

20CYS312 - Principles of Programming Languages

Exploring Programming Paradigms

Assignment-01

Presented by K Hitesh Manjunath

CB.EN.U4CYS21028

TIFAC-CORE in Cyber Security

Amrita Vishwa Vidyapeetham, Coimbatore Campus

Feb 2024



AMRITA
VISHWA VIDYAPEETHAM



- 1 Procedural
- 2 Procedural -Ada
- 3 Event-Driven
- 4 Event-Driven - Vue.js
- 5 Comparison and Discussions
- 6 Bibliography



Procedural programming is a paradigm centered on creating procedures or functions that execute a series of computational steps. It is known for its simplicity and ease of implementation, particularly for beginners. It is widely used in various domains of software development, including critical systems like aviation, healthcare, and transportation.

Key features of procedural programming are:

- It uses a linear top-down approach, where the program is divided into procedures or routines that contain a series of steps to be carried out.
- The data and procedures are treated as separate entities, with data being used as input and output for the procedures.
- It revolves around the concept of procedure calls, where functions or procedures are invoked from other parts of the code.
- Languages often include pre-defined functions, which are standardized instructions included in the programming language itself or in standard libraries.



Ada is a statically typed, high-level programming language designed for reliability and safety in critical systems. It is heavily used in embedded real-time systems. Reasons to use Ada:

- Ease of learning
- Wide range of target processors supported.
- Rich and mature development environments and tool sets.
- It includes control structures like 'if,' 'case,' 'loop,' and 'while,' providing tools for structured flow control in procedural programming.
- Ensures that variables have well-defined types Ada allows the definition of procedures and functions, enabling developers to encapsulate sets of instructions into reusable and modular units.
- Ada's tasking model supports concurrent programming, allowing developers to create parallel and distributed systems.
- There is mostly no type reference



Key features that Support Procedural paradigm in Ada

- Ada supports DbC(Design By Contract), which is a programming technique that emphasizes the use of preconditions and postconditions to ensure that procedures and functions are used correctly. This helps catch errors at compile-time and improves the reliability of the code.
- Provides built-in constructs for defining and calling subroutines, allowing developers to break down complex tasks into smaller, more manageable pieces. Procedures and functions can be called multiple times within a program, allowing for code reuse and modularity.
- Ada's strong typing system helps catch errors at compile-time by ensuring that variables are used correctly and consistently throughout the program. This improves the reliability and maintainability of the code



Example

The code illustrates the use of a package `Math` containing a function to calculate the square of a number.

Code:-

```
with Ada.Text_IO; use Ada.Text_IO;
procedure Example is
package Math is
function Square(X: Integer) return Integer; end Math;
package body Math is
function Square(X: Integer) return Integer is
begin
return X * X;
end Square;
end Math;
A, B: Integer := 5;
begin
PutLine("The square of A is: " & Integer'Image(Math.Square(A)));
PutLine("The square of B is: " & Integer'Image(Math.Square(B)));
end Example;
```



```
The square of A is: 25  
The square of B is: 25
```

Figure: Output

Explanation: Package Math is defined, which contains a function to calculate the square of a number. The main procedure calls this function to demonstrate its reusability. The Math package can be used in other parts of the program without having to rewrite the same code, showcasing the modularity and reusability of the code in Ada. When executed, the output will display the squares of the numbers A and B.



Advantages

Some of the key advantages of using the procedural paradigm in Ada are:

- Ada's emphasis on modularity through procedures, functions, and packages allows for better organization and maintenance of the code, making it easier to understand and update over time
- Ada's strict syntax and strong typing contribute to improved code readability, reducing common programming errors and making the codebase more understandable and maintainable
- Ada's design emphasizes strong typing, explicit control structures, and support for both object-oriented and concurrent programming, making it a versatile tool in system and application software development, particularly in safety-critical and real-time systems such as avionics, air traffic control, railways, and defense systems
- A key aspect of Ada, is known for its efficiency and control, making it a staple in system programming and a suitable choice for projects that demand a high degree of precision, security, and safety



Event Driven

The **Event-Driven paradigm** is a software design pattern that enables systems to detect, process, manage, and react to real-time events as they occur. It promotes the production, detection, consumption, and reaction to events, allowing for the construction of flexible, scalable, and responsive systems.

