УДК 004.9 **doi: 10.26906/SUNZ.2021.3.096**

A. Maslov, O. Dzuban, T. M. Derkach, T. A. Dmytrenko

National University «Yuri Kondratyuk Poltava Polytechnic», Poltava, Ukraine

OPERATIONAL ASPECTS OF WEB-APPLICATIONS DEVELOPMENT BASED ON JAVASCRIPT FRAMEWORK

Abstract. The problem of using hypertext markup language HTML5 and cascade style sheets CSS3 while developing the visual interface of applications in development framework based on programming language JavaScript was studied. The performance features of main components of application infrastructure were analyzed. JSON format usage and influence on application operation of such elements as repository data base, WEB-services, server and client side portion of application and content delivery network were investigated. The main criteria of development socially minded applications with attractive visual interface on JavaScript and HTML5 were defined. The analysis of operational aspects of JavaScript programming language and its usage while developing visual interface of applications and its further promotion in the world of hardware and software was carried out. The research result is the information system development for courier delivery.

Keywords: JavaScript, HTML, JSON, REST, Ajax, browser, application, repository data base.

.

Introduction

After release version of HTML5 [1] appeared in the beginning of current decade, this markup language began to be used not only as language for determining the web pages content, but also as markup language for visual interfaces of applications in development framework based on the programming language JavaScript [2]. JavaScript is a programming language that allows you to make a Web-page interactive, which responds to user actions. It helps to make website pages more interactive, processes actions of website users. JavaScript widely uses the capabilities of the environment, where it is run. JavaScript became even more popular among developers, when AJAX-technology appeared, which led to a new stage in websites development. This programming language is usually used as a built-in language for programmic access to application objects. It is most widely used in browsers as a scripting language to make web-pages interactive. Along with HTML and CSS, JavaScript is the third most important block, on which most standard web interfaces are based. To understand how the language works, the user needs the basic knowledge of HTML and CSS [3].

The aim of the article is to analyze the operational aspects of the JavaScript programming language and its usage while developing visual interfaces of applications and its further promotion in the world of hardware and software.

The main material

For a long time there were no means of external data storage using JavaScript. If it was necessary to save the data, then you had to send the form to a webserver and wait until the page reloaded. This interrupted the process of developing fast and dynamic web applications. With appearance of Ajax technology (Asynchronous Javascript and XML) [3] in Internet Explorer, which was established by Microsoft the situation has changed [3]. Shortly afterwards, software support for all the features of XMLHttpRequest object that underlies this technology was added to other browsers.

In the beginning of 2004, Google launched the Gmail email application. The appearance of this service became an innovative step for Google, as the application provided its users with almost unlimited storage for their e-mails. Gmail also marked an important innovation in the functional structure of web applications: web-page reloading was in the past[4]. In general, Gmail due to using new Ajax technology is single-page, fast and sensitive web-application, which has forever changed the way of developing applications of this type. Since that time, web-developers created applications of almost all types including large-scale, cloud based productivity suites, social API, like Facebook's JavaScript SDK, and even graphically intense video games.

JavaScript has very rich object-oriented (OO) capabilities. JSON standard (JavaScript Object Notation), which is used in almost all modern web applications for both communication and data storage is a subset of an excellent notation of JavaScript object literals. JavaScript uses prototypal inheritance pattern. Prototypes of objects are used instead of classes. New objects automatically inherit the methods and attributes of their parent object through a chain of prototypes [1].

While each application written in JavaScript is unique, most of them have some common elements, such as hosting infrastructure, resource management, presentation, and behaviour of interface user.

The object notation JavaScript (JSON) is an open standard, developed by Douglas Crockford. JSON is a subset of JavaScript object literals syntax for using in data presentation, transmission, and storage. Before the JSON specification appeared, most client-server communications were performed using much more verbose snippets in the XML markup language. JavaScript developers use many web services that use JSON notation and often define internal data using JSON syntax.

The JSON format is almost identical to the syntax of JavaScript object literals with a few important differences: all attribute names and string values must be in double quotes. Other values may appear in their literal form; JSON records cannot contain cyclic links; JSON cannot contain functions.

Before extended markup language XML (Extensible Markup Language) and data repositories JSON appeared almost all services maintained relational database management system (RDBMS). RDBMS contain

discrete data points in tables and group the data that is displayed by viewing the tables in response to requests written in the language of structured SQL requests [2].

NoSQL data repositories store all records as documents or document fragments, without using table-based structured memory. Document-oriented data repositories typically store data in XML format, while object-oriented data repositories typically use JSON format. The latter is particularly well suited for web application development, as the JSON format is used internally for data exchange in JavaScript language.

Examples of popular JSON-based NoSQL data repositories include MongoDB and CouchDB. Despite the current popularity of NoSQL, it is still possible to find modern JavaScript programs based on MySQL databases and similar RDBMS.

