
Contents

| | |
|---------------------------------|-----------|
| Preface | xi |
| The Origins of Go | xii |
| The Go Project | xiii |
| Organization of the Book | xv |
| Where to Find More Information | xvi |
| Acknowledgments | xvii |
| 1. Tutorial | 1 |
| 1.1. Hello, World | 1 |
| 1.2. Command-Line Arguments | 4 |
| 1.3. Finding Duplicate Lines | 8 |
| 1.4. Animated GIFs | 13 |
| 1.5. Fetching a URL | 15 |
| 1.6. Fetching URLs Concurrently | 17 |
| 1.7. A Web Server | 19 |
| 1.8. Loose Ends | 23 |
| 2. Program Structure | 27 |
| 2.1. Names | 27 |
| 2.2. Declarations | 28 |
| 2.3. Variables | 30 |
| 2.4. Assignments | 36 |
| 2.5. Type Declarations | 39 |
| 2.6. Packages and Files | 41 |
| 2.7. Scope | 45 |

| | |
|-------------------------------------------------|------------|
| 3. Basic Data Types | 51 |
| 3.1. Integers | 51 |
| 3.2. Floating-Point Numbers | 56 |
| 3.3. Complex Numbers | 61 |
| 3.4. Booleans | 63 |
| 3.5. Strings | 64 |
| 3.6. Constants | 75 |
| 4. Composite Types | 81 |
| 4.1. Arrays | 81 |
| 4.2. Slices | 84 |
| 4.3. Maps | 93 |
| 4.4. Structs | 99 |
| 4.5. JSON | 107 |
| 4.6. Text and HTML Templates | 113 |
| 5. Functions | 119 |
| 5.1. Function Declarations | 119 |
| 5.2. Recursion | 121 |
| 5.3. Multiple Return Values | 124 |
| 5.4. Errors | 127 |
| 5.5. Function Values | 132 |
| 5.6. Anonymous Functions | 135 |
| 5.7. Variadic Functions | 142 |
| 5.8. Deferred Function Calls | 143 |
| 5.9. Panic | 148 |
| 5.10. Recover | 151 |
| 6. Methods | 155 |
| 6.1. Method Declarations | 155 |
| 6.2. Methods with a Pointer Receiver | 158 |
| 6.3. Composing Types by Struct Embedding | 161 |
| 6.4. Method Values and Expressions | 164 |
| 6.5. Example: Bit Vector Type | 165 |
| 6.6. Encapsulation | 168 |
| 7. Interfaces | 171 |
| 7.1. Interfaces as Contracts | 171 |
| 7.2. Interface Types | 174 |
| 7.3. Interface Satisfaction | 175 |
| 7.4. Parsing Flags with <code>flag.Value</code> | 179 |
| 7.5. Interface Values | 181 |
| 7.6. Sorting with <code>sort.Interface</code> | 186 |

CONTENTS

ix

| | |
|---------------------------------------------------------|------------|
| 7.7. The <code>http.Handler</code> Interface | 191 |
| 7.8. The error Interface | 196 |
| 7.9. Example: Expression Evaluator | 197 |
| 7.10. Type Assertions | 205 |
| 7.11. Discriminating Errors with Type Assertions | 206 |
| 7.12. Querying Behaviors with Interface Type Assertions | 208 |
| 7.13. Type Switches | 210 |
| 7.14. Example: Token-Based XML Decoding | 213 |
| 7.15. A Few Words of Advice | 216 |
| 8. Goroutines and Channels | 217 |
| 8.1. Goroutines | 217 |
| 8.2. Example: Concurrent Clock Server | 219 |
| 8.3. Example: Concurrent Echo Server | 222 |
| 8.4. Channels | 225 |
| 8.5. Looping in Parallel | 234 |
| 8.6. Example: Concurrent Web Crawler | 239 |
| 8.7. Multiplexing with <code>select</code> | 244 |
| 8.8. Example: Concurrent Directory Traversal | 247 |
| 8.9. Cancellation | 251 |
| 8.10. Example: Chat Server | 253 |
| 9. Concurrency with Shared Variables | 257 |
| 9.1. Race Conditions | 257 |
| 9.2. Mutual Exclusion: <code>sync.Mutex</code> | 262 |
| 9.3. Read/Write Mutexes: <code>sync.RWMutex</code> | 266 |
| 9.4. Memory Synchronization | 267 |
| 9.5. Lazy Initialization: <code>sync.Once</code> | 268 |
| 9.6. The Race Detector | 271 |
| 9.7. Example: Concurrent Non-Blocking Cache | 272 |
| 9.8. Goroutines and Threads | 280 |
| 10. Packages and the Go Tool | 283 |
| 10.1. Introduction | 283 |
| 10.2. Import Paths | 284 |
| 10.3. The Package Declaration | 285 |
| 10.4. Import Declarations | 285 |
| 10.5. Blank Imports | 286 |
| 10.6. Packages and Naming | 289 |
| 10.7. The Go Tool | 290 |
| 11. Testing | 301 |
| 11.1. The <code>go test</code> Tool | 302 |

| | |
|-------------------------------------------------------------------------------------|------------|
| 11.2. Test Functions | 302 |
| 11.3. Coverage | 318 |
| 11.4. Benchmark Functions | 321 |
| 11.5. Profiling | 323 |
| 11.6. Example Functions | 326 |
| 12. Reflection | 329 |
| 12.1. Why Reflection? | 329 |
| 12.2. <code>reflect.Type</code> and <code>reflect.Value</code> | 330 |
| 12.3. <code>Display</code> , a Recursive Value Printer | 333 |
| 12.4. Example: Encoding S-Expressions | 338 |
| 12.5. Setting Variables with <code>reflect.Value</code> | 341 |
| 12.6. Example: Decoding S-Expressions | 344 |
| 12.7. Accessing Struct Field Tags | 348 |
| 12.8. Displaying the Methods of a Type | 351 |
| 12.9. A Word of Caution | 352 |
| 13. Low-Level Programming | 353 |
| 13.1. <code>unsafe.Sizeof</code> , <code>Alignof</code> , and <code>Offsetof</code> | 354 |
| 13.2. <code>unsafe.Pointer</code> | 356 |
| 13.3. Example: Deep Equivalence | 358 |
| 13.4. Calling C Code with <code>cgo</code> | 361 |
| 13.5. Another Word of Caution | 366 |
| Index | 367 |