Оглавление

[1. Gosu 1](#_Toc529960527)

[1.1. History 1](#_Toc529960528)

[1.2. Philosophy 2](#_Toc529960529)

[1.3. Syntax and semantics 2](#_Toc529960530)

[2. CoffeeScript 2](#_Toc529960531)

[2.1. History 2](#_Toc529960532)

[2.2. Philosophy 3](#_Toc529960533)

[2.3. Syntax and semantics 3](#_Toc529960534)

[3. Elm 4](#_Toc529960535)

[3.1. History 4](#_Toc529960536)

[3.2. Philosophy 4](#_Toc529960537)

[3.3. Syntax and semantics 4](#_Toc529960538)

[4. Список литературы 5](#_Toc529960539)

[5. Приложение 5](#_Toc529960540)

# Gosu

### History

«Gosu [Рисунок 1 - GosuLogo] is a statically-typed general-purpose programming language that runs on the Java Virtual Machine. Its influences include Java, C#, and ECMAScript. Development of Gosu began in 2002 internally for Guidewire Software, and the language saw its first community release in 2010 under the Apache 2 license.Gosu can serve as a scripting language, having free-form Program types (.gsp files) for scripting as well as statically verified Template files (.gst files). Gosu can optionally execute these and all other types directly from source without precompilation, which also distinguishes it from other static languages. Gosu began in 2002 as a scripting language called GScript at Guidewire Software. It was used to configure business logic in Guidewire's applications and was more of a simple rule definition language. In its original incarnation it followed ECMAScript guidelines. Guidewire enhanced the scripting language over the next 8 years, and released Gosu 0.7 beta to the community in November 2010. The 0.8 beta was released in December 2010, and 0.8.6 beta was released in mid-2011 with additional typeloaders, making Gosu capable of loading XML schema definition files and XML documents as native Gosu types. The latest version is 1.10, released in January 2016, along with a new IntelliJ IDEA editor plugin.» [4.1]

### Philosophy

«A unique feature of Gosu is its Open Type System, which allows the language to be easily extended to provide compile-time checking and IDE awareness of information that is typically checked [Таблица 1 - некоторые классы Gosu] only at runtime in most other languages. Enhancements let you add additional functions and properties to other types, including built-in Java types such as String, List, etc. This example demonstrates adding a print() function to java.lang.String. his general-purpose programming language is used in several open-source software projects including SparkGS and Ragnar DB among several others, and is widely used in the insurance industry via Guidewire Software's commercial products.» [4.1]

### Syntax and semantics

«Gosu follows a syntax resembling a combination of other languages. For instance, declarations follow more along the lines of Pascal with name-first grammar. Gosu classes can have functions, fields, properties, and inner classes as members. Nominal inheritance and composition via delegation are built into the type system as well as structural typing similar to the Go programming language. In addition to standard class types Gosu supports enums, interfaces, structures, and annotations. Program files facilitate Gosu as a scripting language. For example, Gosu's Hello, World! is a simple one-line program:

print("Hello, World!")

Gosu classes are also executable a la Java:

class Main {static function main(args: String[]) {print("Hello, World!")}}

» [4.1]

# CoffeeScript

### History

«CoffeeScript [Рисунок 2 - CofeeLogo] is a programming language that transcompiles to JavaScript. It adds syntactic sugar inspired by Ruby, Python and Haskell in an effort to enhance JavaScript's brevity and readability.Specific additional features include list comprehension and pattern matching. CoffeeScript support is included in Ruby on Rails version 3.1 and Play Framework.In 2011, Brendan Eich referenced CoffeeScript as an influence on his thoughts about the future of JavaScript.On December 13, 2009, Jeremy Ashkenas made the first Git commit of CoffeeScript with the comment: "initial commit of the mystery language."The compiler was written in Ruby. On December 24, he made the first tagged and documented release, 0.1.0On December 24, 2010, Ashkenas announced the release of stable 1.0.0 to Hacker News, the site where the project was announced for the first time.On September 18, 2017, version 2.0.0 was introduced, which "aims to bring CoffeeScript into the modern JavaScript era, closing gaps in compatibility with JavaScript while preserving the clean syntax that is CoffeeScript’s hallmark."» [4.2]

### Philosophy

«The CoffeeScript compiler has been written in CoffeeScript since version 0.5 and is available as a Node.js [Таблица 2 - Опции командной строки CoffeeScript] utility; however, the core compiler does not rely on Node.js and can be run in any JavaScript environment. One alternative to the Node.js utility is the Coffee Maven Plugin, a plugin for the popular Apache Maven build system. The plugin uses the Rhino JavaScript engine written in Java. The official site at CoffeeScript.org has a "Try CoffeeScript" button in the menu bar; clicking it opens a modal window in which users can enter CoffeeScript, see the JavaScript output, and run it directly in the browser. The js2coffee site provides bi-directional translation.» [4.2]

