

OFFICIAL ENGINE REFERENCE

---

# JuJu

## Engine Reference Master

---

*The complete, locked-forever reference for JuJu Programming Language —  
all 60+ keywords, all 4 versions, all Auto-Fix Engine rules in one place.*

◆ v1.0 + v0.2 + v0.3 + v0.4 · Complete Edition

Created by **Sathish Kumar .J** · Tirupur, Tamil Nadu

---

**"WRITESIMPLE.BUILDREAL."**

# Contents

---

|    |                                                           |
|----|-----------------------------------------------------------|
| 01 | Program Structure & Core Rules                            |
| 02 | Data Types — All 7 Types                                  |
| 03 | Input & Output                                            |
| 04 | Comments & String Interpolation                           |
| 05 | Arithmetic Operators                                      |
| 06 | Comparison & Logical Operators                            |
| 07 | Assignment Operators                                      |
| 08 | Conditions — check, check or, or                          |
| 09 | Loops — count, do, do first                               |
| 10 | Functions — fun & back                                    |
| 11 | Collections — list & map                                  |
| 12 | v0.3 Advanced — back, null, len, pop, set, turn, ?:, pick |
| 13 | v0.4 Final 3 — safe/fail, hold, blueprint/build           |
| 14 | Auto-Fix Engine — All 20 Rules                            |
| 15 | Complete Keyword Master List                              |

## CHAPTER 01

# Program Structure

*The foundation of every JuJu program.*

Every JuJu program lives inside a **pro(){ }** block. Think of it as the front door — everything goes inside it.

### THE THREE GOLDEN RULES

| # | RULE                               | EXAMPLE         |
|---|------------------------------------|-----------------|
| 1 | Every program starts with pro(){ } | pro(){ ... }    |
| 2 | Every line ends with a dot .       | dis = "Hello" . |
| 3 | Use double quotes for text         | "Vanakkam"      |

### CORE STRUCTURE

## The pro( ) block

J U J U

```
pro(){  
    dis = "Vanakkam JuJu!" .  
}
```

D A R T (AUTO-GENERATED)

```
void main() {  
    print("Vanakkam JuJu!");  
}
```

### AUTO-FIX ENGINE

If you forget the dot at the end of a line, the JuJu Auto-Fix Engine will automatically add it for you and show a yellow warning panel.

### COMMENTS

## Notes to yourself — ignored by the transpiler

J U J U

```
/ This is my first program .  
pro(){  
    ** This comment  
    spans two lines **  
    dis = "Hello!" .  
}
```

D A R T

```
// This is my first program  
void main() {  
    /* This comment  
    spans two lines */  
    print("Hello!");  
}
```

■ LOCKED FOREVER — THESE KEYWORDS WILL NEVER CHANGE

pro · dis · output · / comment · \*\* comment \*\* · \$ interpolation · join

# Data Types

*Storing and remembering information — all 7 types.*

A variable is like a labelled box where you store information. You give it a name, and use that name later to get the value back.

## THE 7 VARIABLE TYPES

| JuJu Type               | Dart Type | Stores            | Example                               |
|-------------------------|-----------|-------------------|---------------------------------------|
| <code>txt / text</code> | String    | Words, sentences  | <code>txt city = "Tirupur" .</code>   |
| <code>num</code>        | int       | Whole numbers     | <code>num age = 21 .</code>           |
| <code>dec</code>        | double    | Decimal numbers   | <code>dec price = 99.50 .</code>      |
| <code>yon</code>        | bool      | true / false only | <code>yon isOpen = true .</code>      |
| <code>guess</code>      | var       | Anything (auto)   | <code>guess job = "Boss" .</code>     |
| <code>list</code>       | List      | Multiple items    | <code>list names = ["A","B"] .</code> |
| <code>map</code>        | Map       | Key-value pairs   | <code>map person = {} .</code>        |

## STRING INTERPOLATION

### Using variables inside text with \$

```
J U J U
pro(){
  txt name = "Machi" .
  num age = 25 .
  dis = "Name: $name" .
  dis = "Age: $age" .
}
```

```
D A R T
void main() {
  String name = "Machi";
  int age = 25;
  print("Name: $name");
  print("Age: $age");
}
```

