

Например:

```
var days = {‘Январь’: 0, ‘Февраль’: 1, ‘Март’: 2, ‘Апрель’: 3}
var empty = {}

var arbitraryKeyValue = {
  1: ‘num’,
  ‘text’: ‘text’,
  ‘num’: 2,
  true: ‘hi’,
  3: false
}
```

Доступ к элементам

Для доступа к элементам используется такой же синтаксис, как и в массивах, только вместо индекса - ключ.

```
var days = {‘Январь’: 0, ‘Февраль’: 1, ‘Март’: 2, ‘Апрель’: 3}
var day = days[‘Январь’];
console.log(day);

day[‘Май’] = 4
```

```
var empty = {}
empty[1] = ‘text’;
empty[true] = ‘true’
empty[‘true’] = true

console.log(empty[true]);
```

Ключи

Чтобы узнать какие ключи есть в массиве существует функция Object.keys(assocArray).
Она возвращает массив(обычный) ключей.

```
var example = {‘a’ : 1, ‘b’: 2, ‘c’: 3}
var keys = Object.keys(example);
console.log(keys); // [‘a’, ‘b’, ‘c’]

//перебор значений и вывод на экран
for (var i = 0; i < keys.length; i++) {
  var key = keys[i];
  console.log(key + “ -> “ + example[key];
}
```

Цикл

Существует упрощенный цикл для перебора ключей в ассоциативном массиве.
Синтаксис:
for (var key in array) { … }

```
var example = {‘a’ : 1, ‘b’: 2, ‘c’: 3}

//перебор значений и вывод на экран
for (var key in example) {
  console.log(key + “ -> “ + example[key];
}
```

Ассоциативный массив

1. Спросить у пользователя ввести оценку ECTS(болонская система оценивания), вывести на экран соответствующую оценку по 5-ти бальной системе:
A - 5
B - 4
C - 4
D - 3
E - 3
Fx - 2
F - 2

2. Написать функцию createUser - которая принимает два аргумента - имя и возраст и возращает ассоциативный массив с двумя значениями name и age. Написать функцию showMap которая принимает один аргумент - ассоциативный массив и выводит на экран все пары ключ-значение. Проверить их работу.

Пользовательские обьекты*

* - пользователь - это программист, а не пользователь программы.

Для моделирования реального мира неудобно использовать простые типы данных: строки, числа, массивы. Часто реальные обьекты в природе обладают несколькими свойствами одновременно:
У человека есть имя, фамилия, возраст, адрес и тд.

Для этого удобно использовать более сложные типы данных, которые описываются с помощью классов и обьектов.

Классы позволяют создавать свои типы данных, которые обладают несколькими свойствами.

Классы и обьекты

Класс - это описание (шаблон) того, какими свойствами будут обладать значения этого типа данных.

Обьект класса - это созданное значение этого типа данных, созданный экземпляр класса.

Можно сравнить:

Пользовательский ТД	Стандартный ТД
Класс	Number, String
Обьект класса	1, “text”

Классы и обьекты

В классе описывается внутреннее состояние обьектов (переменные, которые могут быть изменены). Эти внутренние переменные назваются полЯми класса. Поле класса - переменная, хранящая значение.

Для обращения к этой переменной определенного обьекта используется оператор точка «.».
Он уже использовался для обращения к свойству массива - length.

Поля

Например, существует класс Person, обладающий двумя полями name и age.

```
//…
//описание класса
//…

var person = … // создание экземпляра класса

var name = person.name // обращение к полю(свойству) name объекта person
                       // принадлежащему классу Person
console.log(person.age)  // ображение к полю age

person.name = ‘John’; // изменение значения поля
```

Создание обьекта. Конструктор

Класс обладает специальной функцией, которая позволяет создавать объекты этого класса - конструктором. Обьекты определенного класса могут быть созданны только с помощью его конструктура.

В конструкторе задается начальное состояние обьекта класса - записываются значения полей.

Класс и конструктор в JavaScript

Для описания класса и его конструктора в JavaScript используется следующий синтаксис:

```
class ClassName {
  constructor(arg1, arg2, arg3) {
    … // тело (блок команд)
  }
}

class Book {
  constructor(author, pages, title) {
    …
  }
}
```

Класс и конструктор в JavaScript

Описать класс Person с конструктором, принимающим два аргумента - name и age.
В теле конструктора выводить на экран текст “About to create Person(___, ___)”, где на месте пропусков - имя и возраст.

