



**C# 3.0:**

---

**The Complete Reference**

## About the Author

---

**Herbert Schildt** is a leading authority on C#, C++, C, and Java. His programming books have sold more than 3.5 million copies worldwide and have been translated into all major foreign languages. He is the author of numerous bestsellers, including *Java: The Complete Reference*, *C++: The Complete Reference*, *C: The Complete Reference*, and *C#: A Beginner's Guide*. Although interested in all facets of computing, his primary focus is computer languages, including compilers, interpreters, and robotic control languages. He also has an active interest in the standardization of languages. Schildt holds both graduate and undergraduate degrees from the University of Illinois. He can be reached at his consulting office at (217) 586-4683. His web site is [www.HerbSchildt.com](http://www.HerbSchildt.com).

## About the Technical Editor

**Michael Howard** (Austin, Texas) is a principal security program manager on the Trustworthy Computing (TwC) Group's Security Engineering team at Microsoft, where he is responsible for managing secure design, programming, and testing techniques across the company. Howard is an architect of the Security Development Lifecycle (SDL), a process for improving the security of Microsoft's software. Howard speaks regularly on the topic of securing code for Microsoft and at conferences worldwide. He regularly publishes articles on security design and is the co-author of six security books, including the award-winning *Writing Secure Code*, *19 Deadly Sins of Software Security*, *The Security Development Lifecycle*, and his most recent release, *Writing Secure Code for Windows Vista*.

# **C# 3.0:**

---

# **The Complete Reference**

**Herbert Schildt**



New York Chicago San Francisco  
Lisbon London Madrid Mexico City  
Milan New Delhi San Juan  
Seoul Singapore Sydney Toronto

Copyright © 2009 by The McGraw-Hill Companies, Inc. All rights reserved. Except as permitted under the United States Copyright Act of 1976, no part of this publication may be reproduced or distributed in any form or by any means, or stored in a data base or retrieval system, without the prior written permission of the publisher.

ISBN: 978-0-07-159842-2

MHID: 0-07-159842-6

The material in this eBook also appears in the print version of this title: ISBN: 978-0-07-158841-6, MHID: 0-07-158841-8.

All trademarks are trademarks of their respective owners. Rather than put a trademark symbol after every occurrence of a trademarked name, we use names in an editorial fashion only, and to the benefit of the trademark owner, with no intention of infringement of the trademark. Where such designations appear in this book, they have been printed with initial caps.

McGraw-Hill books are available at special quantity discounts to use as premiums and sales promotions, or for use in corporate training programs. To contact a representative, please visit the Contact Us page at [www.mhprofessional.com](http://www.mhprofessional.com).

Information has been obtained by McGraw-Hill from sources believed to be reliable. However, because of the possibility of human or mechanical error by our sources, McGraw-Hill, or others, McGraw-Hill does not guarantee the accuracy, adequacy, or completeness of any information and is not responsible for any errors or omissions or the results obtained from the use of such information.

#### TERMS OF USE

This is a copyrighted work and The McGraw-Hill Companies, Inc. (“McGraw-Hill”) and its licensors reserve all rights in and to the work. Use of this work is subject to these terms. Except as permitted under the Copyright Act of 1976 and the right to store and retrieve one copy of the work, you may not decompile, disassemble, reverse engineer, reproduce, modify, create derivative works based upon, transmit, distribute, disseminate, sell, publish or sublicense the work or any part of it without McGraw-Hill’s prior consent. You may use the work for your own noncommercial and personal use; any other use of the work is strictly prohibited. Your right to use the work may be terminated if you fail to comply with these terms.

THE WORK IS PROVIDED “AS IS.” MCGRAW-HILL AND ITS LICENSORS MAKE NO GUARANTEES OR WARRANTIES AS TO THE ACCURACY, ADEQUACY OR COMPLETENESS OF OR RESULTS TO BE OBTAINED FROM USING THE WORK, INCLUDING ANY INFORMATION THAT CAN BE ACCESSED THROUGH THE WORK VIA HYPERLINK OR OTHERWISE, AND EXPRESSLY DISCLAIM ANY WARRANTY, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. McGraw-Hill and its licensors do not warrant or guarantee that the functions contained in the work will meet your requirements or that its operation will be uninterrupted or error free. Neither McGraw-Hill nor its licensors shall be liable to you or anyone else for any inaccuracy, error or omission, regardless of cause, in the work or for any damage resulting therefrom. McGraw-Hill has no responsibility for the content of any information accessed through the work. Under no circumstances shall McGraw-Hill and/or its licensors be liable for any

indirect, incidental, special, punitive, consequential or similar damages that result from the use of or inability to use the work, even if any of them has been advised of the possibility of such damages. This limitation of liability shall apply to any claim or cause whatsoever whether such claim or cause arises in contract, tort or otherwise.