#### ■ LOCKED FOREVER — THESE KEYWORDS WILL NEVER CHANGE

`txt / text · num · dec · yon · guess · list · map · null (v0.3)`

# Input & Output

*Talking to the user — reading what they type.*

| JuJu Code                           | Dart Code                                                              | What it does                          |
|-------------------------------------|------------------------------------------------------------------------|---------------------------------------|
| <code>input name .</code>           | <code>String name = stdin.readLineSync()!;</code>                      | Read text input from user             |
| <code>input txt name .</code>       | <code>String name = stdin.readLineSync()!;</code>                      | Explicit text input                   |
| <code>input num age .</code>        | <code>int age = int.parse(stdin.readLineSync() ?? "0");</code>         | Number input — auto parsed to int     |
| <code>input dec price .</code>      | <code>double price = double.parse(stdin.readLineSync() ?? "0");</code> | Decimal input — auto parsed to double |
| <code>dis = "Hello \$name" .</code> | <code>print("Hello \$name");</code>                                    | Print with \$ interpolation           |
| <code>output "Done \$name" .</code> | <code>print("Done \$name");</code>                                     | Output result with variable           |

J U J U

```
pro(){
  input name .
  input num age .
  dis = "Hi $name! Age: $age" .
}
```

D A R T

```
void main() {
  String name = stdin.readLineSync()!;
  int age = int.parse(stdin.readLineSync() ?? "0");
  print("Hi $name! Age: $age");
}
```

■ LOCKED FOREVER — THESE KEYWORDS WILL NEVER CHANGE

`input · dis · output`

## CHAPTER 04

# Operators

*Math, comparison, and logic — the tools of thinking.*

## ARITHMETIC OPERATORS

### Math — 5 operators

| JuJu               | Dart               | Meaning                           |
|--------------------|--------------------|-----------------------------------|
| <code>a + b</code> | <code>a + b</code> | Addition                          |
| <code>a - b</code> | <code>a - b</code> | Subtraction                       |
| <code>a x b</code> | <code>a * b</code> | Multiplication — x means *        |
| <code>a / b</code> | <code>a / b</code> | Division                          |
| <code>a % b</code> | <code>a % b</code> | Modulo — remainder after division |

## COMPARISON OPERATORS

### Comparing two values — 6 operators

| JuJu                 | Dart                   | Meaning                     |
|----------------------|------------------------|-----------------------------|
| <code>a is b</code>  | <code>a == b</code>    | Equal — same value check    |
| <code>a not b</code> | <code>a != b</code>    | Not equal — different value |
| <code>a gt b</code>  | <code>a &gt; b</code>  | Greater than                |
| <code>a lt b</code>  | <code>a &lt; b</code>  | Less than                   |
| <code>a gte b</code> | <code>a &gt;= b</code> | Greater than or equal       |
| <code>a lte b</code> | <code>a &lt;= b</code> | Less than or equal          |

## LOGICAL OPERATORS

### AND, OR, NOT — 3 operators

| JuJu                        | Dart                                | Meaning                                   |
|-----------------------------|-------------------------------------|-------------------------------------------|
| <code>cond1 tt cond2</code> | <code>cond1 &amp;&amp; cond2</code> | <code>tt</code> = AND — both must be true |
| <code>cond1 or cond2</code> | <code>cond1    cond2</code>         | <code>or</code> = OR — at least one true  |
| <code>opp condition</code>  | <code>!condition</code>             | <code>opp</code> = NOT — opposite/reverse |

## ASSIGNMENT OPERATORS

## Update variable values — 7 operators

| JuJu        | Dart    | Meaning                  |
|-------------|---------|--------------------------|
| x = 10 .    | x = 10; | Assign value to variable |
| x add 5 .   | x += 5; | Add and save — x = x + 5 |
| x sub 3 .   | x -= 3; | Subtract and save        |
| x mul 2 .   | x *= 2; | Multiply and save        |
| x divby 4 . | x /= 4; | Divide and save          |
| x up .      | x++;    | Increment by 1           |
| x dn .      | x--;    | Decrement by 1           |

■ LOCKED FOREVER — THESE KEYWORDS WILL NEVER CHANGE

+ - x / % . is not gt lt gte lte . tt or opp . = add sub mul divby up dn . join . ? :

# Conditions

*Making decisions in your program — check, check or, or.*

Conditions let your program make decisions. If something is true, do this. Otherwise, do that.

