Fusion

Manual de referência da linguagem de programação Fusion para

JVM, Web, Arduino e Código nativo de máquina.

Compila para código: Java, Swift, Arduino e TypeScript.

Desenvolvido e escrito por Gabriel Margarido

Nightly Builds: 183.1 (JVM) 183.2 (Swift) e 183.3 (TypeScript)
183.4 (Arduino)

Manual de referência da linguagem de programação Fusion.

Gabriel Margarido - Nightly 183 - Novo compilador

Propósito:

A ideia principal da criação de Fusion é compilar arquivos de código fonte em paradigma procedural/imperativo para código Java Orientado a Objetos e inteligível pela JVM (Java Virtual Machine), código de máquina nativo (Swift 5.7) e TypeScript (Web). Além de simplificar a sintaxe da linguagem, permitindo que qualquer um aprenda Fusion de forma simples, rápida e efetiva.

Sua sintaxe se assemelha com as linguagens: Swift, Lua, Julia, Ruby, Fortran, Javascript e TypeScript. Assim facilitando o aprendizado de Fusion para os mais familiarizados com estas linguagens de programação.

Características:

- Os blocos de código são delimitados por “do”/”end” ao invés de chaves.
- Compila para bytecode Java.
- Compila para código nativo Apple Swift.
- Compila para código TypeScript.
- Permite herança múltipla.
- Todos os métodos são públicos.
- Permite a criação de pseudoclasses/namespaces.
- Compatível com código Java pré-existente.
- Compatível com código-objeto (.o) pré-existente (LLVM)
- Permite a instanciação de objetos Java.
- Formato próprio de pacotes para distribuição Java (Fuse).

Fusion Standard Edition - Java:

- Alta performance
- Multi-thread
- Possui um compilador JIT dentro da Java Virtual Machine (JVM) acelerando a execução dos bytecodes.
- Sistema de pacote principal para interação de diferentes arquivos de código-fonte.
- A função principal é a **main**, os argumentos da linha de comando são acessados a partir do vetor **args**.
- O arquivo principal dentro do pacote é o **Main.class** ou **Main.fusion**
- Suporte a uma vasta biblioteca de código Java nativo, bastando apenas importar e chamar as funções Java dentro do código Fusion.
- Fortemente e estaticamente tipada
- Tudo é um objeto, mesmo não suportando classes e herança.
- Gerencia memória automaticamente - Coletor de lixo.
- Compilada para código Java, e após isso, para bytecode JVM.
- Distribuído e OpenSource/Software Livre sob licença BSD de 2 cláusulas.
- Executa independentemente do Sistema Operacional sem alterações no código-fonte
- Permite distribuição de código em formato fechado (Fuse) ou (class).
- Executa programas (bytecode JVM) com o mesmo resultado independente da plataforma (multiplataforma).
- Não gera código executável para o Sistema Operacional, ao invés disso se utiliza de bytecodes (um formato intermediário de código para execução na Máquina Virtual Java - JVM).

Fusion

Manual de referência da linguagem de programação Fusion.

Gabriel Margarido - Nightly 183 - Novo compilador

Fusion Web Edition - TypeScript:

- Tipagem estática e forte.
- Suporte ao ESModules
- Não suporta CommonJS
- Exportar funções e variáveis para serem utilizados posteriormente.
- Suporte a execução em Node.js (V8 engine - JIT) e na Web.
- Linguagem interpretada com compilador JIT.

Fusion Native Edition - Swift:

- Linguagem rápida e eficiente
- Compatível com código Objective-C existente.
- Permite desenvolvimento para sistemas padrão UNIX: macOS, iPadOS, iOS e GNU/Linux (SwiftOnLinux).
- Seguro e confiável
- 2.6 vezes mais rápido que Objective-C.
- 8.4 vezes mais rápido que Python 2.7
- Compilado para código nativo de máquina com LLVM.

Fusion Micro Edition - Arduino:

- Linguagem rápida e eficiente
- Compatível com código Arduino-C++ existente.
- Permite desenvolvimento para sistemas embarcados:

Arduino Uno, Arduino Mega, Arduino Nano, Arduino Micro, etc...

- Seguro e confiável
- Compilado para código Arduino-C++.

Atualizações 181.x -> 182.x

Chamada de funções - Não se utiliza mais a palavra reservada **call** para realizar a chamada de funções, agora apenas basta chamar a função pelo seu identificador/nome. Porém se mantém o açúcar sintático para retorno de valores de funções para variáveis.