# Contents at a Glance

---

## [Part I The C# Language](#)

[1 The Creation of C#](#)

[2 An Overview of C#](#)

[3 Data Types, Literals, and Variables](#)

[4 Operators](#)

[5 Program Control Statements](#)

[6 Introducing Classes and Objects](#)

[7 Arrays and Strings](#)

[8 A Closer Look at Methods and Classes](#)

[9 Operator Overloading](#)

[10 Indexers and Properties](#)

[11 Inheritance](#)

[12 Interfaces, Structures, and Enumerations](#)

[13 Exception Handling](#)

[14 Using I/O](#)

[15 Delegates, Events, and Lambda Expressions](#)

[16 Namespaces, the Preprocessor, and Assemblies](#)

[17 Runtime Type ID, Reflection, and Attributes](#)

[18 Generics](#)

[19 LINQ](#)

[20 Unsafe Code, Pointers, Nullable Types, and Miscellaneous Topics](#)

## [Part II Exploring the C# Library](#)

[21 Exploring the System Namespace](#)



[22 Strings and Formatting](#)

---

[23 Multithreaded Programming](#)

[24 Collections, Enumerators, and Iterators](#)

[25 Networking Through the Internet Using System.Net](#)

[26 Use System.Windows.Forms to Create Form-Based Windows Applications](#)

[A Documentation Comment Quick Reference](#)

[Index](#)

# Contents

---

[Special Thanks](#)

[Preface](#)

## **[Part I The C# Language](#)**

### **[1 The Creation of C#](#)**

[C#'s Family Tree](#)

[C: The Beginning of the Modern Age of Programming](#)

[The Creation of OOP and C++](#)

[The Internet and Java Emerge](#)

[The Creation of C#](#)

[The Evolution of C#](#)

[How C# Relates to the .NET Framework](#)

[What Is the .NET Framework?](#)

[How the Common Language Runtime Works](#)

[Managed vs. Unmanaged Code](#)

[The Common Language Specification](#)

### **[2 An Overview of C#](#)**

[Object-Oriented Programming](#)

[Encapsulation](#)

[Polymorphism](#)

[Inheritance](#)

[A First Simple Program](#)

[Using csc.exe, the C# Command-Line Compiler](#)

[Using the Visual Studio IDE](#)

[The First Sample Program, Line by Line](#)

[Handling Syntax Errors](#)

[A Small Variation](#)

[A Second Simple Program](#)

[Another Data Type](#)

[Two Control Statements](#)

[The if Statement](#)

[The for Loop](#)

[Using Code Blocks](#)

[Semicolons, Positioning, and Indentation](#)

[The C# Keywords](#)

---

[Identifiers](#)

[The .NET Framework Class Library](#)

### **[3 Data Types, Literals, and Variables](#)**

[Why Data Types Are Important](#)

[C#'s Value Types](#)

[Integers](#)

[Floating-Point Types](#)

[The decimal Type](#)

[Characters](#)

[The bool Type](#)

[Some Output Options](#)

[Literals](#)

[Hexadecimal Literals](#)

[Character Escape Sequences](#)

[String Literals](#)

[A Closer Look at Variables](#)

[Initializing a Variable](#)

[Dynamic Initialization](#)

[Implicitly Typed Variables](#)

[The Scope and Lifetime of Variables](#)

[Type Conversion and Casting](#)

[Automatic Conversions](#)

[Casting Incompatible Types](#)

[Type Conversion in Expressions](#)

[Using Casts in Expressions](#)

### **[4 Operators](#)**

[Arithmetic Operators](#)

[Increment and Decrement](#)

[Relational and Logical Operators](#)

[Short-Circuit Logical Operators](#)

[The Assignment Operator](#)

[Compound Assignments](#)

[The Bitwise Operators](#)

[The Bitwise AND, OR, XOR, and NOT Operators](#)

[The Shift Operators](#)

---

[Bitwise Compound Assignments](#)

[The ? Operator](#)

[Spacing and Parentheses](#)

[Operator Precedence](#)

## **[5 Program Control Statements](#)**

[The if Statement](#)

[Nested ifs](#)

[The if-else-if Ladder](#)

[The switch Statement](#)