## THE 3 CONDITION KEYWORDS

| Keyword  | Dart Equivalent | When it runs                                  |
|----------|-----------------|-----------------------------------------------|
| check    | if              | First condition — runs when condition is true |
| check or | else if         | Second condition — another check              |
| or       | else            | Fallback — runs when everything else fails    |

J U J U

```
pro(){
  num score = 85 .
  check score gte 90 {
    dis = "A Grade" .
  }
  check or score gte 75 {
    dis = "B Grade" .
  }
  or {
    dis = "Try Again" .
  }
}
```

D A R T

```
void main() {
  int score = 85;
  if (score >= 90) {
    print("A Grade");
  } else if (score >= 75) {
    print("B Grade");
  } else {
    print("Try Again");
  }
}
```

■ LOCKED FOREVER — THESE KEYWORDS WILL NEVER CHANGE

check · check or · or

# Loops

*Repeating actions without repeating code — count & do.*

## COUNT - For Loop

**Use count when you know exactly how many times to repeat.**

```
J U J U
pro(){
    count i from 1 to 5 {
        dis = i .
    }
}
```

```
D A R T
void main() {
    for (int i = 1; i <= 5; i++) {
        print(i);
    }
}
```

## DO - While Loop

**Use do when you want to repeat while a condition is true.**

```
J U J U
pro(){
    num i = 1 .
    do i lte 5 {
        dis = i .
        i add 1 .
    }
}
```

```
D A R T
void main() {
    int i = 1;
    while (i <= 5) {
        print(i);
        i++;
    }
}
```

## DO FIRST - Do - While Loop

**Runs the code at least once, then checks the condition.**

```
J U J U
pro(){
    num i = 1 .
    do first i lte 3 {
        dis = i .
        i add 1 .
    }
}
```

```
D A R T
void main() {
    int i = 1;
    do {
        print(i);
        i++;
    } while (i <= 3);
}
```

| Keyword                 | Dart                      | Use when                       |
|-------------------------|---------------------------|--------------------------------|
| count i from 1 to N { } | for (int i=1; i<=N; i++)  | You know the exact count       |
| do condition { }        | while (condition) { }     | Repeat while condition is true |
| do first condition { }  | do { } while (condition); | Must run at least once         |

■ LOCKED FOREVER – THESE KEYWORDS WILL NEVER CHANGE

count · do · do first

# Functions

*Writing reusable code blocks — fun, back, and parameters.*

A function is a named block of code you can call multiple times. Instead of repeating the same code, write it once in a function.

J U J U

```
fun greet() {
  dis = "Vanakkam!" .
}
pro(){
  greet() .
  greet() .
}
```

D A R T

```
void greet() {
  print("Vanakkam!");
}
void main() {
  greet();
  greet();
}
```

F U N C T I O N S W I T H P A R A M E T E R S &amp; b a c k

**back sends a value back to where the function was called.**

J U J U

```
fun add(num a, num b) {
  back a + b .
}
pro(){
  dis = add(5, 3) .
}
```

D A R T

```
int add(int a, int b) {
  return a + b;
}
void main() {
  print(add(5, 3));
}
```

■ LOCKED FOREVER — THESE KEYWORDS WILL NEVER CHANGE

fun · back

# Lists & Maps

*Storing multiple values in one place.*

## LIST - Multiple Items

**An ordered collection of items — access by index starting from 0.**

```
J U J U
pro(){
    list names = ["Raj","Kumar","Machi"]
    .
    dis = names[0] .
    dis = names len .
    names pop .
}
```

```
D A R T
void main() {
    List names = ["Raj","Kumar","Machi"]
    ;
    print(names[0]);
    print(names.length);
    names.removeLast();
}
```

