Gebze Technical University

Pascal versus JavaScript

Comparison of Procedural & Object-Oriented Paradigms

Sümeyye Zeynep Gürbüz - 200104004039

CSE 341

Yakup Genç

30.10.2023

1. Introduction

Programming paradigms are a way to classify programming languages based on their features[1]. Languages can be classified into multiple paradigms. This essay will focus on procedural and object-oriented programming paradigms. In this essay, the programming language *Pascal* will represent the procedural programming paradigm, and the programming language *JavaScript* will be the representative for the object-oriented programming paradigm.

2. Paradigm Overview

A. Overview of procedural programming paradigm

The *procedural programming paradigm* is where program code is divided up into *procedures*, which are discrete blocks of code that carry out a single task. Procedures, also called subroutines or functions, contain a series of computational steps to be carried out in the order specified by the programmer.[2]
Fundamentally, the procedural code is the one that *directly instructs a device on how to finish a task in logical steps.* This paradigm uses a linear top-down approach and treats data and procedures as two different entities.[3]
Simply put, procedural programming involves writing down a list of instructions to tell the computer what it should do step-by-step to finish the task at hand.

i. Key features

- *Predefined Function*: Typically an instruction identified by a name. Usually, predefined functions are built into higher-level programming languages, but they are derived from the library or the registry, rather than the program. One example of a predefined function is 'indexOf()', which searches for the index of a character/value in a(n) string/array.
- *Local Variable*: A variable that is declared in the main structure of a method and is limited to the local scope it is given. They can only be used in the method it is defined in, and if it were to be used outside the defined method, the code will cease to work.

- *Global Variable*: A variable which is declared outside every other function defined in the code. Due to this, they can be used in all functions, unlike a local variable.
- *Modularity*: When two dissimilar systems have two different tasks at hand but are grouped together to conclude a larger task first. Every group of systems then would have its own tasks finished one after the other until all tasks are complete.
- *Parameter Passing*: A mechanism used to pass parameters to functions, subroutines or procedures. It can be done through 'pass by value', 'pass by reference', 'pass by result', 'pass by value-result' and 'pass by the name'.

B. Overview of object-oriented programming paradigm

The *object-oriented programming paradigm* aims to incorporate the advantages of modularity and reusability. This paradigm is based upon *objects*, which have both data and methods. Objects are usually instances of classes, which used to interact with one another to design applications and computer programs. The software produced using object-oriented programming paradigm is *easier to adapt to the changing requirements, easier to maintain, create modules of functionality, promote greater design, be more robust, and perform desired work efficiently*[4].

i. Key features

- *Class*: A template that consists of the data members or variables and functions and defines the properties and methods for a group of objects.
- *Object*: Nothing but an instance of a class. Each object has its values for the different properties present in its class.
- *Abstraction*: Exposing only the essential information of an object to the user and hiding the other details.
- *Polymorphism*: Being able to add different meanings to a single component.

- *Inheritance*: One of the most important features of object-oriented paradigm. It allows a class to inherit the properties and methods of another class called the parent class, the base class, or the super-class. The class that inherits is called the child class or sub-class.
- *Encapsulation*: Enclosing the data/variables and the methods for manipulating the data into a single entity called a class. It helps to hide the internal implementation of the functions and state of the variables, promoting abstraction.

3. Language Introduction

A. Introduction to Pascal

Pascal is a general-purpose, high-level language, developed for teaching programming as a systematic discipline and to develop reliable and efficient programs. Pascal offers several data types and programming structures. It is easy to understand and maintain the Pascal programs.[5]

i. Key features

- Strongly typed language
- Offers extensive error checking
- Offers several data types like arrays, records, files and sets
- Offers a variety of programming structures

B. Introduction to JavaScript

JavaScript is a dynamic, interpreted language with object-oriented capabilities. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. Its syntax is based on the Java and C languages.

i. **Key features**[6]

- Scripting language
- Interpreter based
- Ability to perform in-built function

- Case sensitive
- Lightweight and delicate
- Dynamic typing

4. Syntax Comparison

A. Syntax of Pascal

Pascal is *case-insensitive* so the compiler will accept any combination of upper and lower case letters for both reserved words and identifiers. For example identifier ‘myprocedure;’ and identifier ‘MyProcedure;’ refer to the same procedure.

i. Choice of tokens

There are five classes of tokens in Pascal.