### Syntax and semantics

«Almost everything is an expression in CoffeeScript, for example if, switch and for expressions (which have no return value in JavaScript) return a value. As in Perl, these control statements also have postfix versions; for example, if can also be written after the conditional statement. Many unnecessary parentheses and braces can be omitted; for example, blocks of code can be denoted by indentation instead of braces, function calls are implicit, and object literals are often detected automatically.

var mass = 72;

var height = 1.78;

var BMI = mass / Math.pow(height, 2);

if (18.5 < BMI && BMI < 25) alert('You are healthy!');

»

[4.2]

# Elm

### History

«Elm [Рисунок 3 - ElmLogo] is a domain-specific programming language for declaratively creating web browser-based graphical user interfaces. Elm is purely functional, and is developed with emphasis on usability, performance, and robustness. It advertises "no runtime exceptions in practice", made possible by the Elm compiler's static type checking. Elm was initially designed by Evan Czaplicki as his thesis in 2012.The first release of Elm came with many examples and an online editor that made it easy to try out in a web browser.Evan Czaplicki joined Prezi in 2013 to work on Elm,and in 2016 moved to NoRedInk as an Open Source Engineer, also starting the Elm Software Foundation. The initial implementation of the Elm compiler targets HTML, CSS, and JavaScript.The set of core tools has continued to expand, now including a REPL, package manager, time-traveling debugger,and installers for Mac and Windows. Elm also has an ecosystem of community created libraries and an advanced online editor that allows saved work and inclusion of community libraries.» [4.3]

### Philosophy

«Elm has a small but expressive set of language constructs, including traditional if-expressions, let-expressions for local state, and case-expressions for pattern matching. As a functional language, it supports anonymous functions, functions as arguments, and partial application (currying) by default. Its semantics include immutable values, stateless functions, and static typing with type inference. Elm programs render HTML through a virtual DOM, and may interoperate with other code by using "JavaScript as a service". All values in Elm are immutable, meaning that a value cannot be modified after it is created. Elm uses persistent data structures to implement its Array, Dict, and Set libraries.» [4.3]

### Syntax and semantics

«Elm uses an abstraction called ports to communicate with JavaScript. It allows values to flow in and out of Elm programs, making it possible to communicate between Elm and JavaScript. Elm has a library called elm-html that a programmer can use to write HTML and CSS within Elm.It uses a virtual DOM approach to make updates efficient.

hypotenuse : Float -> Float -> Float

hypotenuse a b = sqrt (a^2 + b^2)

multiplyBy2 : number -> number

multiplyBy2 =(\*) 2

» [4.3]

# Список литературы

* 1. <https://en.wikipedia.org/wiki/Gosu_(programming_language)>
  2. <https://en.wikipedia.org/wiki/CoffeeScript>
  3. <https://en.wikipedia.org/wiki/Elm_(programming_language)>

# Приложение



Рисунок 1 - GosuLogo



Рисунок 2 - CofeeLogo

Рисунок 3 – ElmLogo

Таблица 1 - некоторые классы Gosu

|  |  |
| --- | --- |
| **Enhancement** | **Description** |
| allMatch( cond(elt1 : T):boolean ) : boolean | Returns true if all elements in this collection match the given condition and false otherwise |
| average( select:block(elt:T):java.lang.Number ) : BigDecimal | Return the average of the mapped value |
| concat( that : Collection<T> ) : Collection<T> | Return a new list that is the concatenation of the two lists |
| Count() : int | Return the number of elements in this Iterable object |
| countWhere( cond(elt:T):boolean ) : int | Return the count of elements in this collection that match the given condition |
| disjunction( that : Collection<T> ) : Set<T> | Returns a the set disjunction of this collection and the other collection, that is, all elements that are in one collection not and not the other |
| each( operation(elt : T) ) | This method will invoke the operation on each element in the Collection |

Таблица 2 - Опции командной строки CoffeeScript

|  |  |
| --- | --- |
| **Option** | **Description** |
| -c, --compile | Compile a .coffee script into a .js JavaScript file of the same name. |
| -t, --transpile | Pipe the CoffeeScript compiler’s output through Babel before saving or running the generated JavaScript. Requires @babel/core to be installed, and options to pass to Babel in a .babelrc file or a package.json with a babel key in the path of the file or folder to be compiled. See [Transpilation](https://coffeescript.org/#transpilation). |
| -m, --map | Generate source maps alongside the compiled JavaScript files. Adds sourceMappingURL directives to the JavaScript as well. |
| -M, --inline-map | Just like --map, but include the source map directly in the compiled JavaScript files, rather than in a separate file. |
| -i, --interactive | Launch an interactive CoffeeScript session to try short snippets. Identical to calling coffee with no arguments. |

# Опрос