Woody International Publish Limited
An Multidisciplinary Academic Journal Publisher
Journal of Artificial Intelligence and Information, Volume 2, 2025
https://www.woodyinternational.com/

Research on the Application of AI Functional Programming in JavaScript

Sihong Hao

School of Computer and Software, Jincheng College, Sichuan University, Chengdu 611731, China

Abstract: ***Traditional JavaScript development mainly uses object-oriented programming patterns, interspersed with some functional programming ideas. In the latest ES6 standard, functional programming is increasingly used in front-end development. This article introduces several main features, applications, and advantages and disadvantages of functional programming in JavaScript.***

Keywords: Functional programming; Closure; Partial function.

1. INTRODUCTION

Functional programming has a long history, but it was previously considered a theoretical concept that could not be applied to practical production environments. However, in recent years, functional programming has become widely used in various computer programming languages, and JavaScript, as an important language for front-end development, naturally needs to join the team of using functional programming.

Functional programming is a programming paradigm that views computer operations as mathematical function operations. A very important theoretical foundation of functional programming is lambda calculus, and the focus of lambda calculus is that functions can also be operated on as parameters.

Several studies demonstrate the power of AI in image processing. Yan et al. (2024) investigate image super-resolution reconstruction using convolutional neural networks. Chen et al. (2022) present a one-stage object referring method integrating gaze estimation, improving object recognition accuracy. Chen et al. (2020) utilize deep learning for printed mottle defect grading, showcasing AI's applicability in industrial quality control. Tian et al. (2024) further advance medical image analysis with an improved U-Net model for brain tumor segmentation. Xu et al. (2024) demonstrate the use of YOLOv5 for real-time object detection in an automated surveillance context. These studies collectively illustrate the increasing sophistication and diverse applications of deep learning in computer vision. A significant portion of the literature focuses on the applications of big data and AI in finance. Tekaya et al. (2020), Hasan et al. (2020), Awotunde et al. (2021), Ravi & Kamaruddin (2017), Eltweri et al. (2021), and Shakya & Smys (2021) collectively explore various applications of big data in finance, including risk management, fraud detection, and customer segmentation. Murugan (2023) specifically focuses on large-scale data-driven financial risk management and analysis using machine learning. VenkateswaraRao et al. (2023) detail a credit investigation and risk management system for commercial banking. Bi et al. (2024) present research on a financial intelligent risk control platform based on big data and deep machine learning. These works highlight the crucial role of data analytics and AI in mitigating financial risks and improving decision-making processes within the financial sector. Qi and Liu (2024) demonstrate the application of big data analysis (using Hadoop) to sales forecasting, offering a practical application of these techniques. The application of NLP is shown in several studies, particularly concerning dialogue systems. Ren (2024) proposes a novel approach for role-oriented dialogue summarization, focusing on balancing contributions from different participants. Lu (2024) utilizes machine learning to enhance chatbot user satisfaction. Wu (2024) explores the use of large language models (LLMs) for semantic parsing in database query engines. These studies highlight the growing role of AI in improving human-computer interaction and information retrieval. Foundational works in NLP by Jurafsky & Martin (2007), Bethard et al. (2008), Nadkarni et al. (2011), and Teller (2000) provide the essential background for these advancements. Chen et al. (2024) demonstrate AI-driven threat detection for enhancing cybersecurity. Chen and Bian (2019) present a streaming media live broadcast system using MSE. Liang and Chen (2019) introduce a high-performance dynamic service orchestration algorithm for hybrid NFV networks. Zheng et al. (2024) propose enhancing deep learning optimizers through adaptive friction. Xie et al. (2024) utilize a Conv1D-based approach for legal citation text classification. Wang et al. (2024) present a graph neural network-based recommendation system for football formations. Zhu et al. (2024) introduce an adversarial approach for sequential recommendations. Li et al. (2024) examine the effects of policies promoting technology and finance integration on

green innovation. Shen et al. (2024) detail a dynamic resource allocation strategy for cloud-native applications. Li (2024) explores multimodal data and multi-recall strategies for enhanced product recommendations in e-commerce. Chen et al. (2024) explore the development and application of computerized data mining techniques.

2. THE CHARACTERISTICS OF JAVASCRIPT IN FUNCTIONAL PROGRAMMING LANGUAGES

Just as JS has three major features in object-oriented programming: inheritance, polymorphism, and encapsulation. JS also has corresponding characteristics in functional programming mode: functions are isomorphic, curried, high-order functions, closures, lazy evaluation, partial functions, etc. Two important features of closures and partial functions are as follows.