Fusion

Manual de referência da linguagem de programação Fusion.
Gabriel Margarido - Nightly 183 - Novo compilador

Criar novo projeto Fusion:

O utilitário FusionNew facilita a criação de projetos Fusion em todas as edições através templates prontos armazenados built-in no utilitário **fusion-new**.

Novo projeto JVM:

```
fusion-new --jvm Main.fusion
```

Novo projeto Swift:

```
fusion-new --native Main.fusion
```

Novo projeto TypeScript/Node.js:

```
fusion-new --web Main.fusion
```

Novo projeto Arduino:

```
fusion-new --arduino Main Main.fusion
```

Fusion

Manual de referência da linguagem de programação Fusion.
Gabriel Margarido - Nightly 183 - Novo compilador

Programa básico em Fusion:

*Arquivo: HelloWorld.fusion

```
package main
@import fusion.std
namespace HelloWorld
   function main(args) : void
      println “Hello world”
   end
endnamespace
```

Importar bibliotecas Java:

```
@import java.util.Scanner
```

Importar bibliotecas Swift:

```
@import Foundation
```

Importar bibliotecas Node.js:

```
@import fs
@import ./meuarquivo
```

Importar bibliotecas Arduino:

```
@import Servo
@import ./MinhaBiblioteca
```

Comentários:

```
@com “Esse eh um comentario”
@com “Este eh outro comentario”
```

Fusion

Manual de referência da linguagem de programação Fusion.
Gabriel Margarido - Nightly 183 - Novo compilador

Compilar pacotes Fusion - Standard Edition:

A linguagem de programação Fusion introduziu em sua versão Nightly Build 130 um formato de pacotes Fusion similar ao JAR encontrado na linguagem Java, porém de mais simples empacotamento evitando o clássico erro dos pacotes JAR: “Nenhum atributo de manifesto principal em <pacote>.jar”.

O compilador FusionC irá compilar o código-fonte para bytecode JVM e em seguida o utilitário Fusion irá executar o pacote compilado, que nada mais é do que um arquivo ZIP organizado compatível com a maioria dos sistemas operacionais UNIX e Microsoft Windows. É apenas necessário que o sistema operacional UNIX no qual o compilador FusionC está sendo executado possua suporte aos utilitários (zip), (unzip) e (rm) além da Java Virtual Machine (JVM 8+).

```
fusion -c HelloWorld.fusion
fusion --run HelloWorld.fuse --ar=main
```

@import fusion.std

Metabiblioteca “fusion.std” - Standard Edition:

Incluir a metabilioteca **fusion.std** é a mesma coisa que importar cada uma das bibliotecas abaixo individualmente.

```
@import java.util.Scanner
@import java.util.ArrayList
@import java.util.Random
@import java.nio.file.Files
@import java.nio.file.Path
@import java.nio.file.Paths
@import java.io.IOException
@import java.io.File
@import java.io.FileWriter
@import javax.swing.*
@import java.awt.event.*
@import java.awt.*
```

Metabiblioteca “fusion.std” - Native Edition:

Incluir a metabilioteca **fusion.std** é a mesma coisa que importar cada uma das bibliotecas abaixo individualmente.

```
@import Foundation
```

Fusion

Manual de referência da linguagem de programação Fusion.
Gabriel Margarido - Nightly 183 - Novo compilador

Tipo de dados Variável	Tipo de dados Vetor	Tipo de dados Tabela	Caraterística
String	**String[]**	**String{}**	Cadeia de caracteres
int	**int[]**	**Int{}**	Números inteiros
float	**float[]**	**Float{}**	Números reais
bool	**bool[]**	**Bool{}**	Verdadeiro ou falso
void	*	*	Vazio - Sem retorno
expression	*	*	Expressão matemática

Inicialização e declaração de variáveis:

```
String nome = “Gabriel Margarido”
int idade = 16
float salario = 750.89
bool empregado = true
expression e = “12*(45+4)/4”
```

Declaração de variáveis:

```
String nome = undefined
int idade = undefined
float salario = undefined
bool empregado = undefined
```

Reatribuição de valores:

```
nome := “Paulo Cerqueira”
idade := 18
salario := 1204.75
empregado := false
```

*Operadores lógicos: is (==), not (!), isnot (!=), and (&&), or (||)