## MAP - Key - Value Pairs

**A dictionary — store and retrieve data by name (key).**

```
J U J U
pro(){
    map person = {} .
    person set name = "Sathish" .
    person set age = 21 .
    dis = person set name .
}
```

```
D A R T
void main() {
    Map person = {};
    person["name"] = "Sathish";
    person["age"] = 21;
    print(person["name"]);
}
```

| JuJu Code                            | Dart Code                          | What it does                |
|--------------------------------------|------------------------------------|-----------------------------|
| <code>list items = [1,2,3] .</code>  | <code>List items = [1,2,3];</code> | Create a list               |
| <code>dis = items[0] .</code>        | <code>print(items[0]);</code>      | Access item by index        |
| <code>items pop .</code>             | <code>items.removeLast();</code>   | Remove last item (v0.3)     |
| <code>items pop 0 .</code>           | <code>items.removeAt(0);</code>    | Remove item at index (v0.3) |
| <code>dis = items len .</code>       | <code>print(items.length);</code>  | Get list length (v0.3)      |
| <code>map person = {} .</code>       | <code>Map person = {};</code>      | Create a map                |
| <code>person set name = "x" .</code> | <code>person["name"] = "x";</code> | Set map value (v0.3)        |

| JuJu Code               | Dart Code              | What it does         |
|-------------------------|------------------------|----------------------|
| dis = person set name . | print(person["name"]); | Get map value (v0.3) |

■ LOCKED FOREVER – THESE KEYWORDS WILL NEVER CHANGE

list · map · len · pop · set

# v0.3 — Advanced Features

*back · null · len · pop · set · turn · ?: · pick*

Version 0.3 adds 8 new features on top of v1.0 + v0.2.

All previous rules remain LOCKED. Only new keywords added.

| #  | Feature        | JuJu Keyword | Dart Equivalent            |
|----|----------------|--------------|----------------------------|
| 16 | Return Value   | back         | return                     |
| 17 | Null Safety    | null         | null / String?             |
| 18 | Length         | len          | .length                    |
| 19 | List Remove    | pop          | .removeLast() / removeAt() |
| 20 | Map Access/Set | set          | map["key"] = value         |
| 21 | Type Casting   | turn         | int.parse() / toString()   |
| 22 | Ternary        | ? :          | condition ? a : b          |
| 23 | Switch/Case    | pick / is    | switch / case              |

*t u r n - T y p e C a s t i n g*

**Convert between types — especially useful after input.**

| JuJu                          | Dart                             | Converts         |
|-------------------------------|----------------------------------|------------------|
| <code>turn num "21" .</code>  | <code>int.parse("21")</code>     | String to int    |
| <code>turn dec "9.5" .</code> | <code>double.parse("9.5")</code> | String to double |
| <code>turn txt 42 .</code>    | <code>42.toString()</code>       | int to String    |

*p i c k - S w i t c h / C a s e*

**Choose between many options. is = case, or = default.**

J U J U

```
num day = 1 .
pick day {
  is 1 : dis = "Monday" .
  is 2 : dis = "Tuesday" .
  or : dis = "Other" .
}
```

D A R T

```
int day = 1;
switch(day) {
  case 1: print("Monday"); break;
  case 2: print("Tuesday"); break;
  default: print("Other");
}
```

#### T e r n a r y - O n e - L i n e C o n d i t i o n

J U J U

```
num age = 20 .
txt result = age gt 18 ? "Adult" : "Mi
nor" .
dis = result .
```

D A R T

```
int age = 20;
String result = age > 18 ? "Adult" : "
Minor";
print(result);
```

#### ■ LOCKED FOREVER – THESE KEYWORDS WILL NEVER CHANGE

back · null · len · pop · set · turn · ?: · pick · is(case)

# v0.4 — The Final 3 Keywords

*safe/fail · hold · blueprint/build*

v0.4 adds the 3 power keywords that separate a learner from a builder.