2.1 Closure

In the unique "nested scope" of the JS language, an internal object can search up layer by layer for all variables of its external objects that contain it, but external objects cannot access the internal object's variables inward. Therefore, it can be imagined that as long as a function can be placed inside another function and returned, its local variables can be accessed outside the function scope, and because the function inside references the variables of the external function, it will not be eliminated by the garbage collection mechanism after the external function is called.

So it is concluded that when a function defined inside a function is referenced outside its scope, a closure of that internal function is created. The function of a closure is to store internal data and assist external access to internal data.

2.2 Partial functions

Partial functions can be considered as functions used to fix one or more parameters of a function, and then return a new function that accepts parameters that are not fixed. Partial functions can be implemented through the bind function, and their expression can be semantically understood as: new function=original function. bind (parameters 1, 2,...), where the body parts of the new function and the original function are the same. When calling a new function, you only need to receive an undefined parameter value (remaining parameter value) and then perform the operation on the function body.

By fixing different parameter values to the function, different new functions can be obtained, which adds many variant functions to the original function. These variant functions are more convenient and efficient to use, without having to write a lot of repetitive code like the original function. So the function of a partial function is to add universal variants to the function, reduce code volume, and improve code writing efficiency.

3. THE APPLICATION OF FUNCTIONAL PROGRAMMING IN JAVASCRIPT

In the actual JavaScript development process, developers have used many functional programming methods to solve problems, among which the use of closures is the most widespread. Here are two practical applications of closures and currying.

3.1 Changing the Size of Page Content by Closing Packages

When creating web pages, it is inevitable to encounter animation effects that cannot be achieved solely through CSS, such as clicking on one component triggering another component's size or other style changes. If closures are not used for implementation, then a separate click event needs to be bound for each component and a separate function needs to be written for them. Because under the condition of using a regular function, if all click events are also called to the same function, it will cause each time this function is called and new parameters are passed to it, the previously passed parameters will be overwritten, and only the last passed parameter will be retained. When the mouse clicks on other components, the corresponding page changes cannot be achieved because the parameters are overwritten. So for ordinary functions, only a separate function can be written for each component's click event, and they cannot share the same function. This results in redundant writing of functions with the same body, making

JS code less concise and increasing memory consumption. This problem can also be solved by using an object constructor to create a function

As a prototype, this function has a parameter that controls the size of fonts and pictures, binds the currently called context object through this, modifies the size of page content in the object method, creates a new instance object for each component through the new keyword, and binds the method of the instance object to the click event. It is complicated to implement such an object constructor, and it is not easy to understand. Especially, pay attention to the context object currently called.

Compared to the above two solutions, using closures to implement this problem has more advantages because it requires less code and is easier to understand. In the core code of using closed packets to solve the problem in Figure 1, the peripheral function changeSize

The actual parameter of () is passed in with the value size that needs to be changed, while the inner function returns the processed text size and image width. Here, the inner layer references the outer layer parameters to save the data and prevent it from being processed by the garbage collection mechanism. Then, based on the different parameters passed in, different variant functions are created (similar to fixed parameters for partial functions), and these variant functions are bound to the click events of different components according to the corresponding parameter values.

```
function changeSize(size){
    return function(){
         document.body.style.fontSize = size+"px";
         p1.style.width = size*10+"px";
         p1.style.height = size*10+"px";
     }}
```

Figure 1: Closure code

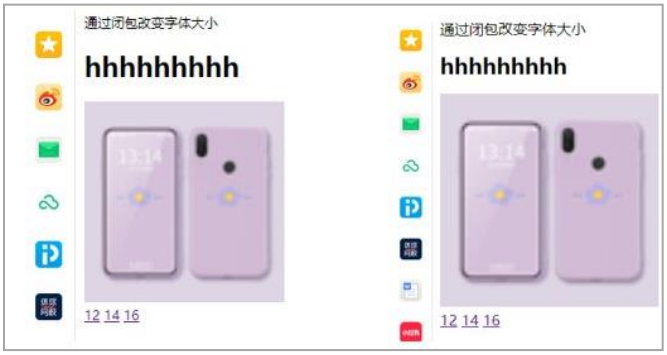

Figure 2: Page effect

When the mouse clicks on 12 to display the left side of Figure 2, and clicks on 14 to display the right side of Figure 2, comparing the navigation bar on the right, it can be seen that when clicking on 12, the font size and image height and width are smaller, which proves that using function closures can change the size of page content.