Создание экземпляра класса

Для создания обьекта (экземпляра класса) используется его конструктор. Для вызова конструктора используется синтаксис:

new ClassName(arg1, arg2, …, argN)

Например:

var book1 = new Book(‘Uncle Bob’, 1000, ‘Clean code’);
var book2 = new Book(‘David Flanagan’, 1100, ‘JavaScript: The Definitive Guide’);

В момент выполнения строки new Book(…) интерпретатор выделяет место в оперативной памяти для хранения обьекта, а потом вызывает его конструктор, который задает начальное состояние обьекта.

Создание экземпляра класса

Создать три экземпляра класса Person.

Какими свойствами обладают объекты класса Person?

Конструктор и ключевое слово this

При создании объекта для описания его полей необходим механизм обращения к будущему объекту. Для этого существует ключевое слово this. this выступает в виде переменной, которая хранит в себе будущий объект.

```
class Book {
  constructor(author, pages, title) {
    this.author = author;
    this.pages = pages;
    this.title = title;
  }
}

var book = new Book(“Me”, 0, “My diary”);
console.log(book.title);
```

Конструктор и ключевое слово this

Переписать конструктор класса Person, чтобы он создавал необходимые поля в будущем объекте.

Написать функцию showPerson которая принимает один аргумент - person и выводит на экран “___ is ___ years old”, где вместо пропусков - имя и возраст переданного значения.

Функции внутри класса

ООП позволяет хранить необходимые функции внутри того касса, к которому они имеют отношение.

Функции внутри класса называются мЕтодами.
Метод позволяет выполнять блок команд на основе текущего состояния того объекта на котором он был вызван.

Для вызова метода объекта также используется оператор точка «.».
Вы вызывали методы, когда работали со строками .split(...) и массивами .push(…).

Функции внутри класса

Например, если бы в классе Book был метод под названием isTooBig() который бы возвращал true если в книге больше 500 страниц, чтобы вызвать этот метод на объекте использовали бы:

```
//… class Book …

var book1 = new Book(“Me”, 0, “My diary”);
var book2 = new Book(“Flanagan”, 1100, “JavaScript: The definitive guide”);

var book1Big = book1.isTooBig(); // false
var book2Big = book2.isTooBig(); // true
```

Описание методов

```
class Book {
  constructor(author, pages, title) {
    this.author = author;
    this.pages = pages;
    this.title = title;
  }  //end of constructor

  isTooBig() { //обычная функция в которой можно обратиться к свойствам объекта
    return this.pages > 500
  } //end of isTooBig function

}
var book1 = new Book(“Me”, 10, “My diary”);
var book2 = new Book(“Flanagan”, 1100, “JavaScript: The definitive guide”);
var book1Big = book1.isTooBig(); // false
var book2Big = book2.isTooBig(); // true
```

Метод VS Поле

Поле позволяет хранить состояние объекта.

Метод позволяет выполнять какие-либо действия на основе текущего состояния.

date.getTime()	console.log(‘hi’)	text.replace(‘a’, ‘A’)
text.length	process.arch	array.length
array.push(1)	url.host	book.isTooBig()

Метод

Создать метод show в классе Person который выводит на экран “___ is ___ years old”.

Наследование

ООП позволяет наследовать функционал другого класса.
Например, в можно выделить класс Животное, которое обладает некоторыми свойствами. И создать два класса наследников: собака и птица.

Наследование

```
class Animal {
        constructor(speed, sound) {
          this.speed = speed;
          this.sound = sound;
        }

        say() {
          console.log(sound);
        }

        move() {
          console.log(this.type + "ing with speed " + this.speed);
        }
}
```

Наследование

```
class Dog extends Animal {
        constructor() {
                super(10, "Bark");
                this.type = "Run";
        }

        eat() {
                console.log('Eating meat');
        }
}
```

Наследование

```
class Bird extends Animal {
        constructor() {
                super(15, "tweet");
                this.type = "Fly";
        }

        fly() {
                this.type = 'Fly';
        }

        walk() {
                this.type = 'Walk';
        }
}
```

Наследование

```
var animal = new Bird();
animal.move()
animal.walk()
animal.move()
animal.fly()
animal.move()
animal = new Dog();
animal.move()
animal.eat()
```

Наследование

Animal	Dog	Bird
speed	speed	speed
sound	sound	sound
	type	type
move()	move()	move()
say()	say()	say()
	eat()	fly()
		walk()

Библиотеки, фреймворки. Что это

Библиотеки и фреймворки - это написанный кем-то другим код, упрощающий разработку ПО. Они содержат набор классов, объектов и функций, решающих определенную задачу.