(a) reserved words

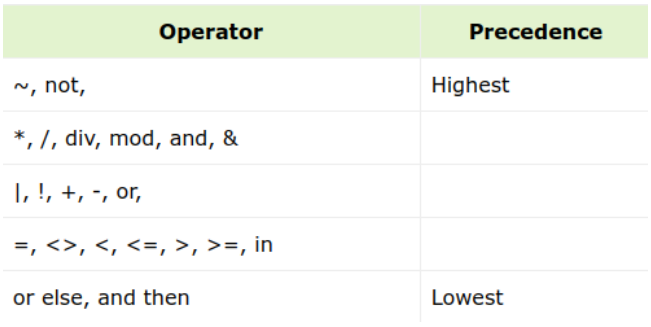
Exp: absolute, and, array, begin, case, const, else, file, object, string, etc.

(b) identifiers

Consist of between 1 and 127 significant characters (letters, digits, the underscore), of which the first must be a letter (a-z/A-Z), or an underscore.

(c) operators

Pascal has arithmetic, relational, boolean, bit, set, also string operators. Exp: +, -, *, /, =, < >, >=, *and, or, not, &*, |, *!*



Operator	Precedence
~, not,	Highest
*, /, div, mod, and, &	
|, !, +, -, or,	
=, <>, <, <=, >, >=, in	
or else, and then	Lowest

(d) separators

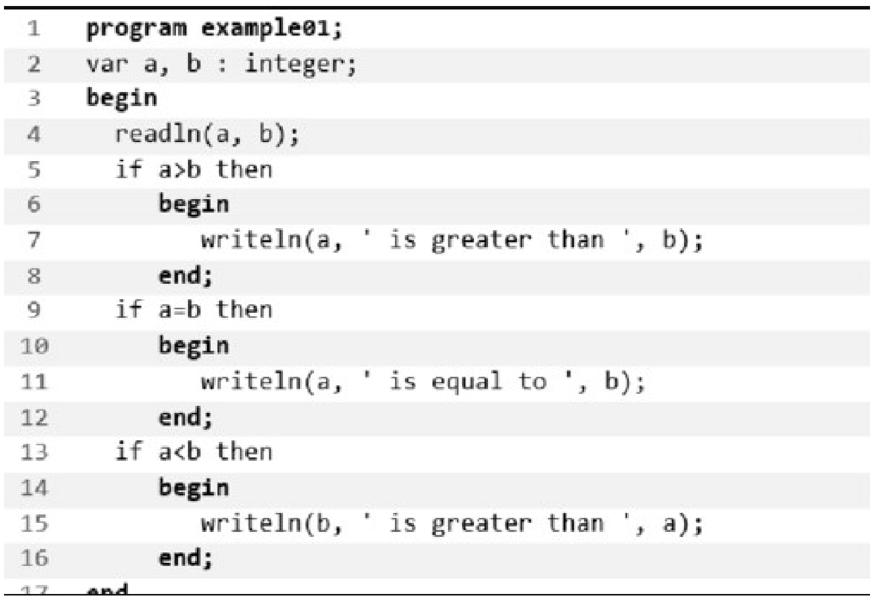
In Pascal, at least one separator must appear between consecutive identifiers, reserved words, integers, and separators cannot occur inside any other language element. The separators in Pascal are *spaces, tabs, end-of-lines, and comments.*

(e) constants

ii. Program Structure

A Pascal program basically consists of the following parts:

- program name
- uses command
- type declarations
- constant declarations
- variables declarations
- functions declarations
- procedures declarations
- main program block
- statements and expressions within each block
- comments



```
program example01;
var a, b : integer;
begin
  readln(a, b);
  if a>b then
     begin
        writeln(a, ' is greater than ', b);
     end;
  if a=b then
     begin
        writeln(a, ' is equal to ', b);
     end;
  if a<b then
     begin
        writeln(b, ' is greater than ', a);
     end;
end
```

B. Syntax of JavaScript

JavaScript *ignores spaces, tabs, and newlines*. It *allows omitting the semicolon* if each of the statements are placed on a separate line. JavaScript is a *case-sensitive* language.

i. Choice of tokens

(a) keywords

Exp: *break, if, this, true, for, false, else, function, return, continue, etc.*

(b) identifiers

Simply names that represent variables, methods, or objects. They consist of a combination of characters and digits. Identifiers can include letters (a-z/A-Z), digits (0-9), or underscore.

Valid	Invalid	Reason
current-WebSite	current WebSite	contains a space
NumberOfHits	#ofIslands	pound sign is prefixed
n	2bOrNotToBe	begins with a number
N	Return	Keyword

(c) operators[7]

JavaScript has arithmetic, assignment, comparison, string, logical, bitwise, ternary, type operators. Exp: +, -, *, **, %, ++, +=, !==, <=, &&, ||, *typeof, instanceof, &, ~, etc.*

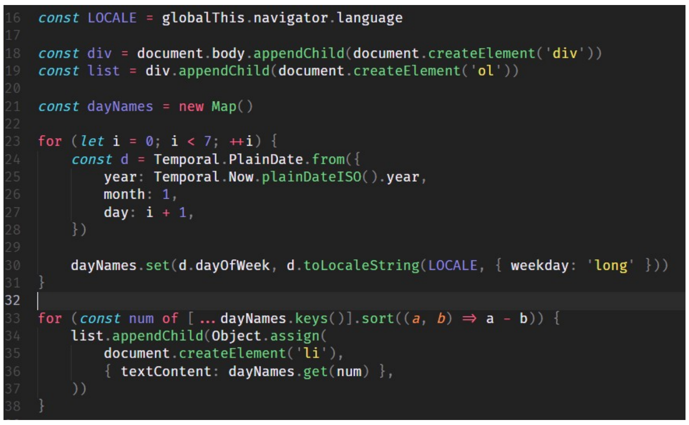
(d) separators[8]

Exp: *comma, underscore, space, equal, colon, etc.*

(e) literals (constants)

ii. Program structure

- variable declarations
- function declarations
- main program logic
- program execution



```javascript
const LOCALE = globalThis.navigator.language

const div = document.body.appendChild(document.createElement('div'))
const list = div.appendChild(document.createElement('ol'))

const dayNames = new Map()

for (let i = 0; i < 7; ++i) {
    const d = Temporal.PlainDate.from({
        year: Temporal.Now.plainDateISO().year,
        month: 1,
        day: i + 1,
    })

    dayNames.set(d.dayOfWeek, d.toLocaleString(LOCALE, { weekday: 'long' }))
}

for (const num of [...dayNames.keys()].sort((a, b) => a - b)) {
    list.appendChild(Object.assign(
        document.createElement('li'),
        { textContent: dayNames.get(num) },
    ))
}
```

C. Comparison of two syntaxes

	Pascal	**JavaScript**
Comments	// single line comment {multi-line comment}	// single line comment /* multi-line comment*/
Variable declarations	Using *var* or *const* ‘var *x: Integer’;*	Using *var, let,* or *const* *‘var x = 30;’ or ‘const z = 10;’*
Data types	Strongly typed, must be explicitly declared. *Integer, Real, Char, String, Boolean.*	Dynamic typing, not explicitly declared. *Number, String, Boolean, Object, Array, Function*
Conditionals	if condition then begin // code end else begin // code end;	if (condition) { // code } else { // code }
Loops	for i := 0 to 4 do begin // code end; while condition do begin // code end;	for (let i = 0; i < 5; i++) { // code } while (condition) { // code }
Functions	function Add(a, b: Integer): Integer; begin Result := a + b; end;	function add(a, b) { return a + b; }
Arrays	var myArray: array[1..5] of Integer;	const myArray = [1, 2, 3, 4, 5];
Object-oriented syntax	---	class Person { constructor(name) { this.name = name; } sayHello() { console.log(‘Hello ${this.name}’); } } const person = new Person(“Zeynep”); person.sayHello()

5. Semantics and Language Features

The semantics of a programming language refer to the meaning of the language's constructs and how programs written in that language are executed.