3.2 Recording monthly expenses through function currying

In JS functional programming, currying is an extension of closures, which refers to the process of splitting a function that originally contained multiple parameters into a new function that accepts a single parameter. Curry functions do not execute immediately after passing parameters, but instead form a closure through nested functions to save data. When the function needs to be used, the previously passed parameters will be used for evaluation at once.

If function currying is not used for this problem, the solution is to first put all monthly expenses into an array, and then iterate and accumulate the array. By using this method, each time an element is added to the array, the array will be traversed and evaluated again. This solution is not flexible enough, because whenever the elements in the array change, the subsequent evaluation operation will be repeated, and the elements that were not originally changed will be forced to perform the operation again, which increases some unnecessary repetitive calculations and makes the overall running efficiency of the code low.

But when using function currying, the code becomes very flexible and adding elements is also convenient. It will perform elastic evaluation based on the provided elements. The input of parameters is also very flexible. Each time, you only need to call the Curry function and pass the monthly cost as an actual parameter. The value can be saved through the closure generated by currying. Finally, when evaluation is needed, all previously passed parameters will participate in the unified evaluation. As shown in Figure 3 of the core code below, first design a currying function. In the currying function, when the arguments object parameter is empty, that is, when the first parameter is passed in, the previously defined array ARGs is used to receive all parameters. When the parameter is not empty, the parameters in the following methods are merged with ARGs, and the function containing the arguments object is called. In this problem, when the cost function is curried and passed with a parameter such as cost (100), the function cost is not evaluated, but returns a function and passes 100 into the arguments. This process is repeated no matter how many times the value is passed. Only when the cost() function is called separately at the end, will the function with arguments object inside the cost be executed, evaluated once, and then return the total money value.

Using currying to record costs will be more flexible, as the values of arguments can be increased or decreased at any time through currying functions, and a unified evaluation will be performed at the end. This final evaluation method involves delayed execution, which is another important feature of functional programming. From this, it can be seen that currying functions are more suitable for performing accumulation calculations on large data. Because when the delayed execution determines that the data behind it is no longer needed or an error occurs, the function will immediately stop and no longer execute. This greatly reduces the probability of the program performing meaningless calculations, increases the efficiency of the program, and makes it easier for operators to understand and practice through this method.

```
var currying = function( fn ){
var args = [];
return function(){
if ( arguments.length === 0 ){
return fn.apply( this, args );
}else{
[].push.apply( args, arguments );
return arguments.callee; } } };
var cost = (function(){
var money = 0;
return function(){
for ( var i = 0, l = arguments.length; i < l; i++ ){
money += arguments[ i ]; }
return money;
} })();
var cost = currying( cost ); // 转化成 currying 函数
cost( 100 ); // 未真正求值
cost( 200 ); // 未真正求值
cost( 300 ); // 未真正求值
alert ( cost() ); // 求值并输出：600
```

Figure 3: Curry code

4. ADVANTAGES OF JAVASCRIPT FUNCTIONAL PROGRAMMING

JavaScript functional programming has many advantages, such as concise code, fast development, being more like natural language, easy to understand, and using pure functions.

4.1 Simple code and fast development

Functional programming contains features such as closures and partial functions, which make it faster than object-oriented programming in the actual project development process. This is because it uses a large number of functions to solve problems, and a small amount of function code can achieve the same goal.

4.2 Using Pure Functions

Functional programming has the characteristic of pure functions (returning the same output for the same input), which brings many benefits to functional programming, such as no side effects, testability, ability to write concurrent code, and caching [3].

5. DISADVANTAGES OF JAVASCRIPT FUNCTIONAL PROGRAMMING

In JavaScript programming, functional programming still has some shortcomings, such as the lack of immutable data structures, the absence of tail recursion optimization, and excessive encapsulation.

5.1 Lack of immutable data structures

Immutability refers to the feature that does not allow data structures to be transformed through expressions, but is missing in the JS language. For example, arrays and objects can be transformed through expressions. Variable data structures can affect the cached data if the user of the function modifies the return value.

5.2 Lack of tail recursion optimization

Tail recursion refers to a recursive call that occurs at the end of a function, so there is no need to maintain the context and stack for the next call. In JS, tail recursion optimization is lacking, and the stack space is not infinite. When encountering too many call layers, stack overflow and performance degradation may occur.