Fusion

Manual de referência da linguagem de programação Fusion.
Gabriel Margarido - Nightly 183 - Novo compilador

Condicionais:

```
if (a is 7) do
      ...

elseif (a > 8) do
      ...
else
      ...
end
```

Repetição:

```
while (a < 7)
   ...
end
```

Iteração I - Standard Edition/Native Edition/Web Edition/Micro Edition:

Para (i) de 0 até 5 faça....

```
for i in 0..5
      ...
end
```

Iteração I.II - Web Edition/Native Edition/Micro Edition:

Para (i) de 0 até 5 faça....

```
for i in x
      ...
end
```

Fusion

Manual de referência da linguagem de programação Fusion.
Gabriel Margarido - Nightly 183 - Novo compilador

Iteração II - Standard Edition/Native Edition/Web Edition/Micro Edition:

Faça 5 vezes...

```
5 times
   ...
end
```

Saída padrão (Macros) - Standard Edition/Web Edition/Native Edition/Micro Edition:

```
print “Hello world”
println “Hello world”

int a = 33
print a
println a
```

Entrada padrão (Macro) - Standard Edition/Native edition:

```
Scanner : String entrada
Scanner : <tipo> <variavel>
```

*Listas e vetores sempre armazenam a primeiro elemento na posição 0 (zero).

Sendo assim o índice 1 (um) é onde o segundo elemento da lista é armazenado.

Vetor:

Um vetor ou variável composta é uma “variável” que é capaz de armazenar vários valores ao mesmo tempo, separados e identificados por índices. Um vetor em Fusion é imutável, ao contrário das tabelas, como pode-se observar no próximo item.

```
String[] alunos = (“Gabriel”, “Joana”)
```

Tabela:

Uma tabela é basicamente um vetor mutável, possuindo métodos para sua manipulação. São sempre declarados com um par de chaves, tanto no tipo quanto do valor correspondente a sua inicialização.

```
String{} alunos = ()
```

Obter valor de indice - Standard Edition/Native Edition/Web Edition/Micro Edition:

```
int a = vetor[0]
int b = vetor[1]
```

Fusion

Manual de referência da linguagem de programação Fusion.
Gabriel Margarido - Nightly 183 - Novo compilador

Adicionar elementos - Standard Edition:

```
AppendString(lista,elemento)
AppendInt(lista,elemento)
AppendFloat(lista,elemento)
AppendBool(lista,elemento)
```

```
AppendString(alunos,“Gabriel”)
AppendString(alunos,“Joana”)
```

Remover elementos - Standard Edition:

```
RemoveString(lista,elemento)
RemoveInt(lista,elemento)
RemoveFloat(lista,elemento)
RemoveBool(lista,elemento)
```

```
RemoveString(alunos,“Gabriel”)
RemoveString(alunos,“Joana”)
```

Obter tamanho da lista - Standard Edition:

```
TableLenString(lista) : int <variavel>
TableLenInt(lista)  : int <variavel>
TableLenFloat(lista)    : int <variavel>
TableLenBool(lista)     : int <variavel>
```

```
TableLenString(alunos) : int tamanho
```

Obter elemento pelo índice da lista - Standard Edition:

```
TableGetString(lista, indice)   : String <variavel>
TableGetInt(lista, indice)      : int <variavel>
TableGetFloat(lista, indice)    : float <variavel>
```

```
TableGetString(alunos,1) : String elemento
```

Adicionar elementos (Macro) - Native Edition:

```
lista := Append(elemento)
lista := Append(“Hello world”)
```

Fusion

Manual de referência da linguagem de programação Fusion.
Gabriel Margarido - Nightly 183 - Novo compilador

Remover elementos - Native Edition:

```
lista := Remove(elemento)
lista := Remove(“Hello world”)
lista := RemoveAt(indice)
lista := RemoveFirst
lista := RemoveLast
lista := RemoveAll
```

Fatiar vetor - Native Edition:

```
lista := Slice(inicio, fim) => variavelRetorno
```

Funções com retorno - Standard Edition/Micro Edition:

É possível realizar chamadas para retorno de funções nativas dos compiladores dentro de código Fusion.

```
function minhaFuncaoDeCalculo(int::a, int::b) : int
   expression c = “a+b”
   return c
end

minhaFuncaoDeCalculo(3,7) : int resultado
int resultado = minhaFuncaoDeCalculo(3,7)
```