A. Pascal's semantics and language features

(a) strong typing

Pascal is known for its strong typing, which means that variables must be declared with a specific data type, and the compiler enforces type compatibility strictly. For example, if an integer variable declared, a string cannot be assigned to it. *This enhances program reliability and safety.*

(b) static typing

Pascal employs static typing, which means that variable types are determined at compile time, and type checking is done at this stage. This contrasts with dynamically typed languages like JavaScript, where type checking occurs at runtime.

(c) block structure

Pascal uses a block structure, which allows for the nesting of procedures and functions. Variables declared within a block are scoped to that block, and they don't interfere with variables in other blocks.

(d) procedural programming

Pascal is primarily a procedural programming language. It emphasizes the use of procedures and functions for structuring code. These procedures and functions can have parameters and can return values.

(e) record types

Pascal supports record types, which are used for defining complex data structures. Record types allow you to group related variables under a single name.

(f) memory management

Pascal provides manual memory management. Programmers can explicitly allocate and de-allocate memory using constructs like 'new' and 'dispose'. This level of control over memory management is a key semantic feature.

(g) strong error handling

Pascal enforces strong error handling through mechanisms like *exceptions*. It has an extensive set of built-in exception handling features to catch and handle errors gracefully.

(h) sequential execution

Pascal programs execute sequentially, from the beginning of the program to the end. There are no built-in features for parallelism or asynchronous programming as found in JavaScript.

(i) clear syntax rules

Pascal's syntax is designed to be highly readable, which contributes to the language's semantic clarity. *The use of structured blocks and clear syntax rules enhances program understandability.*

(j) scope rules

Pascal employs static scoping, which means that the scope of a variable is determined at compile time. This can lead to more predictable variable behavior within nested blocks.

B. JavaScript's semantics and language features

(a) dynamic typing

JavaScript is dynamically typed, which means that variables are not bound to a specific data type when declared. Instead, the *type of a variable is determined at runtime* based on the assigned value.

(b) prototypal inheritance

JavaScript uses prototype-based inheritance, where objects inherit properties and behaviors from other objects. There are no classes in the traditional sense, but objects can serve as prototypes for creating new objects.

(c) closures

JavaScript supports closures, which allow functions to *remember* and access variables from their containing scope, even after that scope has exited. This is a powerful feature for managing state and encapsulation.

(d) event-driven and asynchronous

JavaScript is often used in event-driven and asynchronous programming, particularly in web development. It can handle events like user interactions and perform asynchronous operations without blocking the main thread. This is essential for building responsive web applications.

(e) garbage collection

JavaScript uses automatic memory management through garbage collection. This means that developers don't need to explicitly allocate and de-allocate memory; instead, the JavaScript engine manages memory by identifying and cleaning up unused objects to prevent memory leaks.

(f) lexical scoping

JavaScript uses lexical scoping, which means that the scope of a variable is determined by its location within the source code. Inner functions can access variables from their containing functions. This is also related to closures.

(g) hoisting

JavaScript hoists variable declarations and function declarations to the top of their containing scope during compilation. This allows you to use variables and functions before they are declared in the code. However, only the declarations are hoisted, not the initializations.