State transmission architecture REST (Representational State Transfer) is client-server communication architecture, which provides division of entities between data resources and user interfaces (or other consumers of information resources, such as data analysis tools and aggregators). Web services that fully follow REST architecture are called RESTful. The server manages data resources (such as user records), but does not implement or include user interface. Clients are free to implement the user interface (or not implement it) in any form and in any language [4]. REST architecture does not deal with how user interfaces are implemented. It only deals with maintaining the state of the application between the client and the server.

RESTful web services use HTTP method verbs to report the server what actions the client means. The following actions are provided (Fig. 1): create a new item in a resource collection: HTTP POST; get a resource view: HTTP GET; update (replace) resource: HTTP PUT; delete a resource: HTTP DELETE. This corresponds to the four basic functions of the CRUD interface (create, retrieve, update, delete), designed to work with persistent data repositories. It is just necessary to remember that in the REST view, the "update" action actually means the "replace" action. Beginners spend more than one day to create such crud. An experienced developer, in well-developed framework, this process takes several hours at the most. Slim, like other micro-frameworks, does not provide any means of automation, so you have to do a lot by yourself. For training purposes, this is justified, but in industrial development, what can be automated must be automated.

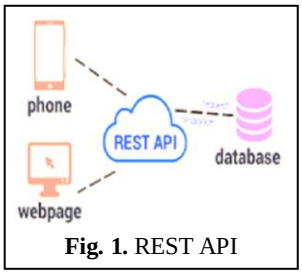

Fig. 1. REST API

PUT or POST are usually confused to change a resourse. REST makes it easier to resolve this collision PUT is used if the client is able to generate its own secure identifiers (IDs). In all other cases, POST is carried out to create a new resource. In such cases, the server generates a resource identifier (ID) and returns it to the client.

For example, you can create a new user including / users / in the POST request, in the result the server will generate a unique ID that can then be used to access the new resource in / users / userid. The server will return a new user view with its own unique URI. An existing resource cannot be modified with a POST request; a descendant can only be added to it [1].

Official HTTP RFC defines POST:

- Annotation of existing resources;
- Posting a message on a blackboard, in a newsgroup, mailing list or similar group of articles;
- Providing a block of data, such as the result of sending the form, the data processing;
- Extension of the database with an additive operation. [9]

PUT request must be used if you want to change the displayed user name by specifying / users / userid to record the user being updated. It should be noted that this request completely replaces the user record, so you need to make sure that PUT view contains everything that is needed. The difference between PUT and POST is that PUT is idempotent, it means that single and multiple calls of this method, with an identical data set, will have the same result (without irrelevant effects), in the case of POST, a multiple call with an identical data set can cause irrelevant effects.

Based on the analysis, an information system for courier delivery was developed.

The use case diagram was built, which reflected the relationship between actors and use cases in the system design (Fig. 2). For the system functioning, the base has been designed and developed, in which connections between objects are built (Fig. 3). The emphasis was on creating an easy-to-understand and use a web resource in the architecture design. All fields and necessary functionality are easily accessible (Fig. 4, 5)