[Nested switch Statements](#)

[The for Loop](#)

[Some Variations on the for Loop](#)

[The while Loop](#)

[The do-while Loop](#)

[The foreach Loop](#)

[Using break to Exit a Loop](#)

[Using continue](#)

[return](#)

[The goto](#)

## **[6 Introducing Classes and Objects](#)**

[Class Fundamentals](#)

[The General Form of a Class](#)

[Define a Class](#)

[How Objects Are Created](#)

[Reference Variables and Assignment](#)

[Methods](#)

[Add a Method to the Building Class](#)

[Return from a Method](#)

[Return a Value](#)

[Use Parameters](#)

[Add a Parameterized Method to Building](#)

[Avoiding Unreachable Code](#)

[Constructors](#)

[Parameterized Constructors](#)

[The new Operator Revisited](#)

[Using new with Value Types](#)

[Garbage Collection and Destructors](#)

[Destructors](#)

[The this Keyword](#)

## **[7 Arrays and Strings](#)**

[Arrays](#)

[One-Dimensional Arrays](#)

[Multidimensional Arrays](#)

[Two-Dimensional Arrays](#)

[Arrays of Three or More Dimensions](#)

[Initializing Multidimensional Arrays](#)

[Jagged Arrays](#)

[Assigning Array References](#)

[Using the Length Property](#)

[Using Length with Jagged Arrays](#)

[Implicitly Typed Arrays](#)

[The foreach Loop](#)

[Strings](#)

[Constructing Strings](#)

[Operating on Strings](#)

[Arrays of Strings](#)

[Strings Are Immutable](#)

[Strings Can Be Used in switch Statements](#)

## **[8 A Closer Look at Methods and Classes](#)**

[Controlling Access to Class Members](#)

[C#'s Access Modifiers](#)

[Applying Public and Private Access](#)

[Controlling Access: A Case Study](#)

[Pass References to Methods](#)

[How Arguments Are Passed](#)

[Use ref and out Parameters](#)

[Use ref](#)

[Use out](#)

[Use ref and out on References](#)

---

[Use a Variable Number of Arguments](#)

[Return Objects](#)

[Return an Array](#)

[Method Overloading](#)

[Overload Constructors](#)

[Invoke an Overloaded Constructor Through this](#)

[Object Initializers](#)

[The Main\(\) Method](#)

[Return Values from Main\(\)](#)

[Pass Arguments to Main\(\)](#)

[Recursion](#)

[Understanding static](#)

[Static Constructors](#)

[Static Classes](#)

## **[9 Operator Overloading](#)**

[Operator Overloading Fundamentals](#)

[Overloading Binary Operators](#)

[Overloading Unary Operators](#)

[Handling Operations on C# Built-in Types](#)

[Overloading the Relational Operators](#)

[Overloading true and false](#)

[Overloading the Logical Operators](#)

[A Simple Approach to Overloading the Logical Operators](#)

[Enabling the Short-Circuit Operators](#)

[Conversion Operators](#)

[Operator Overloading Tips and Restrictions](#)

[Another Example of Operator Overloading](#)

## **[10 Indexers and Properties](#)**

[Indexers](#)

[Creating One-Dimensional Indexers](#)

[Indexers Can Be Overloaded](#)

[Indexers Do Not Require an Underlying Array](#)

[Multidimensional Indexers](#)

[Properties](#)

[Auto-Implemented Properties](#)

---

[Use Object Initializers with Properties](#)

[Property Restrictions](#)

[Use Access Modifiers with Accessors](#)

[Using Indexers and Properties](#)

## **11 Inheritance**

[Inheritance Basics](#)

[Member Access and Inheritance](#)

[Using Protected Access](#)

[Constructors and Inheritance](#)

[Calling Base Class Constructors](#)

[Inheritance and Name Hiding](#)

[Using base to Access a Hidden Name](#)

[Creating a Multilevel Hierarchy](#)

[When Are Constructors Called?](#)

[Base Class References and Derived Objects](#)

[Virtual Methods and Overriding](#)

[Why Overridden Methods?](#)

[Applying Virtual Methods](#)

[Using Abstract Classes](#)

[Using sealed to Prevent Inheritance](#)

[The object Class](#)

[Boxing and Unboxing](#)

[Is object a Universal Data Type?](#)

## **12 Interfaces, Structures, and Enumerations**

[Interfaces](#)

[Implementing Interfaces](#)

[Using Interface References](#)

[Interface Properties](#)

[Interface Indexers](#)