Библиотеки и фреймворки 1) в интерпретируемых языках - это один или несколько файлов, содержащих исходный код; 2) в компилируемых языках - скомпилированные файлы.

Библиотеки, фреймворки. Как использовать

Необходимо подключить библиотеку в том исходном коде, где она будет использована.

В JavaScript для этого используется функция require(…) принимающая имя файла, который необходимо подключить, результат выполнения - загруженный модуль библиотеки.

Например. Существует файл MyLibrary.js, содержащий мой набор функций.

var myLibrary = require('MyLibrary')

Библиотеки, фреймворки. Как получить

1) На официальном сайте создателя(ей) библиотеки. Скачать файлы и переместить их в нужное место.
2) Использовать менеджеры пакетов - специальные программы, которые «знают» откуда качать библиотеку и куда разместить ее. В node.js есть такой менеджер пакетов - npm

npm

npm позволяет загружать необходимые библиотеки.
Пример: библиотека позволяющая запрашивать ввод пользователя в node.js.

1) Создать папку на рабочем столе lesson8
2) Перейти в нее с помощью консольного окна
3) В консольном окне с помощью npm скачать библиотеку: «npm install readline-sync»
4) Удостовериться что появилась папка node_modules
5) С помощью notepad++ создать файл example.js в папке lesson8:

```
var readline = require('readline-sync');

var username = readline.question('What is your user name? ');
console.log("User name is " + username);
```

readline-sync

Подключить библиотеку readline-sync.
С помощью метода question(…) запросить у пользователя имя, потом возраст.
Создать экземпляр класса Person.
Вывести его на экран с помощью метода show().

Summary

1. Ассоциативный массив - удобный способ хранить несколько значений.
2. Класс - шаблон, объект - экземпляр класса.
3. Поля - переменные, методы - функции.
4. Конструктор - функция, которая создает объект.
5. Библиотеки - набор функций упрощающих разработку.

Домашнее задание

1. Создать функцию toMap которая принимает массив, и создает асоциативный массив, в котором ключи - это элементы масива, а значения - их индексы. Например: toMap([‘a’, ‘b’, ‘c’]) // {‘a’ : 0, ‘b’: 1, ‘c’: 2}
2. Создать класс User со свойствами name и friends. friend - это массив других объектов User. Создать метод в классе User - makeFriends(other) который принимает другого пользователя и делает их друзьями; и метод isFriends(other) который проверяет что два пользователя являются друзьями.

```
var u1 = new User(‘John’);
var u2 = new User(‘Mike’);
console.log(u1.isFriends(u2)); //false
u1.makeFriends(u2)
console.log(u1.isFriends(u2)); //true
console.log(u2.isFriends(u1)); //true --- Важно проверить чтобы работало!!!
```

2** Создать метод isFriendsOfFriends который проверяет является ли пользователь другом друга. u1 -> friend of u2; u2 -> friend of u3; u1 friend of friend u3? true

Домашнее задание

3.1 Создать класс Book (title, author), создать метод show() - выводит на экран книгу.

3.2 Создать класс BookStore, в котором хранится список книг. Создать методы в классе addBook(book) - добавляет книгу, getAll() - возвращает список книг, findByTitle(title) - находит книгу по названию, findByAuthor(author) - находит книгу по имени автора.

3.3. Написать консольную программу, которая в цикле запращивает пользователя ввести команду: (Add, Find, Exit).

Если Add -> запрашивает пользователя ввести название книги и автора, добавляет книгу в хранилище (bookstore.addBook(…)).

Если Find -> запрашивает ввести команду (All, Author, Title). Если Author или Title -> запрашивает ввести имя автора либо название и выводит на экран найденную книгу. Если All -> выводит на экран все книги.

Если Exit -> завершает работу программы.

Спасибо за внимание!