Every NeXTSTEP application is built using the same fundamental features: object-oriented programming, kits of predefined functionality, and an object-oriented graphical development environment.

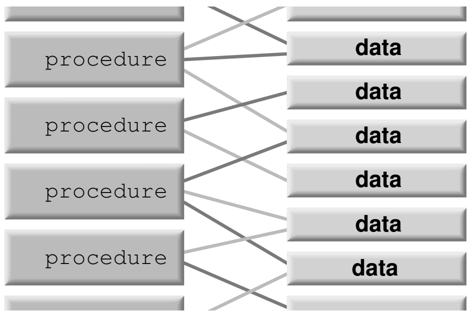
This proven foundation helps make all NeXTSTEP programs more robust—and NeXTSTEP programmers more productive.

In this section, you'll see how the basic features of NeXTSTEP—individually and in concert—provide a solid foundation for your programming efforts.

OBJECT-ORIENTED PROGRAMMING

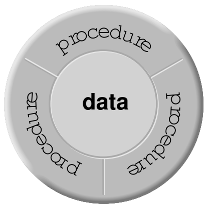
"Object-oriented programming" has become a first rank buzzword within the computer industry. To understand why, it's important to cut through the hype and focus on the problem from which the object-oriented approach sprang.

In classic *procedural programming* (used with COBOL, Fortran, C, and other languages), programs are made of two fundamental components: *data* and *code*. The data represents what the user needs to manipulate, while the code does the manipulation. To improve project management and maintenance, code is compartmentalized into *procedures*. However, much of the data is global, and each procedure may manipulate any part of that global data directly.

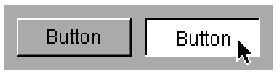


With the procedural approach, the network of interaction between procedures and data becomes increasingly complex as an application grows. Inevitably, the interrelationships become a hard-to-maintain tangle—spaghetti code. Procedural programming also leads to nasty, hard-to-find bugs in which one function inadvertently changes data that another function relies on.

The reasoning behind object-oriented programming is simple. Just as procedures compartmentalize code, objects compartmentalize both code *and* data. This results in *data encapsulation*, effectively surrounding data with the procedures for manipulating that data.



Like objects in the physical world, objects in a program have identifying characteristics and behavior. For example, an object such as a button includes the data and code to generate a familiar appearance on the screen, and a familiar response to user action.



A button object highlights its on-screen representation when the user clicks

Similarly, an object representing a database record both stores data and provides well-defined ways to access that data.

Using this *modularity*, object-oriented programs can be divided into distinct objects for specific data and specific tasks. Programming teams can easily parcel out areas of responsibility among them, agreeing on interfaces to the distinct objects while implementing data structures and code in the most efficient way for their specific area of functionality.

SOME OBJECT-ORIENTED PROGRAMMING TERMINOLOGY

While object-oriented programming introduces new vocabulary to computer programming, much of it simply restates and extends concepts familiar to most programmers. As the text illustrates, object-oriented programming itself simply extends procedural programming to code and data. Here are some other object-oriented terms you'll encounter throughout this guide.

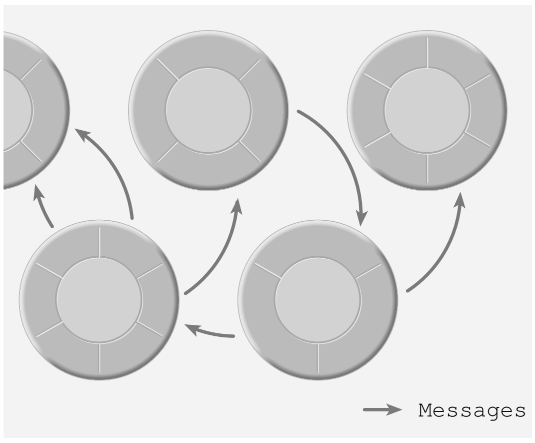
A *class* is a template for a particular type of object. Much as you can define types of data structures in procedural programming languages such as C and Pascal, you define classes of objects in object-oriented programming. The class defines both procedures and data for a particular object type.

An *object* is a specific instance of a particular class, both data and the code to operate on that data. Much as you create instances of data structures in procedural programming, you create objects in object-oriented programming.

A *method* is a procedure defined for an object by its class. An object's methods can operate directly on its data.