Fusion

Manual de referência da linguagem de programação Fusion.
Gabriel Margarido - Nightly 183 - Novo compilador

Funções sem retorno - Standard Edition/Micro Edition:

É possível realizar chamadas para retorno de funções nativas dos compiladores dentro de código Fusion. É permitido realizar chamada das funções **System.out.println** por exemplo em Java e **Serial.begin** em Arduino-C++.

```
function minhaFuncaoDeCalculo(int::a, int::b) : void
   expression c = “a+b”
   println c
end

minhaFuncaoDeCalculo(3,7)
```

Funções com retorno - Web Edition/Native Edition:

É possível realizar chamadas para retorno de funções nativas dos compiladores dentro de código Fusion.

```
function minhaFuncaoDeCalculo(a::int, b::int) : int
   expression c = “a+b”
   return c
end

minhaFuncaoDeCalculo(3,7) : int resultado
int resultado = minhaFuncaoDeCalculo(3,7)
```

Fusion

Manual de referência da linguagem de programação Fusion.
Gabriel Margarido - Nightly 183 - Novo compilador

Funções sem retorno - Web Edition/Native Edition:

É possível realizar chamadas para retorno de funções nativas dos compiladores dentro de código Fusion. É permitido realizar chamada das funções **console.log** por exemplo em TypeScript e **readLine** em Swift.

```
function minhaFuncaoDeCalculo(a::int, b::int) : void
   expression c = “a+b”
   println c
end

minhaFuncaoDeCalculo(3,7)
```

Funções Override (POO) com retorno - Standard Edition:

Quando utilizamos classes Java, muitas vezes precisamos reescrever os métodos.

```
override function minhaFuncaoDeCalculo(int::a, int::b) : int
   expression c = “a+b”
   return c
end

minhaFuncaoDeCalculo(3,7) : int resultado
int resultado = minhaFuncaoDeCalculo(3,7)
```

Funções Override (POO) sem retorno - Standard Edition:

Quando utilizamos classes Java, muitas vezes precisamos reescrever os métodos.

```
override function minhaFuncaoDeCalculo(int::a, int::b) : void
   expression c = “a+b”
   System.out.println(c)
end

minhaFuncaoDeCalculo(3,7)
```

Fusion

Manual de referência da linguagem de programação Fusion.
Gabriel Margarido - Nightly 183 - Novo compilador

Funções dinâmicas (POO) com retorno - Standard Edition:

Quando utilizamos classes Java, muitas vezes precisamos criar métodos dinâmicos, isso é, só poderão ser utilizados quando criarmos um objeto ou herança a partir do nome da classe onde declaramos a função dinâmica ou "não-estática". Veja abaixo como criar uma função dinâmica.

```
dynamic function  minhaFuncaoDeCalculo(int::a, int::b) : int
   expression c = "a+b"
   return c
end

/* "A partir de um objeto" */
@new objeto meuObjeto : MinhaClasse
objeto.minhaFuncaoDeCalculo(3,7) : int resultado
int resultado = objeto.minhaFuncaoDeCalculo(3,7)


/* "A partir de herança" */
namespace HelloWorld < MinhaClasse
            ...
   minhaFuncaoDeCalculo(3,7) : int resultado
   int resultado = minhaFuncaoDeCalculo(3,7)
            ...
endnamespace
```

Funções dinâmicas (POO) sem retorno - Standard Edition:

Quando utilizamos classes Java, muitas vezes precisamos criar métodos dinâmicos, isso é, só poderão ser utilizados quando criarmos um objeto ou herança a partir do nome da classe onde declaramos a função dinâmica ou "não-estática". Veja abaixo como criar uma função dinâmica.

```
dynamic function minhaFuncaoDeCalculo(int::a, int::b) : void
   expression c = "a+b"
   System.out.println(c)
end

minhaFuncaoDeCalculo(3,7)
```

Manual de referência da linguagem de programação Fusion.