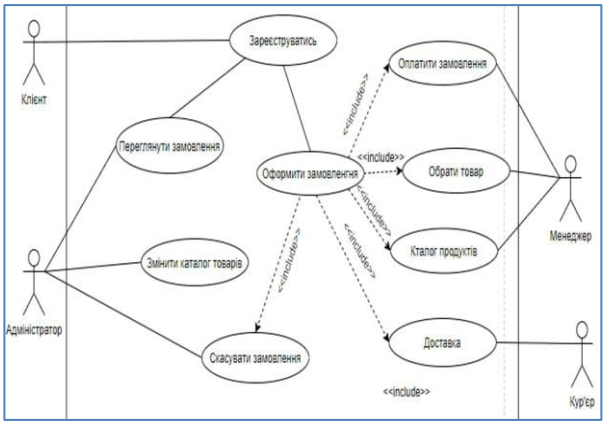

Fig. 2. The usage options diagram

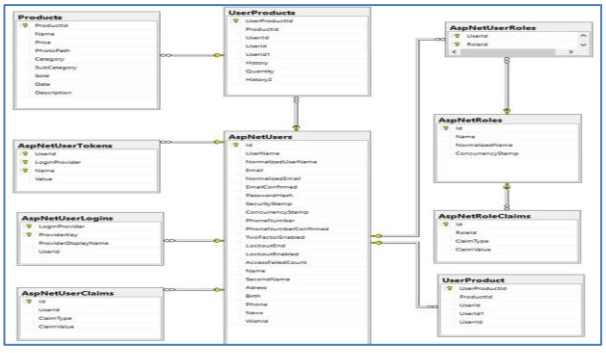

Fig. 3. The information system database scheme

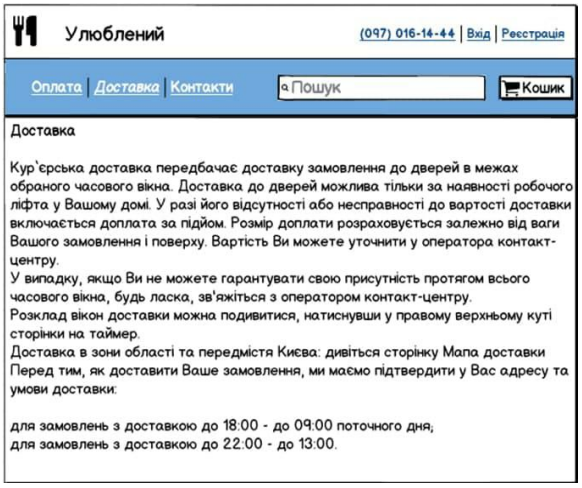

Fig. 4. The information system homepage

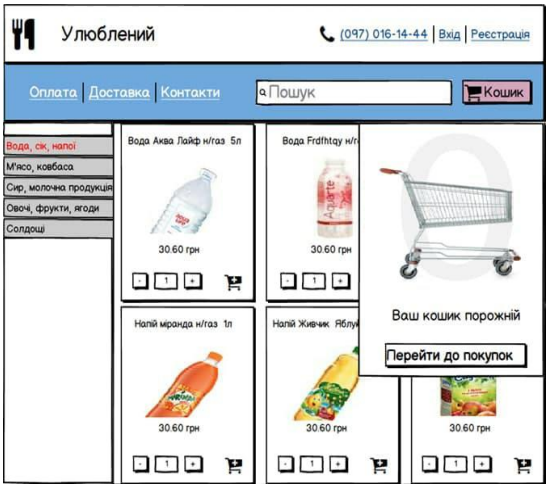

Fig. 5. Order selection page

Conclusions

The JavaScript programming language has a lot in common with languages such as C #, C ++, Java, or PHP.

The syntax of JavaScript is clear to anyone who has experience with these languages, and this explains the popularity of JavaScript, although it is important to understand that internally JavaScript is implemented quite differently from the compilers of these languages.

JavaScript has very rich object-oriented (OO) capabilities. JSON standard (JavaScript Object Notation), which is used in almost all modern web applications for both communication and data storage is a subset of an excellent notation of JavaScript object literals [4].

Currently, modern applications are developed on JavaScript, the visual interfaces of which are better than the visual interfaces of desktop applications. Recently, JavaScript is actively promoted in the world of hardware and software. Such projects as Arduino, Tessel, Espruino and NodeBots are approaching the time when JavaScript can be a common language for embedded systems and robotics. Modern JavaScript programs are the most sensitive and socially attractive among programs which were ever written. We can say that JavaScript developers are in the centre of "revolutionary" events in the history of computer engineering and the Internet in real time.

On the basis of the analysis carried out, an information system was designed and developed for creating a courier delivery order in the work.

REFERENCES

1. Brown T. Jump Start HTML5 / Tiffany B. Brown, Kerry Butters, Sandeep Panda. - Collingwood : SitePoint Pty Ltd, 2014. – 313 p.
2. Flanagan D. JavaScript: The Definitive Guide, Sixth Edition / David Flanagan. - Sebastopol : O'Reilly Media, Inc., 2011. – 1098 p.
3. Revill L. jQuery 2.0 Development Cookbook / Leon Revill. - Birmingham : Packt Publishing Ltd, 2014. – 410 p.
4. Elliott E. Programming JavaScript Applications / Eric Elliott. - Sebastopol : O'Reilly Media, Inc., 2014. – 253 p.
5. Herbert Shieldt Java 8. Complete Guide: Williams. – 2016. – 1376 p.
6. Herman David The Power of JavaScript. 68 ways to use JS efficiently; Peter - M., 2015. – 952 p.
7. Oleg Herman, Julia Geraman Programming in Java and C # for a student: Peter. – 2015 – 512 p
8. Chaffer D. Learning jQuery 1.3. Effective JavaScript web development; Plus symbol - M., 2015. – 391 p.
9. Spielman Sue JSTL. A practical guide for JSP programmers; KUDITS-Image - M., 2016. - 272 p.
10. Stanislav Davydov IntelliJ IDEA. Professional Java programming:Peter. – 2015. – 800 p.

Received (Надійшла) 23.06.2021
Accepted for publication (Прийнята до друку) 18.08.2021

Функціональні аспекти створення web-додатків на основі технологій JavaScript

А. В. Маслов, О. С. Дзюбан, Т. М. Деркач, Т. А. Дмитренко

Анотація. Розглянуто питання використання мови гіпертекстової розмітки HTML5 та каскадних таблиць стилів CSS3 при розробці візуальних інтерфейсів додатків в платформах розробки на базі мови програмування JavaScript. Проаналізовані особливості функціонування основних компонентів інфраструктури додатка. Досліджено використання формату JSON та вплив на роботу додатка таких елементів, як сховище даних, веб-сервіси, серверна та клієнтська частини додатка та мережа доставки контенту. Сформульовані основні критерії розробки на JavaScript і HTML5, соціально значущих додатків з привабливими візуальними інтерфейсами. Проведено аналіз функціональних особливостей мови програмування JavaScript та використання її під час розробки візуальних інтерфейсів додатків й подальшого просування її у світ технічних засобів.

Ключові слова: JavaScript, HTML, JSON, REST, Ajax, браузер, додаток, сховище даних.