Real apps. Real data. Real objects.

| #  | JuJu Keyword           | Dart Equivalent              | What it does                             |
|----|------------------------|------------------------------|------------------------------------------|
| 24 | safe {} fail {}        | try {} catch(e) {}           | Exception handling — catch errors safely |
| 25 | hold fun / hold call() | async / await                | Async — wait without freezing app        |
| 26 | blueprint Name {}      | class Name {}                | Define reusable object structure         |
| 27 | build ClassName var .  | ClassName var = ClassName(); | Create object from blueprint             |

s a f e / f a i l — Exception Handling

## Don't let your app crash — catch errors safely.

```
J U J U
pro(){
  safe {
    dis = divide(10, 0) .
  } fail {
    dis = "Error caught!" .
  }
}
```

```
D A R T
void main() {
  try {
    print(divide(10, 0));
  } catch (e) {
    print("Error caught!");
  }
}
```

### N A M E D E R R O R V A R I A B L E

Use fail e {} to name the error and read exactly what went wrong: fail e { dis = e . } → catch(e) { print(e); }

h o l d — Async / Await

## Wait for data without freezing your app.

J U J U

```
hold fun fetchUser() {  
    txt data = hold getAPI() .  
    dis = data .  
}
```

D A R T

```
Future fetchUser() async {  
    String data = await getAPI();  
    print(data);  
}
```

| JuJu                           | Dart                                         | Use                       |
|--------------------------------|----------------------------------------------|---------------------------|
| hold fun name() { }            | Future<void> name() async { }                | Async function, no return |
| hold fun name() { back val . } | Future<dynamic> name() async { return val; } | Async with return value   |
| txt data = hold getAPI() .     | String data = await getAPI();                | Await text result         |
| hold saveData() .              | await saveData();                            | Await void call           |

b l u e p r i n t & b u i l d – Classes & Objects

## Define a reusable structure. Create real objects from it.

J U J U

```
blueprint Shirt {  
    txt color .  
    num size .  
    dec price .  
    fun stitch() {  
        dis = "Stitching $color" .  
    }  
}  
pro(){  
    build Shirt s1 .  
    s1 set color = "Blue" .  
    s1 set size = 42 .  
    dis = s1 set color .  
    s1 stitch() .  
}
```

D A R T

```
class Shirt {  
    String color;  
    int size;  
    double price;  
    Shirt() : color="", size=0, price=0.  
    0;  
    void stitch() {  
        print("Stitching $color");  
    }  
}  
void main() {  
    Shirt s1 = Shirt();  
    s1.color = "Blue";  
    s1.size = 42;  
    print(s1.color);  
    s1.stitch();  
}
```

| JuJu                 | Dart               | What it does                       |
|----------------------|--------------------|------------------------------------|
| blueprint Name { }   | class Name { }     | Define a reusable object structure |
| build Name var .     | Name var = Name(); | Create an object from blueprint    |
| var set prop = val . | var.prop = val;    | Set a property value               |
| dis = var set prop . | print(var.prop);   | Read a property value              |

| JuJu                        | Dart                       | What it does                |
|-----------------------------|----------------------------|-----------------------------|
| <code>var method() .</code> | <code>var.method();</code> | Call a method on the object |

#### AUTODEFAULTVALUES

JuJu automatically sets defaults when you build an object: `txt` → `""` · `num` → `0` · `dec` → `0.0` · `yon` → `false` — You do not write a constructor. JuJu handles it.

#### ■ LOCKED FOREVER — THESE KEYWORDS WILL NEVER CHANGE

`safe` · `fail` · `hold` · `blueprint` · `build`

## CHAPTER 11

# Auto-Fix Engine

*All 20 rules — how JuJu cleans your code automatically.*

JuJu's philosophy: the language adapts to the developer, not the other way around. The Auto-Fix Engine runs before the transpiler and cleans every mistake automatically.

### TIER 1 – Core Auto-Fix ( 8 rules )

| User Types                       | Auto-Fixed To              | What Happened                         |
|----------------------------------|----------------------------|---------------------------------------|
| <code>dis = "hi" (no dot)</code> | <code>dis = "hi" .</code>  | Dot missing at line end — auto-added  |
| <code>dis = "hi";</code>         | <code>dis = "hi" .</code>  | Semicolon found — removed, dot added  |
| <code>dis = 'hello'</code>       | <code>dis = "hello"</code> | Single quotes to double quotes        |
| <code>dis = "hello"</code>       | <code>dis = "hello"</code> | Smart/curly quotes to straight quotes |
| <code>DIS = "hi"</code>          | <code>dis = "hi"</code>    | Uppercase keywords to lowercase       |
| <code>dis = "hi"</code>          | <code>dis = "hi"</code>    | Extra spaces around = trimmed         |
| <code>dis="hi"</code>            | <code>dis = "hi"</code>    | No space around = — spaces added      |
| <code>pro(){ (no )}</code>       | <code>pro(){ ... }</code>  | Missing closing } — auto-added at end |