Gabriel Margarido - Nightly 183 - Novo compilador

```
/* “A partir de um objeto” */
@new objeto meuObjeto : MinhaClasse
objeto.minhaFuncaoDeCalculo(3,7)


/* “A partir de herança” */
namespace HelloWorld < MinhaClasse
            ...
   minhaFuncaoDeCalculo(3,7)

            ...
endnamespace
```

Escrever/Sobrescrever arquivos de texto - Standard Edition:

```
String arquivo = “teste.txt”
String msg = “Hello world”
WriteFile(arquivo,msg)
```

Adicionar ao arquivo de texto - Standard Edition:

```
String arquivo = “teste.txt”
String msg = “Hello world”
AppendFile(arquivo,msg)
```

Ler arquivos de texto - Standard Edition:

```
String arquivo = “teste.txt”
ReadFile(arquivo) : String msg
```

Fusion

Manual de referência da linguagem de programação Fusion.
Gabriel Margarido - Nightly 183 - Novo compilador

Gerar número inteiro aleatório - Standard Edition:

```
RandomIntValue(152) : int randomico
```

Calcular média aritmética - Standard Edition:

```
MathAverage(3+7+4,3) : int media
```

Conversão de tipos - TypeCasting : (Web Edition/Native Edition/Standard Edition):

```
float num = 15.6786
@cast num : int casted_num

println casted_num
```

Estrutura de conversão:

```
@cast <origem> : <tipo> <destino>
```

Fusion

Manual de referência da linguagem de programação Fusion.
Gabriel Margarido - Nightly 183 - Novo compilador

Tratamento de exceções - Standard Edition:

Assim como em Java, é possível e até mais fácil tratar exceções em Fusion, analise a estrutura deste pedaço de código abaixo:

I. Forma usual

```
try
   @com “Tente executar”
   ...

catch (Throwable error) do
   @com “Capture o erro e trate-o”
   ...
end
```

II. Forma extensa

```
try
   @com “Tente executar”
   ...

catch (Throwable error) do
   @com “Capture o erro e trate-o”
   ...

finally
   @com “Finalmente termine”
   ...
end
```

Fusion

Manual de referência da linguagem de programação Fusion.
Gabriel Margarido - Nightly 183 - Novo compilador

Tratamento de exceções - Web Edition:

É possível tratar exceções em Fusion, analise a estrutura deste pedaço de código abaixo:

I. Forma usual

```
try
   @com “Tente executar”
   ...

catch (Throwable error) do
   @com “Capture o erro e trate-o”
   ...
end
```

Funcionalidades web básicas DOM (Document Object Model) - Web Edition:

I. Para obter valores de elementos HTML nas páginas Web com o DOM, o tipo a ser passado para a variável é **HTMLElement.**

```
HTMLElement elemento = window.document.getElementById(“caixa”)

HTMLElement elemento =
window.document.getElementsByTagName(“p”)

HTMLElement elemento = window.alert(“Hello world”)
HTMLElement elemento = window.location.hostname
```

Fusion

Manual de referência da linguagem de programação Fusion.
Gabriel Margarido - Nightly 183 - Novo compilador

Pacotes - Standard Edition:

Pacotes são essencialmente uma árvore de diretórios que armazena bytecodes Java de forma organizada atribuindo-a identificadores, para mais tarde empacotar esta árvore em forma de um arquivo Fusion distribuível (fuse).

```
package <nomedopacote>
package main
```

Empacotamento - Standard Edition:

O compilador FusionC nos provê a possibilidade de empacotar os programas escritos em Fusion em um formato fechado distribuível, para compilar e empacotar os pacotes Fusion utilize os seguintes comandos:

*Lembre-se o nome do pacote estará sempre no padrão *snakecase*, isso é, todas as letras estarão em sua forma minúscula e sem espaços.

II. Para compilar um único arquivo de código-fonte para bytecode JVM e empacotar em FUSE.

```
fusion -c Main.fusion
fusion -l --ar=meuprimeiroprograma
```

III. Para compilar vários arquivos de código-fonte para bytecode JVM e empacotar em FUSE.

```
fusion -o Main.fusion
fusion -j Main.java
fusion -l --ar=meuprimeiroprograma
```

Exemplos práticos:

Para facilitar o entendimento, abaixo estão exemplos de chamadas de funções em diferentes arquivos, a compilação dos fontes e empacotamento dos bytecodes em um único pacote Fuse.

*Arquivo: Main.fusion

```
package meuprimeiroprograma
@import fusion.std
namespace Main
   function main(args) : void
      println “Hello world”
      call imprimir()
   end
endnamespace
```

Fusion

Manual de referência da linguagem de programação Fusion.
Gabriel Margarido - Nightly 183 - Novo compilador

*Arquivo: Imprimir.fusion

```
package meuprimeiroprograma
@import fusion.std

namespace Imprimir
   function imprimir() : void
      println “Esta impresso”
   end
endnamespace
```

I. Compilar vários arquivos de código-fonte Fusion em um único pacote FUSE

Opções:

fusion <opções> <parâmetros>

-c : Compilar de Fusion para bytecode JVM.