[Interfaces Can Be Inherited](#)

[Name Hiding with Interface Inheritance](#)

[Explicit Implementations](#)

[Choosing Between an Interface and an Abstract Class](#)

[The .NET Standard Interfaces](#)

[Why Structures?](#)

[Enumerations](#)

[Initialize an Enumeration](#)

[Specify the Underlying Type of an Enumeration](#)

[Use Enumerations](#)

**[13 Exception Handling](#)**

[The System.Exception Class](#)

[Exception Handling Fundamentals](#)

[Using try and catch](#)

[A Simple Exception Example](#)

[A Second Exception Example](#)

[The Consequences of an Uncaught Exception](#)

[Exceptions Let You Handle Errors Gracefully](#)

[Using Multiple catch Clauses](#)

[Catching All Exceptions](#)

[Nesting try Blocks](#)

[Throwing an Exception](#)

[Rethrowing an Exception](#)

[Using finally](#)

[A Closer Look at the Exception Class](#)

[Commonly Used Exceptions](#)

[Deriving Exception Classes](#)

[Catching Derived Class Exceptions](#)

[Using checked and unchecked](#)

**[14 Using I/O](#)**

[C#'s I/O Is Built Upon Streams](#)

[Byte Streams and Character Streams](#)

[The Predefined Streams](#)

[The Stream Classes](#)

[The Stream Class](#)

[The Byte Stream Classes](#)

[The Character Stream Wrapper Classes](#)

[Binary Streams](#)

[Console I/O](#)



[Reading Console Input](#)

---

[Using ReadKey\(\)](#)

[Writing Console Output](#)

[FileStream and Byte-Oriented File I/O](#)

[Opening and Closing a File](#)

[Reading Bytes from a FileStream](#)

[Writing to a File](#)

[Using FileStream to Copy a File](#)

[Character-Based File I/O](#)

[Using StreamWriter](#)

[Using a StreamReader](#)

[Redirecting the Standard Streams](#)

[Reading and Writing Binary Data](#)

[BinaryWriter](#)

[BinaryReader](#)

[Demonstrating Binary I/O](#)

[Random Access Files](#)

[Using MemoryStream](#)

[Using StringReader and StringWriter](#)

[Converting Numeric Strings to Their Internal Representation](#)

## **[15 Delegates, Events, and Lambda Expressions](#)**

[Delegates](#)

[Delegate Method Group Conversion](#)

[Using Instance Methods as Delegates](#)

[Multicasting](#)

[Covariance and Contravariance](#)

[System.Delegate](#)

[Why Delegates](#)

[Anonymous Functions](#)

[Anonymous Methods](#)

[Pass Arguments to an Anonymous Method](#)

[Return a Value from an Anonymous Method](#)

[Use Outer Variables with Anonymous Methods](#)

[Lambda Expressions](#)

[The Lambda Operator](#)

[Expression Lambdas](#)

[Events](#)

[A Multicast Event Example](#)

[Instance Methods vs. Static Methods as Event Handlers](#)

[Using Event Accessors](#)

[Miscellaneous Event Features](#)

[Use Anonymous Methods and Lambda Expressions with Events](#)

[.NET Event Guidelines](#)

[Use EventHandler](#)

[Applying Events: A Case Study](#)

**[16 Namespaces, the Preprocessor, and Assemblies](#)**

[Namespaces](#)

[Declaring a Namespace](#)

[Namespaces Prevent Name Conflicts](#)

[using](#)

[A Second Form of using](#)

[Namespaces Are Additive](#)

[Namespaces Can Be Nested](#)

[The Global Namespace](#)

[Using the :: Namespace Alias Qualifier](#)

[The Preprocessor](#)

[#define](#)

[#if and #endif](#)

[#else and #elif](#)

[#undef](#)

[#error](#)

[#warning](#)

[#line](#)

[#region and #endregion](#)

[#pragma](#)

[Assemblies and the internal Access Modifier](#)

[The internal Access Modifier](#)

**[17 Runtime Type ID, Reflection, and Attributes](#)**

[Runtime Type Identification](#)

[Testing a Type with is](#)

[Using as](#)

---

[Using typeof](#)

[Reflection](#)

[The Reflection Core: System.Type](#)

[Using Reflection](#)

[Obtaining Information About Methods](#)

[Calling Methods Using Reflection](#)

[Obtaining a Type's Constructors](#)

[Obtaining Types from Assemblies](#)

[Fully Automating Type Discovery](#)

[Attributes](#)