6. CONCLUSION

JavaScript lacks immutable data structures and tail recursion optimization in functional programming, but with its continuous development and more standardization of its own language, it still has a lot of room for development in functional programming languages. JavaScript and functional programming can be said to achieve mutual success. JS has become increasingly widely used and the most widely used programming language in the world, which can indirectly promote functional programming to further enter the public eye, allowing more people to use functional programming and experience its charm. Functional programming will also make the JavaScript language more standardized, providing JS with simpler and more efficient code and expanding its application scenarios.

REFERENCES

[1] Yan, H., Wang, Z., Xu, Z., Wang, Z., Wu, Z., & Lyu, R. (2024, July). Research on image super-resolution reconstruction mechanism based on convolutional neural network. In Proceedings of the 2024 4th International Conference on Artificial Intelligence, Automation and High Performance Computing (pp. 142-146).
[2] Tekaya, B., Feki, S. E., Tekaya, T., & Masri, H. (2020, October). Recent applications of big data in finance. In Proceedings of the 2nd International Conference on Digital Tools & Uses Congress (pp. 1-6).
[3] Chen, J., Zhang, X., Wu, Y., Ghosh, S., Natarajan, P., Chang, S. F., & Allebach, J. (2022). One-stage object referring with gaze estimation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 5021-5030).
[4] Murugan, M. S. (2023). Large-scale data-driven financial risk management & analysis using machine learning strategies. Measurement: Sensors, 27, 100756.
[5] Chen, J., Lin, Q., & Allebach, J. P. (2020). Deep learning for printed mottle defect grading. Electronic Imaging, 32, 1-9.
[6] Wu, Z. (2024). Large Language Model Based Semantic Parsing for Intelligent Database Query Engine. Journal of Computer and Communications, 12(10), 1-13.
[7] Awotunde, J. B., Adeniyi, E. A., Ogundokun, R. O., & Ayo, F. E. (2021). Application of big data with fintech in financial services. In Fintech with artificial intelligence, big data, and blockchain (pp. 107-132). Singapore: Springer Singapore.
[8] Qi, T., & Liu, H. (2024, September). Research on the Design of a Sales Forecasting System Based on Hadoop Big Data Analysis. In Proceedings of the 2024 2nd International Conference on Internet of Things and Cloud Computing Technology (pp. 193-198).
[9] Lu, J. (2024). Enhancing Chatbot User Satisfaction: A Machine Learning Approach Integrating Decision Tree, TF-IDF, and BERTopic.