C. Analyzing impact of differences on programming

Semantics of Pascal make it a well-structured and strongly-typed language that emphasizes good programming practices and clear code organization. For applications that prioritize safety, clarity, and structured design, Pascal can be advantageous, but it may not be the perfect choice for every application since its strong typing and structured nature can be restrictive in certain scenarios, particularly for tasks that require more flexibility. On the other hand, JavaScript's semantics offer a blend of flexibility, expressiveness, and versatility, making it well-suited for a wide range of applications, particularly web development.

6. Use Cases and Domains

A. Typical use cases for Pascal

Pascal had the explicit goals of teaching programming in a structured fashion and for the development of system software. A generation of students used Pascal as an introductory language in undergraduate courses[9]. Pascal is nearly a dead language but in its time it was used for mostly operating systems and also banking for it is a secure language.

Some examples for operating systems that has used Pascal[10]:

- LisaOS
- Classic Mac OS
- StreamOs
- FPOS (Free Pascal OS)
- Apple

Some examples of Pascal's use cases in banking:

- Automating banking transactions
- Calculating interest and other financial calculations
- Database management
- Risk management
- Automation of reports
- Automation of compliance

B. Typical use cases for JavaScript

JavaScript is mostly used in web development for it allows developers to create dynamic and interactive web pages. Most websites use JavaScript for validation and to support external applications, including PDF documents, widgets, flash applications.

A very popular application of JavaScript is to create interactive presentations as websites.

A recent feature of HTML5 in JavaScript is the canvas element, which allows drawing 2D and 3D graphics easily on a web page.

JavaScript can even be used to program a flying robot. With Node.js ecosystem, users can control numerous small robots, creative maker projects, and IoT devices.[11]

Some other most popular use cases of JavaScript:

- Web applications
- Server applications
- Web servers
- Games
- Smartwatch apps
- Mobile apps

7. Learning Curve

A. Learning curve for Pascal

The learning curve for Pascal can vary depending on the programmer's previous programming experience and the specific dialect of Pascal the programmer's using. However, in general, Pascal is often considered relatively easy to learn, especially for beginners.

Here's an overview of the learning curve for Pascal:

- Simplicity
- Structured programming
- Strong typing
- Standardized language

B. Learning curve for JavaScript

Mastering JavaScript is not an easy task and requires considerable effort and time investment.

Here's a general overview of the learning curve associated with JavaScript:[12]

- Basic syntax
- Fundamentals
- Browser environment
- Asynchronous programming
- Frameworks and libraries
- Advanced topics

C. Comparison of the ease of learning each language for beginners

Despite the fact that, both language has its own advantages and disadvantages, for a beginner, things might seem more complicated, particularly, at the beginning. Thus, analyzing and comparing the learning curve of each language may come into prominence.

The learning curve of Pascal distinguishes itself by its simplicity, clear syntax, and structured programming. As a matter of fact, Pascal had been used for lecturing purposes in the past. So, it can be said that learning this language might come easy for beginners.

Although Pascal is simple, the language itself and also its domain are also limited, especially for the today's technology. So, a beginner who want to code in a flexible way, and want to use one language for many different projects, JavaScript might come in handy for its flexibility, object-oriented programming, frameworks and libraries.

8. Community and Support

A. Community of Pascal

Since the original Pascal is considered as almost dead, the language itself does not have a community that would make learning the language, and solving bugs and problems easy.

B. Community of JavaScript

JavaScript has been a popular language from past to present for its advantages like flexibility and object-oriented nature. For these reasons, its domain has been extremely wide in years, and this required many people to use it. As a result of these situations, JavaScript has big and more than one communities to help beginners out and developers to solve bugs and problems.