[Attribute Basics](#)

[Positional vs. Named Parameters](#)

[Three Built-in Attributes](#)

[AttributeUsage](#)

[The Conditional Attribute](#)

[The Obsolete Attribute](#)

## **[18 Generics](#)**

[What Are Generics?](#)

[A Simple Generics Example](#)

[Generic Types Differ Based on Their Type Arguments](#)

[How Generics Improve Type Safety](#)

[A Generic Class with Two Type Parameters](#)

[The General Form of a Generic Class](#)

[Constrained Types](#)

[Using a Base Class Constraint](#)

[Using an Interface Constraint](#)

[Using the new\( \) Constructor Constraint](#)

[The Reference Type and Value Type Constraints](#)

[Using a Constraint to Establish a Relationship Between Two Type Parameters](#)

[Using Multiple Constraints](#)

[Creating a Default Value of a Type Parameter](#)

[Generic Structures](#)

[Creating a Generic Method](#)

[Using Explicit Type Arguments to Call a Generic Method](#)

[Using a Constraint with a Generic Method](#)

[Generic Delegates](#)

---

[Generic Interfaces](#)

[Comparing Instances of a Type Parameter](#)

[Generic Class Hierarchies](#)

[Using a Generic Base Class](#)

[A Generic Derived Class](#)

[Overriding Virtual Methods in a Generic Class](#)

[Overloading Methods That Use Type Parameters](#)

[How Generic Types Are Instantiated](#)

[Some Generic Restrictions](#)

[Final Thoughts on Generics](#)

## **19 LINQ**

[What Is LINQ?](#)

[LINQ Fundamentals](#)

[A Simple Query](#)

[A Query Can Be Executed More Than Once](#)

[How the Data Types in a Query Relate](#)

[The General Form of a Query](#)

[Filter Values with where](#)

[Sort Results with orderby](#)

[A Closer Look at select](#)

[Use Nested from Clauses](#)

[Group Results with group](#)

[Use into to Create a Continuation](#)

[Use let to Create a Variable in a Query](#)

[Join Two Sequences with join](#)

[Anonymous Types](#)

[Create a Group Join](#)

[The Query Methods](#)

[The Basic Query Methods](#)

[Create Queries by Using the Query Methods](#)

[Query Syntax vs. Query Methods](#)

[More Query-Related Extension Methods](#)

[Deferred vs. Immediate Query Execution](#)

[Expression Trees](#)

## **[20 Unsafe Code, Pointers, Nullable Types, and Miscellaneous Topics](#)**

### [Unsafe Code](#)

[Pointer Basics](#)

[Using unsafe](#)

[Using fixed](#)

[Accessing Structure Members Through a Pointer](#)

[Pointer Arithmetic](#)

[Pointer Comparisons](#)

[Pointers and Arrays](#)

[Pointers and Strings](#)

[Multiple Indirection](#)

[Arrays of Pointers](#)

[stackalloc](#)

[Creating Fixed-Size Buffers](#)

### [Nullable Types](#)

[Nullable Basics](#)

[Nullable Objects in Expressions](#)

[The ?? Operator](#)

[Nullable Objects and the Relational and Logical Operators](#)

### [Partial Types](#)

### [Partial Methods](#)

### [Friend Assemblies](#)

### [Miscellaneous Keywords](#)

[lock](#)

[readonly](#)

[const and volatile](#)

[The using Statement](#)

[extern](#)

## **[Part II Exploring the C# Library](#)**

### **[21 Exploring the System Namespace](#)**

[The Members of System](#)

[The Math Class](#)

[The .NET Structures Corresponding to the Built-in Value Types](#)

[The Integer Structures](#)

[The Floating-Point Structures](#)

---

[Decimal](#)

[Char](#)

[The Boolean Structure](#)

[The Array Class](#)

[Sorting and Searching Arrays](#)

[Reversing an Array](#)

[Copying an Array](#)

[Using a Predicate](#)

[Using an Action](#)

[BitConverter](#)

[Generating Random Numbers with Random](#)

[Memory Management and the GC Class](#)

[Object](#)

[The IComparable and IComparable<T> Interfaces](#)

[The IEquatable<T> Interface](#)

[The IConvertible Interface](#)

[The ICloneable Interface](#)

[IFormatProvider and IFormattable](#)

## **[22 Strings and Formatting](#)**

[Strings in C#](#)

[The String Class](#)

[The String Constructors](#)

[The String Field, Indexer, and Property](#)

[The String Operators](#)