[10] Zheng Ren, "Balancing role contributions: a novel approach for role-oriented dialogue summarization," Proc. SPIE 13259, International Conference on Automation Control, Algorithm, and Intelligent Bionics (ACAIB 2024), 1325920 (4 September 2024); https://doi.org/10.1117/12.3039616
[11] Hasan, M. M., Popp, J., & Oláh, J. (2020). Current landscape and influence of big data on finance. Journal of Big Data, 7(1), 21.
[12] Shakya, S., & Smys, S. (2021). Big data analytics for improved risk management and customer segregation in banking applications. Journal of IoT in Social, Mobile, Analytics, and Cloud, 3(3), 235-249.
[13] Ravi, V., & Kamaruddin, S. (2017). Big data analytics enabled smart financial services: opportunities and challenges. In Big Data Analytics: 5th International Conference, BDA 2017, Hyderabad, India, December 12-15, 2017, Proceedings 5 (pp. 15-39). Springer International Publishing.
[14] Chen, T., Lian, J., & Sun, B. (2024). An Exploration of the Development of Computerized Data Mining Techniques and Their Application. International Journal of Computer Science and Information Technology, 3(1), 206-212.
[15] Li, S. (2024). Harnessing Multimodal Data and Mult-Recall Strategies for Enhanced Product Recommendation in E-Commerce.
[16] Chen, H., Shen, Z., Wang, Y., & Xu, J. (2024). Threat Detection Driven by Artificial Intelligence: Enhancing Cybersecurity with Machine Learning Algorithms.
[17] Bethard, S. , Jurafsky, D. , & Martin, J. H. . (2008). Instructor's Solution Manual for Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition (Second Edition).
[18] Zhu, Z., Wang, Z., Wu, Z., Zhang, Y., & Bo, S. (2024). Adversarial for Sequential Recommendation Walking in the Multi-Latent Space. Applied Science and Biotechnology Journal for Advanced Research, 3(4), 1-9.
[19] Chen, H., & Bian, J. (2019, February). Streaming media live broadcast system based on MSE. In Journal of Physics: Conference Series (Vol. 1168, No. 3, p. 032071). IOP Publishing.
[20] Zheng, H., Wang, B., Xiao, M., Qin, H., Wu, Z., & Tan, L. (2024). Adaptive Friction in Deep Learning: Enhancing Optimizers with Sigmoid and Tanh Function. arXiv preprint arXiv:2408.11839.
[21] Xie, Y., Li, Z., Yin, Y., Wei, Z., Xu, G., & Luo, Y. (2024). Advancing Legal Citation Text Classification A Conv1D-Based Approach for Multi-Class Classification. Journal of Theory and Practice of Engineering Science, 4(02), 15–22. https://doi.org/10.53469/jtpes.2024.04(02).03
[22] Z. Ren, "Enhancing Seq2Seq Models for Role-Oriented Dialogue Summary Generation Through Adaptive Feature Weighting and Dynamic Statistical Conditioninge," 2024 6th International Conference on Communications, Information System and Computer Engineering (CISCE), Guangzhou, China, 2024, pp. 497-501, doi: 10.1109/CISCE62493.2024.10653360.
[23] Lin, S., Tan, H., Zhao, L., Zhu, B., & Ye, T. (2024). The Role of Precision Anesthesia in High-risk Surgical Patients: A Comprehensive Review and Future Direction. International Journal of Advance in Clinical Science Research, 3, 97-107.
[24] Bi, S., Lian, Y., & Wang, Z. (2024). Research and Design of a Financial Intelligent Risk Control Platform Based on Big Data Analysis and Deep Machine Learning. arXiv preprint arXiv:2409.10331.
[25] VenkateswaraRao, M., Vellela, S., Reddy, V., Vullam, N., Sk, K. B., & Roja, D. (2023, March). Credit Investigation and Comprehensive Risk Management System based Big Data Analytics in Commercial Banking. In 2023 9th International Conference on Advanced Computing and Communication Systems (ICACCS) (Vol. 1, pp. 2387-2391). IEEE.
[26] Xu, G., Xie, Y., Luo, Y., Yin, Y., Li, Z., & Wei, Z. (2024). Advancing Automated Surveillance: Real-Time Detection of Crown-of-Thorns Starfish via YOLOv5 Deep Learning. Journal of Theory and Practice of Engineering Science, 4(06), 1–10. https://doi.org/10.53469/jtpes.2024.04(06).01
[27] Jurafsky, D. , & Martin, J. H. . (2007). Speech and language processing: an introduction to speech recognition, computational linguistics and natural language processing. Prentice Hall PTR.
[28] Li, L., Gan, Y., Bi, S., & Fu, H. (2024). Substantive or strategic? Unveiling the green innovation effects of pilot policy promoting the integration of technology and finance. International Review of Financial Analysis, 103781.
[29] Liang, X., & Chen, H. (2019, August). HDSO: A High-Performance Dynamic Service Orchestration Algorithm in Hybrid NFV Networks. In 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS) (pp. 782-787). IEEE.
[30] Teller, & Virginia. (2000). Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition daniel jurafsky and james h. martin (university of colorado, boulder) upper saddle river, nj: prentice hall (prentice hall ser. Computational Linguistics, 26(4), 638-641.

[31] Wang, Z., Zhu, Y., Li, Z., Wang, Z., Qin, H., & Liu, X. (2024). Graph neural network recommendation system for football formation. Applied Science and Biotechnology Journal for Advanced Research, 3(3), 33-39.
[32] Tian, Q., Wang, Z., Cui, X. Improved Unet brain tumor image segmentation based on GSConv module and ECA attention mechanism. arXiv preprint arXiv:2409.13626.
[33] Eltweri, A., Faccia, A., & Khassawneh, O. S. A. M. A. (2021, December). Applications of big data within finance: fraud detection and risk management within the real estate industry. In Proceedings of the 2021 3rd International Conference on E-Business and E-commerce Engineering (pp. 67-73).
[34] Xu Y, Shan X, Guo M, Gao W, Lin Y-S. Design and Application of Experience Management Tools from the Perspective of Customer Perceived Value: A Study on the Electric Vehicle Market. World Electric Vehicle Journal. 2024; 15(8):378. https://doi.org/10.3390/wevj15080378
[35] Shen, Z., Ma, Y., & Shen, J. (2024). A Dynamic Resource Allocation Strategy for Cloud-Native Applications Leveraging Markov Properties. International Journal of Advance in Applied Science Research, 3, 99-107.
[36] Nadkarni, P. M. , Ohno-Machado, L. , & Chapman, W. W. . (2011). Natural language processing: an introduction. Journal of the American Medical Informatics Association Jamia, 18(5), 544.