Here some of the JavaScript communities: DEV Community, CodeNewbie Community, JS.dev Community

9. Advantages and Disadvantages

A. Comparing advantages of Pascal and JavaScript

Pascal	JavaScript
✔ The syntax is very simple. ✔ The language is complete, and it can be used to solve almost any programming problem. ✔ The language is well-structured. ✔ Safe and reliable ✔ Very fast compiler. ✔ Allows making insertions in the program's code in assembler for low-level programming and optimization of the program. ✔ Very low system requirements ✔ Allows using different approaches in programming. ✔ The elements of the array can be numbered starting at least from zero, even from a thousand.	✔ Versatility, can be used for both front-end and back-end development. ✔ Supported by all major web browsers ✔ Has a vast ecosystem of libraries, frameworks, and tools, such as React. ✔ High performance ✔ Asynchronous capabilities are crucial for building responsive and non-blocking web applications. ✔ Community and support ✔ Often recommended for beginners due to its simple and accessible syntax. ✔ Used for web, mobile, and desktop applications, allowing developers to create applications that run on multiple platforms. ✔ Enables dynamic and interactive web pages.

B. Comparing disadvantages of Pascal and JavaScript

Pascal	JavaScript
✗ Pascal and most of its programming environments do not have automatic garbage collection and cleaning. ✗ Limited industry use ✗ Outdated in some contexts ✗ Lack of modern features ✗ Less community and resources ✗ Strong typing overhead ✗ Compatibility issues, such as converting and integrating legacy Pascal code into modern systems can be a challenge due to differences in language versions and dialects.	✗ JavaScript code can behave differently in various web browsers, leading to cross-browser compatibility. ✗ JavaScript code runs on the client-side, making it susceptible to security vulnerabilities if not properly secured. ✗ Callback Hell, managing asynchronous code with callbacks can lead to deeply nested and hard-to-read code structures. ✗ Lack of strong typing ✗ Quirks and inconsistencies ✗ Concurrency challenges

10. Conclusion

In this essay, two distinct programming paradigms are explored, represented by Pascal in the procedural paradigm and JavaScript in the object-oriented paradigm. These languages have unique characteristics that make them suitable for different types of applications and development approaches.

A. Summary of key points

Pascal is a structured and strongly typed language known for simplicity and reliability. It excels in applications where clarity and structured programming are essential. Pascal's procedural nature promotes modular code design, making it well-suited for scientific computing, educational software, and applications that prioritize safety and maintainability.

On the other hand, JavaScript is a versatile language that supports both object-oriented and functional programming paradigms. It is primarily used for web development, enabling dynamic and interactive front-end applications. JavaScript's object-oriented features allow code organization and reusing through the creation of objects and classes.

B. Offering insights into when to consider Pascal or JavaScript on specific project requirements

Pascal can be considered for projects where structured and reliable code is paramount. It is well-suited for scientific computing, educational software, and applications requiring strong typing and modular design. Pascal is also an excellent choice when safety and clarity are top priorities.

For JavaScript, it can be considered for web development, particularly for building interactive and dynamic front-end applications. JavaScript's object-oriented capabilities make it a natural fit for creating rich user interfaces and web-based applications. Additionally, JavaScript is an essential language for full-stack development when used with Node.js.

11. References

- [1][https://en.wikipedia.org/wiki/Programming_paradigm]
- [2][https://isaaccomputerscience.org]
- [3][https://hackr.io/blog/procedural-programming]
- [4][https://www.educba.com/object-oriented-programming-paradigm]
- [5][https://www.tutorialspoint.com/pascal]
- [6][https://www.devstringx.com/features-of-javascript]
- [7][https://www.w3schools.com/js/js_operators.asp]
- [8][https://www.geeksforgeeks.org/javascript-split-a-string-with-multiple-separators]
- [9][https://en.wikipedia.org/wiki/Pascal_(programming_language)]
- [10][https://www.codeavail.com/blog/what-is-pascal-programming-language-used-for/]
- [11][https://www.simplilearn.com/applications-of-javascript-article]
- [12][https://medium.com/@teamcode20233/understanding-the-learning-curve-of-javascript-by-lightly-7bc65dd07061]

- [https://www.tutorialspoint.com/javascript]
- [https://developer.mozilla.org]
- [https://www.codingninjas.com/studio/library/what-are-the-features-of-object-oriented-programming]
- [https://www.irietools.com/iriepascal/]
- [https://www.freepascal.org/]
- [https://chat.openai.com/]