-j : Compilar de Java para bytecode JVM.

-o : Transpilar de Fusion para Java.

-l : Gerar um pacote FUSE a partir de bytecodes JVM.

*Compilar todos os arquivos de código-fonte em um único pacote:

```
fusion –o Main.fusion
fusion –j Main.java

fusion –o Imprimir.fusion
fusion –j Imprimir.java
fusion –l ––ar=meuprimeiroprograma
```

*Executar o pacote Fuse:

```
fusion ––run meuprimeiroprograma.fuse –ar=meuprimeiroprograma
```

II. Compilar um único arquivo de código-fonte em um único pacote FUSE

Opções:

fusion <opções> <parâmetros>

-c : Compilar de Fusion para bytecode JVM.

-j : Compilar de Java para bytecode JVM.

-o : Transpilar de Fusion para Java.

-l : Gerar um pacote FUSE a partir de bytecodes JVM.

--run : Executar pacote FUSE

*Compilar somento o arquivo de código-fonte “Main.fusion” em um único pacote:

```
fusion –c Main.fusion
fusion –l ––ar=meuprimeiroprograma
```

*Executar o pacote Fuse:

```
fusion ––run meuprimeiroprograma.fuse –ar=meuprimeiroprograma
```

II. Compilar vários arquivos de código-fonte *Fusion e Java* em um único pacote FUSE

Todos os arquivos de código-fonte devem pertencer ao mesmo pacote, isso é, se o nome do pacote for “meuprimeiroprograma” os arquivos de código Fusion devem conter:

```
package meuprimeiroprograma
@import fusion.std

namespace Programa
    ...
endnamespace
```

E os arquivos de código-fonte Java, o comando equivalente com ponto-e-vírgula no final dos comandos

```
package meuprimeiroprograma;
public class ClasseJava {
    ...
}
```

Opções:

fusion <opções> <parâmetros>

-c : Compilar de Fusion para bytecode JVM.

-j : Compilar de Java para bytecode JVM.

-o : Transpilar de Fusion para Java.

-l : Gerar um pacote FUSE a partir de bytecodes JVM.

--run : Executar pacote FUSE

Arquivos:

- Main.fusion

- Imprimir.fusion

- Calcular.java

- Deduzir.java

*Compilar todos os arquivos de código-fonte em um único pacote:

```
fusion -o Main.fusion
fusion -j Main.java

fusion -o Imprimir.fusion
fusion -j Imprimir.java
fusion -l --ar=meuprimeiroprograma
```

*Executar o pacote Fuse:

```
fusion --run meuprimeiroprograma.fuse -ar=meuprimeiroprograma
```

*O pacote Fuse gerado não poderá ser renomeado, caso contrário apresentará inconsistências.

Fusion

Manual de referência da linguagem de programação Fusion.
Gabriel Margarido - Nightly 183 - Novo compilador

Instanciar objetos - (Standard Edition/Web Edition/Native Edition):

Para permitir compatibilidade entre Fusion e as bibliotecas escritas em Java, existe a possibilidade de instanciar objetos Java dentro de código Fusion. Lembre-se a linguagem Fusion não nos permite criar classes diretamente, mas nos permite acessar as classes já existentes em Java através de objetos.

```
@new <objeto> : <classe>
@new janela : JFrame(“Meu primeiro programa”)
```

Criar pseudoclasses - (Standard Edition):

Em Fusion não existe uma palavra reservada para criação de classes, como “class” em Java. Mas internamente, ao declararmos um novo programa “namespace”, o compilador FusionC cria uma classe Java por de trás dos panos. Vale ressaltar que o arquivo de código-fonte deve possuir o mesmo nome do namespace (classe) do programa, caso contrário, o compilador JavaC retornará um erro de compilação.

```
namespace Programa
    ...
endnamespace
```

*Arquivo: Programa.fusion

Compilação - Native Edition:

O compilador FusionC nos provê a possibilidade de compilar os programas escritos em Fusion Native em um formato fechado distribuível (executável), para compilar utilize os seguintes comandos:

IV. Para compilar um único arquivo de código-fonte para binário.

```
fusion-swift -o Main.fusion
./Main
```

V. Para compilar vários arquivos de código-fonte para um único binário.

```
fusion-swift -c Main.fusion
fusion-swift -c Arquivo1.fusion
fusion-swift -c Arquivo2.fusion
swiftc Main.swift Arquivo1.swift Arquivo2.swift -o Main

./Main
```

Fusion

Manual de referência da linguagem de programação Fusion.
Gabriel Margarido - Nightly 183 - Novo compilador

Compilação - Web Edition:

O compilador FusionC nos provê a possibilidade de compilar os programas escritos em Fusion Web em um formato aberto distribuível (Javascript), para compilar utilize os seguintes comandos:

VI. Para configurar o ambiente local e compilar um único arquivo de código-fonte para Javascript.

```
fusion-ts -o Main.fusion
./Main
```

VII. Para transpilar um único arquivo de código-fonte para TypeScript.

```
fusion-ts -c Main.fusion
```

VIII. Configurar ambiente local TypeScript.

```
fusion-ts -e
```

Compilação - Micro Edition:

O compilador FusionC nos provê a possibilidade de compilar os programas escritos em Fusion Micro em um formato aberto distribuível para Arduino (Arduino-C++), para transpilar para Arduino-C++ e gerar um pacote Arduino utilize os seguintes comandos:

```
fusion-arduino -o <Pacote> <Sketch>.fusion
fusion-arduino -o Main Main.fusion
```

*Ambos os argumentos devem começar com a primeira letra maiúscula.

Após isso, será necessário abrir o arquivo Main.ino gerado e fazer o upload para a placa microcontroladora através da IDE - Arduino IDE 2.x
disponível em https://www.arduino.cc/en/software

Fusion

Manual de referência da linguagem de programação Fusion.
Gabriel Margarido - Nightly 183 - Novo compilador

Corrente elétrica - Ligando e desligando circuitos (Macro)

- Definindo a porta 13 como saída

```
SetOutput 13
```

- Definindo a porta 13 como entrada

```
SetInput 13
```

- Ligando a porta 13

```
Enable 13
```

- Desligando a porta 12

```
Disable 12
```

- Esperando 1000ms (1 segundo)

```
wait 1000
```

- Inicia comunicação serial com 9600bit/seg

```
SerialCom 9600
```

- Exemplo de código - Piscar led na porta 7

  Arquivo: Main.fusion => Main/Main.ino

```
package main
@import fusion.std

namespace Main
   function setup() : void
      SetOutput 7
   end

   function loop() : void
      Enable 7
      wait 500
      Disable 7
   end

endnamespace
```

- O pino 7 está ligado? (true) or (false)

```
bool x = digitalRead(7)
```

- Exemplo de código - Liga e desliga led com botão

```
package main
@import fusion.std

namespace Main
   int pin = 7
   int led = 13
   bool x = digitalRead(pin)

   function setup() : void
   end

   function loop() : void
      if (x is ON) do
         Disable led
      elseif (x is OFF)
         Enable led
      end
   end
endnamespace
```

- Utilizar funções de bibliotecas de terceiros com Arduino através de variáveis.

Maneira I:

```
bool x = digitalRead(7)
```

Maneira II:

```
digitalRead(7) : bool x
```

Fusion

Manual de referência da linguagem de programação Fusion.
Gabriel Margarido - Nightly 183 - Novo compilador

Exemplo - Maneira I:

```
<tipo> <variavelRetorno> = funcao(param1, param2, param3, ...)
```

Exemplo - Maneira II:

```
funcao(param1, param2, param3, ...) : <tipo> <variavelRetorno>
```

Importar bibliotecas de terceiros localizadas dentro do mesmo pacote Arduino : (MinhaBiblioteca.h)

```
@import ./MinhaBiblioteca
```

Importar bibliotecas nativas do Arduino : (Servo.h)

```
@import Servo
```

Operadores lógico-literais da linguagem Fusion:

```
is (==)  isnot (!=)  not (!)  or (||)  and (&&)
```

Exemplos:

```
if (a is 7)        if (a not b)      if (a is 7 and b is 4)
if (a isnot 7)     if (a is 7 or b is 4)
```

Criar funções Java para Fusion SE (Standard Edition):

Todas as funções Java a serem chamadas devem ser públicas e preferencialemente estáticas, porém o compilador FusionC também permite a instanciação de objetos a partir de classes. Vale lembrar que o nome do pacote deve ser o mesmo dentro da declaração de pacote nos códigos-fonte de classes Java e nos namespaces do Fusion. É possível também chamar funções, variáveis e constantes Java dentro de código Fusion, como por exemplo, um simples: **System.out.println("Hello world")**
Porém em Fusion não se utiliza ponto-e-vírgula, diferentemente do Java.

