

These pages contain selected sections of [Structure and Interpretation of Computer Programs](#) by Harold Abelson and Gerald Sussman with Julie Sussman.

The programs are given in JavaScript, translated from Scheme by Martin Henz. The JavaScript programs can be executed (evaluated) by clicking on them.

If you are interested, here is [some information](#) on how these pages were generated, and what tools are used “under the hood”.

Contents

[1 Building Abstractions with Functions](#)

[1.1 The Elements of Programming](#)

[1.1.1 Expressions](#)

[1.1.2 Naming and the Environment](#)

[1.1.3 Evaluating Operator Combinations](#)

[1.1.4 Functions](#)

[1.1.5 The Substitution Model for Function Application](#)

[1.1.6 Conditional Statements and Predicates](#)

[1.1.7 Example: Square Roots by Newton’s Method](#)

[1.1.8 Functions as Black-Box Abstractions](#)

[1.2 Functions and the Processes They Generate](#)

[1.2.1 Linear Recursion and Iteration](#)

[1.2.2 Tree Recursion](#)

[1.2.3 Orders of Growth](#)

[1.2.4 Exponentiation](#)

[1.2.5 Greatest Common Divisors](#)

[1.2.6 Example: Testing for Primality](#)

[1.3 Formulating Abstractions with Higher-Order Functions](#)

[1.3.1 Functions as Arguments](#)

[1.3.2 Function Definition Expressions](#)

[1.3.3 Functions as General Methods](#)

[1.3.4 Functions as Returned Values](#)

[2 Building Abstractions with Data](#)

[2.1 Introduction to Data Abstraction](#)

[2.1.1 Example: Arithmetic Operations for Rational Numbers](#)

[2.1.2 Abstraction Barriers](#)

[2.1.3 What Is Meant by Data?](#)

[2.1.4 Extended Exercise: Interval Arithmetic](#)

[2.2 Hierarchical Data and the Closure Property](#)

[2.2.1 Representing Sequences](#)

[2.2.2 Hierarchical Structures](#)[2.2.3 Sequences as Conventional Interfaces](#)[2.2.4 Example: A Picture Language](#)[2.3 Symbolic Data](#)[2.3.1 Strings](#)[2.3.2 Example: Symbolic Differentiation](#)[2.3.3 Example: Representing Sets](#)[2.3.4 Example: Huffman Encoding Trees](#)[2.4 Multiple Representations for Abstract Data](#)[2.4.1 Representations for Complex Numbers](#)[2.4.2 Tagged data](#)[2.4.3 Data-Directed Programming and Additivity](#)[2.5 Systems with Generic Operations](#)[2.5.1 Generic Arithmetic Operations](#)[2.5.2 Combining Data of Different Types](#)[3 Modularity, Objects, and State](#)[3.1 Assignment and Local State](#)[3.1.1 Local State Variables](#)[3.1.2 The Benefits of Introducing Assignment](#)[3.1.3 The Costs of Introducing Assignment](#)[3.2 The Environment Model of Evaluation](#)[3.2.1 The Rules for Evaluation](#)[3.2.2 Applying Simple Functions](#)[3.2.3 Frames as the Repository of Local State](#)[3.2.4 Internal Definitions](#)[3.3 Modeling with Mutable Data](#)[3.3.1 Mutable List Structure](#)[3.3.2 Representing Queues](#)[3.3.3 Representing Tables](#)[3.4 Concurrency: Time Is of the Essence](#)[3.5 Streams](#)[3.5.1 Streams Are Delayed Lists](#)[3.5.2 Infinite Streams](#)[4 Metalinguistic Abstraction](#)[4.1 The Metacircular Evaluator](#)[4.1.1 The Core of the Evaluator](#)[4.1.2 Representing Statements and Expressions](#)[4.1.3 Evaluator Data Structures](#)[4.1.4 Running the Evaluator as a Program](#)

[5 Computing with Register Machines](#)