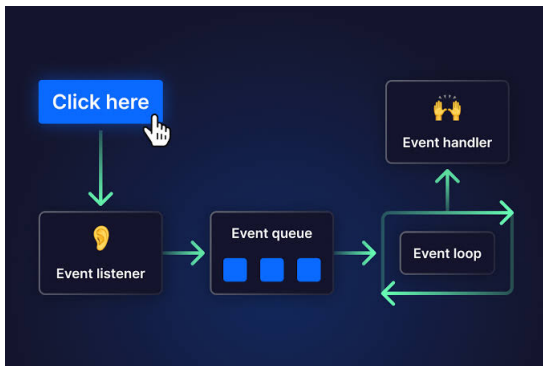


Figure: Event-Driven Architecture



- Modularity and Reusability are two key features of Event Driven Programming that make it simple to construct, maintain, and enhance applications. In this paradigm, event handlers and other components are separated, allowing developers to compose their applications using well-defined modules.
- The Event-Driven paradigm promotes loose coupling between components, allowing for the construction of flexible, scalable, and responsive systems.
- It can help organizations design and deploy robust systems that can expand and adjust to changing business requirements, thereby providing fault tolerance and reliability.



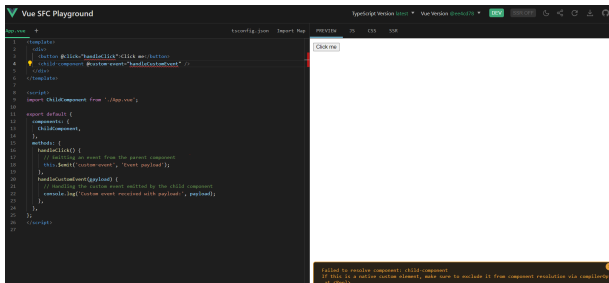
Event-Driven in Vue.js

Vue.js supports the Event-Driven paradigm through its event system, which allows components to communicate with each other by emitting and listening to events. Vue.js's event system is built upon the native event system implemented in browsers, typically using JavaScript. Vue.js adds several enhancements to this native system, including event modifiers, key modifiers, and mouse button modifiers, which make it easier to work with events and unify the browser implementation of the event system, thus eliminating the need to worry about differences between browsers.

When an event is triggered, Vue.js propagates it through the component hierarchy, enabling parent components to listen and respond to events emitted by child components. Vue.js's event system is a key feature that supports the Event-Driven paradigm, making it a popular choice for building modern, reactive, and scalable applications.



Example



```
Vue SFC Playground
typescript version latest • Vue Version @next/2 • NEW • PREVIEW • 25 • CSS • SASS

1 <template>
2   <div>
3     <button @click="handleClick">Click me</button>
4     <child-component @custom-event="handleCustomEvent" />
5   </div>
6 </template>
7
8 <script>
9   import ChildComponent from './App.vue';
10
11 export default {
12   components: {
13     ChildComponent,
14   },
15   methods: {
16     handleClick() {
17       // Emitting an event from the parent component
18       this.$emit('custom-event', 'Event payload');
19     },
20     handleCustomEvent(payload) {
21       // Handling the custom event emitted by the child component
22       console.log('Custom event received with payload:', payload);
23     },
24   },
25 };
26 </script>
27
```

Failed to resolve component: child-component
If this is a custom element, make sure to include it from component resolution via compiler at (0:0:0)

Figure: VueJS code

Explanation: The code demonstrates the event-driven nature of Vue.js, where the parent component emits an event that the child component listens for and responds to. This exemplifies the decoupled nature of events, allowing for modular and reusable component development without tight coupling.



Comparisons and discussions

Ada, a general-purpose programming language, primarily supports procedural programming, which involves organizing code into procedures or routines. On the other hand, Vue.js is a JavaScript framework that follows an event-driven programming paradigm, where actions or events trigger corresponding responses. Ada programs are structured around procedures and functions, emphasizing step-by-step execution.

It follows a linear control flow, with procedures being called sequentially. Ada relies on the concept of variables and data structures to manage program state.

Vue.js organizes code around components and relies on events to trigger actions. It follows a non-linear control flow, where components react to events asynchronously. Vue.js manages state using a reactive data model and employs a virtual DOM for efficient updates. Vue.js supports concurrency through its reactive model, allowing multiple components to update independently.



<https://thenewstack.io/the-basics-of-event-driven-architectures/>
<https://piembsystech.com/introduction-to-ada-programming-language/>
<https://dreamix.eu/insights/vue-js-why-event-driven-programming-is-amazing/>
<https://innstelios.blogspot.com/>

