Bellevue University

JavaScript

Timothy Jelinek

CS300-T301 Programming Languages

2/11/2024

1. Programming paradigm
   1. Imperative
      1. Procedural
      2. Object-Oriented
   2. Declarative
      1. Logical
      2. Functional
2. Abstraction Level
   1. Data Abstraction
      1. Data Types
      2. Data Structures
      3. Unit Abstractions
   2. Control Abstractions
      1. Syntactic Sugar
      2. Selection and Iteration
      3. Procedures
      4. Functions
      5. Higher-Order Functions
      6. Unit Abstraction
3. Language Definition
   1. Syntax
      1. Lexical Structure
      2. Tokens
   2. Semantics
      1. Operational Semantics
      2. Denotational Semantics
      3. Axiomatic Semantics
4. Language Translations
   1. Compiler
   2. Interpreter
5. Language Future
   1. Advantages
   2. Disadvantages

This business recommendation is to use JavaScript as one of our main programming languages to enhance the programs that we create. JavaScript is a practical, versatile language for creating interactive and responsive web applications. This recommendation is for optimizing our web development, contributing to our digital landscape's success and competitiveness. This language is vital in modern-day web development to engage users and to obtain and maintain their attention. Adding JavaScript to our main coding languages will be the wisest decision we have made regarding our web presence and marketing.

**Programming Paradigm**

JavaScript is both object-oriented and functional programming, which shows that it is a multi-paradigm language. Both object-oriented and functional programming are important and have their own strengths and weaknesses, but can be combined in code to take advantage of both of the paradigms to write strong code.

**Imperative**

JavaScript is a language that incorporates object-orientated programming. Object-oriented programming involves the definition by a programmer of classes of objects and then creating instances of those classes to represent individual classes. An example of using object-oriented programming would be to create a “Dog” class that has different properties like “name” and “breed” and behaviors like “bark” and fetch” that you can then create an instance of and give the instance a name, breed, and then use methods to make the dog bark or fetch.

**Declarative**

JavaScript also incorporates functional programming. Functional programming is a paradigm that is based on writing code as a series of functions. A programmer utilizes functional programming through writing small, reusable functions that are able to be combined in a variety of ways to solve problems. A good example of functional programming is writing a function to calculate the square of a number and another function to calculate the square root of a number and then combining them to solve problems like finding the distance between two points of a graph.

**Abstraction Level**

Abstraction is the means in programming to hide simplistic details to focus on higher-level problems. The two classes of abstraction mechanisms are control abstraction and data abstraction. The primitive data types used in JavaScript are number, string, Boolean, null, and undefined. The number data type includes integer and floating point numbers. The string data type contains a sequence of characters that are different lengths. The Boolean data type can only be “true” or “false”. The null data type is used for a variable that doesn’t contain anything. The undefined data type is when a value is not assigned.

**Data Abstraction**

Data abstraction is used when handling complexity through hiding not needed details for the user so that the programmer can implement more complex logic without worrying about the hidden complexity. JavaScript utilizes inheritance and composition to incorporate data abstraction. Prototypes and closures are used in JavaScript to help with data abstraction. Prototypes help abstraction take place through defining interfaces and using them. Closure gives the ability to break through the scope chain by using nested functions.

**Control Abstractions**

Control abstraction is the process of extracting the important characteristics of control through defining abstract mechanisms and the associated characteristics of them and disregarding the low-level details and entities to be controlled. JavaScript uses first-order functions and higher-order functions. First-order functions are treated like variables to store data-driven behaviors. First-order functions work fine for working on data-oriented tasks, but they are not ideal to use to multiply a set of numbers. Higher-order functions are used to take a function as an argument or return a function after execution. These higher-order functions can be very useful for linking several tasks together into a program.

**Language Definition**

JavaScript is considered the most powerful of the three front end languages.

**Syntax**

A JavaScript program is a collection of written instructions where every line of instruction is known as a statement. A statement type that is used all the time is a variable, which is used to create, name, and assign data to the variable.

**Semantics**

JavaScript is case-sensitive, and the programs won’t run if the letters are uppercase when they are supposed to be lowercase or lowercase when they are supposed to be uppercase. JavaScript also uses explicit variables to be as clear as possible of what a variable is used for, and it uses camelCase instead of spaces to show the start of different words in a variable. To show the end of a statement in JavaScript, semicolons are used. It is important to have each statement on its own line.