### TIER 2 – Smart Fix ( 9 rules )

| User Types                              | Auto-Fixed To                      | What Happened                         |
|-----------------------------------------|------------------------------------|---------------------------------------|
| <code>dis = hello</code>                | <code>dis = "hello"</code>         | No quotes on string — auto-wrapped    |
| <code>count i from 1 to 5 (no {)</code> | <code>count i from 1 to 5 {</code> | Missing { on loop — auto-added        |
| <code>dis = 42</code>                   | <code>print(42.toString());</code> | Number in dis — auto toString()       |
| <code>(blank lines)</code>              | <code>(skipped)</code>             | Empty/blank lines — ignored           |
| <code>print("hi")</code>                | <code>dis = "hi" .</code>          | Dart keyword used — converted to JuJu |
| <code>a + b (string concat)</code>      | <code>a join b</code>              | + operator — recognized, join applied |
| <code>"hello (no close quote)</code>    | <code>"hello"</code>               | Unclosed quote — auto-closed          |
| <code>fun greet( (no ))</code>          | <code>fun greet()</code>           | Unclosed bracket — auto-closed        |
| <code>group . length</code>             | <code>group.length</code>          | Space around dot — auto-trimmed       |

### TIER 3 – Warning System ( 3 rules )

| Condition         | What User Sees                                     |
|-------------------|----------------------------------------------------|
| Any auto-fix made | Yellow panel: "Line 3: dot missing — auto fixed ✓" |
| No issues found   | Green panel: "No issues found — clean code ✓"      |

| Condition      | What User Sees                              |
|----------------|---------------------------------------------|
| Multiple fixes | Counter: "3 auto-corrections made this run" |

## CHAPTER 12

# Complete Keyword Master List

*All 60+ keywords — locked forever.*

### PROGRAM STRUCTURE

| JuJu                           | Dart                                                          | Use Case                                |
|--------------------------------|---------------------------------------------------------------|-----------------------------------------|
| <code>pro(){ }</code>          | <code>void main(){ }</code>                                   | Entry point — every program starts here |
| <code>fun name(){ }</code>     | <code>void name(){ }</code>                                   | Define a reusable function              |
| <code>back value .</code>      | <code>return value;</code>                                    | Return a value from a function          |
| <code>dis = "text" .</code>    | <code>print("text");</code>                                   | Display / print to screen               |
| <code>output "text" .</code>   | <code>print("text");</code>                                   | Same as dis                             |
| <code>input name .</code>      | <code>String name = stdin.readLineSync()!;</code>             | Get text input from user                |
| <code>input num age .</code>   | <code>int age = int.parse(stdin.readLineSync()??""0");</code> | Get number input                        |
| <code>input dec price .</code> | <code>double price = double.parse(...);</code>                | Get decimal input                       |

### DATATYPES

| JuJu                           | Dart                              | Stores                                      |
|--------------------------------|-----------------------------------|---------------------------------------------|
| <code>txt / text</code>        | String                            | Text, words, sentences — anything in quotes |
| <code>num</code>               | int                               | Whole number — age, count, quantity         |
| <code>dec</code>               | double                            | Decimal number — price, weight, percentage  |
| <code>yon</code>               | bool                              | Yes/No — true or false only                 |
| <code>guess</code>             | var                               | Auto detect type — JuJu guesses             |
| <code>list</code>              | List                              | Multiple items in one place                 |
| <code>map</code>               | Map                               | Key-value pairs — dictionary                |
| <code>txt name = null .</code> | <code>String? name = null;</code> | Nullable variable — null safety (v0.3)      |