[The String Methods](#)

[Padding and Trimming Strings](#)

[Inserting, Removing, and Replacing](#)

[Changing Case](#)

[Using the Substring\( \) Method](#)

[The String Extension Methods](#)

[Formatting](#)

[Formatting Overview](#)

[The Numeric Format Specifiers](#)

[Understanding Argument Numbers](#)

[Using String.Format\( \) and ToString\( \) to Format Data](#)

[Using String.Format\(\) to Format Values](#)

---

[Using ToString\(\) to Format Data](#)

[Creating a Custom Numeric Format](#)

[The Custom Format Placeholder Characters](#)

[Formatting Date and Time](#)

[Creating a Custom Date and Time Format](#)

[Formatting Enumerations](#)

## **[23 Multithreaded Programming](#)**

[Multithreading Fundamentals](#)

[The Thread Class](#)

[Creating and Starting a Thread](#)

[Some Simple Improvements](#)

[Creating Multiple Threads](#)

[Determining When a Thread Ends](#)

[Passing an Argument to a Thread](#)

[The IsBackground Property](#)

[Thread Priorities](#)

[Synchronization](#)

[An Alternative Approach](#)

[The Monitor Class and lock](#)

[Thread Communication Using Wait\(\), Pulse\(\), and PulseAll\(\)](#)

[An Example That Uses Wait\(\) and Pulse\(\)](#)

[Deadlock and Race Conditions](#)

[UsingMethodImplAttribute](#)

[Using a Mutex and a Semaphore](#)

[The Mutex](#)

[The Semaphore](#)

[Using Events](#)

[The Interlocked Class](#)

[Terminating a Thread](#)

[An Abort\(\) Alternative](#)

[Canceling Abort\(\)](#)

[Suspending and Resuming a Thread](#)

[Determining a Thread's State](#)

[Using the Main Thread](#)

## **[24 Collections, Enumerators, and Iterators](#)**

[Collections Overview](#)

[The Non-Generic Collections](#)

[The Non-Generic Interfaces](#)

[The DictionaryEntry Structure](#)

[The Non-Generic Collection Classes](#)

[Storing Bits with BitArray](#)

[The Specialized Collections](#)

[The Generic Collections](#)

[The Generic Interfaces](#)

[The KeyValuePair<TK, TV> Structure](#)

[The Generic Collection Classes](#)

[Storing User-Defined Classes in Collections](#)

[Implementing IComparable](#)

[Implementing IComparable for Non-Generic Collections](#)

[Implementing IComparable<T> for Generic Collections](#)

[Using an IComparer](#)

[Using a Non-Generic IComparer](#)

[Using a Generic IComparer<T>](#)

[Accessing a Collection via an Enumerator](#)

[Using an Enumerator](#)

[Using the IDictionaryEnumerator](#)

[Implementing IEnumerable and IEnumerator](#)

[Using Iterators](#)

[Stopping an Iterator](#)

[Using Multiple yield Directives](#)

[Creating a Named Iterator](#)

[Creating a Generic Iterator](#)

[Collection Initializers](#)

## **[25 Networking Through the Internet Using System.Net](#)**

[The System.Net Members](#)

[Uniform Resource Identifiers](#)

[Internet Access Fundamentals](#)



- [High Exposure: An Enduring Passion for Everest and Unforgiving Places pdf, azw \(kindle\), epub](#)
- [Rats: Observations on the History and Habitat of the City's Most Unwanted Inhabitants pdf, azw \(kindle\), epub, doc, mobi](#)
- [read online Justice and the Politics of Difference](#)
- [download online The Routledge Handbook of Stylistics book](#)
- [read online The First World War: A Concise Global History \(2nd Edition\) \(Exploring World History\) for free](#)
- **[download online Become](#)**
  
- <http://musor.ruspb.info/?library/High-Exposure--An-Enduring-Passion-for-Everest-and-Unforgiving-Places.pdf>
- <http://betsy.wesleychapelcomputerrepair.com/library/Rats--Observations-on-the-History-and-Habitat-of-the-City-s-Most-Unwanted-Inhabitants.pdf>
- <http://damianfoster.com/books/Justice-and-the-Politics-of-Difference.pdf>
- <http://junkrobots.com/ebooks/The-Routledge-Handbook-of-Stylistics.pdf>
- <http://transtrade.cz/?ebooks/Commodify-Your-Dissent--Salvos-from-The-Baffler.pdf>
- <http://betsy.wesleychapelcomputerrepair.com/library/Become.pdf>