**Language Translations**

The interpreter and compiler are the two ways of translating to machine language. The interpreter reads the files line by line on the fly. Interpreting takes a set of instructions and returns an answer. The advantages of using interpreters are that they are easy to run because the code doesn’t have to be converted by programmers into another language, there is no compilation step before running the code, and an interpreter can be given a file and work from that translating it line by line on the fly. The disadvantage of using an interpreter is that when running the same code over and over, like when in a loop, it gets really slow.

The compiler is different from the interpreter by not translating on the fly. It works ahead of time and creates a translation of the code that is written then compiles it down to a language that the machine understands. A compiler takes a single look through the code to understand what it does. The advantages of a compiler are that they generate the code faster as they don’t need to repeat the translation during each pass through a loop, and the code gets faster over time as optimizations are done through the compiler. The disadvantage of a compiler is that it takes more time to start as it looks through the entire code at the beginning.

**Language Future**

JavaScript has been a popular language for years, and it can be said that it won’t disappear in the years to come. New JavaScript frameworks and libraries are being developed to ease the coding further of programmers to build great web applications. Recently, JavaScript has been increasingly used for tasks normally performed on the server side, such as rendering and back-end logic. This has led to the development of Node.js, and other similar technologies which allow JavaScript to be run server-side. Seeing the consistent evolution of JavaScript over the years shows that JavaScript will continue evolving in the future years.

JavaScript has been used for years and is constantly evolving. With the evolution of JavaScript, it has advantages and disadvantages.

**Advantages**

The advantages of using JavaScript are that it has client-side security, less overhead, is inherently fast, is easy to implement, is popular, reduces server load, is versatile, and has rich interfaces. JavaScript's biggest advantage is its enhanced security for client-side applications through frameworks like React and Angular. The next advantage of less overhead is that JavaScript is lightweight and doesn’t require any heavy software installations. JavaScript is also inherently fast and is able to be utilized when building high-performance applications when wanting to perform calculations and operations. JavaScript is also easy to learn and use due to its simple syntax, dynamic nature, and object-orientated nature. JavaScript is also very popular, which has led to the creation of many libraries, frameworks, and tools to make programming easier. JavaScript can run client-side, which helps reduce server load and overhead while executing the code on the programmer’s device instead of on the server. JavaScript is also very useful in the ability to be able to use it in front-end and back-end development with the use of Angular, React, and Node.js. JavaScript also allows for the creation of rich interfaces used to enhance UI.

**Disadvantages**

The disadvantages of JavaScript include security risks, interoperability, increased server load at times, lack of browser support, debugging tools not advanced, single inheritance, and slow rendering. JavaScript can have poor security if it is not implemented properly. The client-side execution can lead to increased server load and reduced performance. Unfortunately, older browsers might not support JavaScript, and it is important to test the JavaScript code on different browsers to ensure it is interpreted correctly. It is harder to identify and fix bugs and errors in JavaScript as its debugging tools aren’t as advanced as other programming languages. JavaScript might also be less useful in certain situations due to its lack of a multiple inheritance feature. Due to JavaScript’s interpreted nature and the code being executed line-by-line, it is slower than other coding languages like C++ or Java.

Sources:

Sakhniuk, M. (2024, February 12). *Two programming paradigms in JavaScript?* https://iq.js.org/questions/javascript/javascript-programming-paradigms

UH COSC 4315 Group 4. (2021, December 8). JavaScript: abstraction, data types, and expressions. *Medium*. <https://medium.com/@csg.riskgame/javascript-abstraction-data-types-and-expressions-b52767bb529e>

*JavaScript basics: syntax, semantics and best practices – Daniel Puiatti*. (n.d.). https://danielpuiatti.com/javascript-basics-syntax-semantics-and-best-practices/

Sendagi, A. (2021, December 13). Inside the Javascript engine: compiler and interpreter. *Medium*. <https://medium.com/@allansendagi/inside-the-javascript-engine-compiler-and-interpreter-c8faa638b0d9>

Panchal, R. (2023, January 3). What is the future of JavaScript ? - Rahul Panchal - Medium. *Medium*. <https://medium.com/@rahulpanchaloff/what-is-the-future-of-javascript-69b1b839108a>

Neville, M. (2023, November 28). The Advantages and Disadvantages of JavaScript. *Softjourn Inc*. https://softjourn.com/insights/the-advantages-and-disadvantages-of-javascript