### CONTROLFLOW

| JuJu                                  | Dart                                          | Use                                      |
|---------------------------------------|-----------------------------------------------|------------------------------------------|
| <code>check cond { }</code>           | <code>if (cond) { }</code>                    | If condition                             |
| <code>check or cond { }</code>        | <code>else if (cond) { }</code>               | Else if condition                        |
| <code>or { }</code>                   | <code>else { }</code>                         | Else — when all conditions fail          |
| <code>count i from 1 to 10 { }</code> | <code>for (int i=1; i&lt;=10; i++) { }</code> | For loop — fixed count                   |
| <code>do cond { }</code>              | <code>while (cond) { }</code>                 | While loop — repeat while condition true |
| <code>do first cond { }</code>        | <code>do { } while (cond);</code>             | Do-while — runs at least once            |

| JuJu                                        | Dart                                              | Use                |
|---------------------------------------------|---------------------------------------------------|--------------------|
| <code>pick val { is 1: ... or: ... }</code> | <code>switch(val) { case 1: ... default: }</code> | Switch/case (v0.3) |

#### v 0 . 3 – A D V A N C E D

| JuJu                                     | Dart                                           | Use                          |
|------------------------------------------|------------------------------------------------|------------------------------|
| <code>name len .</code>                  | <code>.length</code>                           | Get length of string or list |
| <code>items pop .</code>                 | <code>.removeLast()</code>                     | Remove last item             |
| <code>items pop 0 .</code>               | <code>.removeAt(0)</code>                      | Remove item at index         |
| <code>person set key = val .</code>      | <code>person["key"] = val;</code>              | Map write                    |
| <code>person set key .</code>            | <code>person["key"]</code>                     | Map read                     |
| <code>turn num "21" .</code>             | <code>int.parse("21")</code>                   | String to int                |
| <code>turn dec "9.5" .</code>            | <code>double.parse("9.5")</code>               | String to double             |
| <code>turn txt 42 .</code>               | <code>42.toString()</code>                     | int to String                |
| <code>txt x = null .</code>              | <code>String? x = null;</code>                 | Null safety                  |
| <code>txt r = a gt 18 ? "A" : "B"</code> | <code>String r = a &gt; 18 ? "A" : "B";</code> | Ternary                      |

#### v 0 . 4 – T H E F I N A L 3

| JuJu                                    | Dart                                             | Use                                      |
|-----------------------------------------|--------------------------------------------------|------------------------------------------|
| <code>safe { } fail { }</code>          | <code>try { } catch(e) { }</code>                | Exception handling — catch errors safely |
| <code>safe { } fail e { }</code>        | <code>try { } catch(e) { print(e); }</code>      | Named error variable                     |
| <code>hold fun name() { }</code>        | <code>Future&lt;void&gt; name() async { }</code> | Async function                           |
| <code>txt data = hold getAPI() .</code> | <code>String data = await getAPI();</code>       | Await a call                             |
| <code>hold saveData() .</code>          | <code>await saveData();</code>                   | Await void call                          |
| <code>blueprint Name { }</code>         | <code>class Name { }</code>                      | Define reusable object structure         |
| <code>build ClassName var .</code>      | <code>ClassName var = ClassName();</code>        | Create object from blueprint             |
| <code>var set prop = val .</code>       | <code>var.prop = val;</code>                     | Set property on object                   |
| <code>dis = var set prop .</code>       | <code>print(var.prop);</code>                    | Read property from object                |
| <code>var method() .</code>             | <code>var.method();</code>                       | Call method on object                    |

#### ■ LOCKED FOREVER — THESE KEYWORDS WILL NEVER CHANGE

v1.0: `pro . dis . output . input . txt/text . num . dec . yon . guess + - x / % . is not gt lt gte lte . tt or opp = add sub mul divby up dn . count . do . do first / comment . ** comment ** . $ interpolation . join`  
v0.2: `check . check or . or . fun`  
v0.3: `back . null . len . pop . set . turn . ?: . pick . is(case)`  
v0.4: `safe . fail . hold . blueprint . build`  
TOTAL: 60+ keywords — ALL LOCKED FOREVER.

*"You have now learned all 60+ keywords of JuJu. From your very first dis = \"Vanakkam!\" to building async apps with blueprints — you have walked the entire path. Write Simple. Build Real. Understand Deeply."*

— END OF JUJU ENGINE REFERENCE MASTER · TIRUPUR, TAMIL NADU