Fusion

Manual de referência da linguagem de programação Fusion.
Gabriel Margarido - Nightly 183 - Novo compilador

1. Exemplo estático sem retorno (recomendado)

```
package <pacote>;
public class <Classe> {
   public static void <nome>(<args>) {

   }
}
```

Chamada em Fusion SE:

```
<nome>(<args>)
```

2. Exemplo estático com retorno de valor (recomendado)

```
package <pacote>;
public class <Classe> {
   public static <tipo> <nome>(<args>) {
                  ...
      return <valor>;
   }
}
```

Chamada em Fusion SE com retorno de valor:

```
<tipo> <nome_variavel> = <nome>(<args>)
```

3. Exemplo orientado a objetos sem retorno de valor

```
package <pacote>;
public class <Classe> {
   public void <nome>(<args>) {
            ...
   }
}
```

Chamada em Fusion SE:

```
@new <obj> : <Classe>()
<obj>.<nome>(<args>)
```

4. Exemplo orientado a objetos com retorno de valor

```
package <pacote>;
public class <Classe> {
   public <tipo> <nome>(<args>) {
                ...
      return <valor>
   }
}
```

Chamada em Fusion SE:

```
@new <obj> : <Classe>()
<tipo> <nome_variavel> = <obj>.<nome>(<args>)
```

Criar funções Swift para Fusion Native Edition (Swift):

Todas as funções Swift a serem chamadas devem ser públicas, porém o compilador FusionC também permite a instanciação de objetos a partir de classes. É possível também chamar funções, variáveis e constantes Swift dentro de código Fusion NE, como por exemplo, um simples:

String leitura = readLine()

*Vale lembrar que em Fusion não se utiliza ponto-e-vírgula, diferentemente do Swift que é opcional.

5. Exemplo estático sem retorno (recomendado)

```
func <nome>(<args>) {

}
```

Chamada em Fusion NE:

```
<nome>(<args>)
```

6. Exemplo estático com retorno (recomendado)

```
func <nome>(<args>) -> <Tipo> {

}
```

Chamada em Fusion NE:

```
<tipo> <nome_variavel> = <nome>(<args>)
```

7. Exemplo usando classes sem retorno (recomendado)

```
class <Classe> {
   func <nome>(<args>) {
         ...
   }
}
```

Chamada em Fusion NE:

```
@new <obj> : <Classe>()
<obj>(<args>)
```

8. Exemplo usando classes com retorno (recomendado)

```
class <Classe> {
   func <nome>(<args>) -> <Tipo> {
                ...
      return <valor>
   }
}
```

Chamada em Fusion NE:

```
@new <obj> : <Classe>()
<tipo> <nome_variavel> = <obj>(<args>)
```

Criar funções Arduino-C++ para Fusion Micro Edition (Arduino):

Todas as funções Arduino a serem chamadas devem ser públicas, porém o compilador FusionC também permite a instanciação de objetos a partir de classes. É possível também chamar funções, variáveis e constantes Swift dentro de código Fusion ME, como por exemplo, um simples:

digitalWrite(13, HIGH)

*Vale lembrar que em Fusion não se utiliza ponto-e-vírgula, diferentemente do Arduino-C++ que é obirgatório.

9. Exemplo estático sem retorno (recomendado)

```
void <nome>(<args>) {
      ...
}
```

Chamada em Fusion ME:

```
<nome>(<args>)
```

10. Exemplo estático com retorno (recomendado)

```
<tipo> <nome>(<args>) {
         ...
      return <valor>;
}
```

Chamada em Fusion ME:

```
<tipo> <nome_variavel> = <nome>(<args>)
```

Criar funções TypeScript/Javascript para Fusion Web Edition (TypeScript):

Todas as funções TypeScript/Javacript a serem chamadas devem ser públicas, porém o compilador FusionC também permite a instanciação de objetos a partir de classes. É possível também chamar funções, variáveis e constantes Javascript dentro de código Fusion ME, como por exemplo, um simples:

console.log(“Hello world”)

*Vale lembrar que em Fusion não se utiliza ponto-e-vírgula, diferentemente do Javascript que é obirgatório ou até opcional em versões mais recentes.