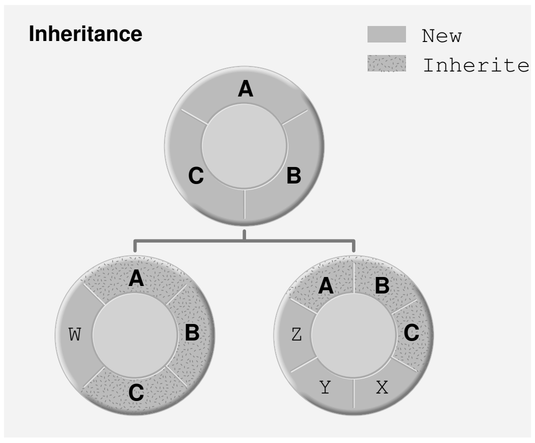
A *message* tells an object to perform one of its methods. Sending a message to an object is analogous to invoking a procedure on a particular data structure in procedural programming.

Data encapsulation and modularity result in simplified program structure. An object-oriented program can be thought of as a network of objects that interact by sending messages to one another.

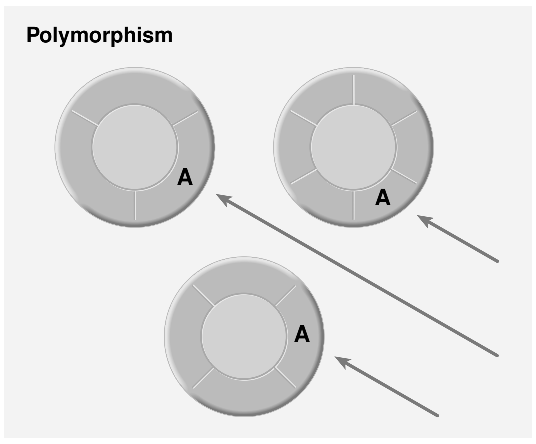


Once objects are defined, creating a program is a matter of creating connections for objects to use in their interaction. Simpler structure means simpler debugging, since errant behavior can be traced directly to the responsible object.

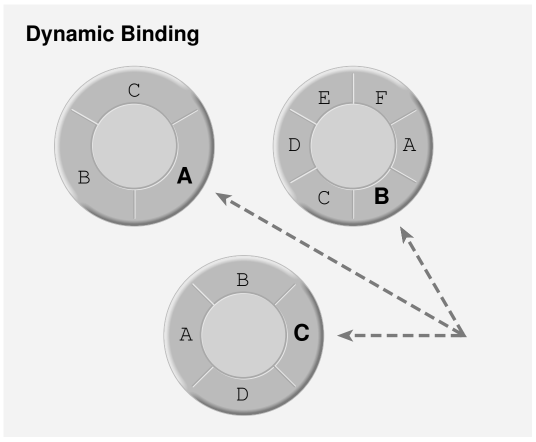
Other important benefits of object-oriented programming with Objective C include *inheritance*, *polymorphism*, and *dynamic binding*.



Inheritance lets you define new classes of objects that build on the behavior of existing classes, reducing the amount of code you write and debug to make incremental changes.



Polymorphism means that different classes of objects respond to the same message in their own ways—effectively increasing program flexibility while maintaining code simplicity.



Dynamic binding means that both the object receiving a message and the message that an object receives can be set within your program as it runs. This is particularly important in a graphical, user-driven environment, where one user command—say Copy or Paste—may apply to any number of user-interface objects.

Thanks to the object-oriented approach, NeXTSTEP applications are easier to design, easier to code, easier to debug, and easier to update than procedural programs.

KITS OF READY-TO-USE PROGRAM FEATURES

One guiding principle behind NeXTSTEP is this: The only way to significantly reduce the time it takes to write an application is to significantly reduce the volume of code you have to write.

When you create an application, there are almost always standard features you need. For example, an application with a graphical user interface is usually built around the framework of an *event loop*. The event loop is the code that receives user-generated events—mouse clicks, keystrokes—and invokes procedures in response to those events. Other standard features required by applications include text editing, file opening and saving, and data exchange with other applications.

In most programming environments, you have to recode these standard features each time you create a new application. But the NeXTSTEP environment provides kits of ready-to-use objects that implement exactly those features required by most applications.

NeXTSTEP's Application Kit‰ is a set of objects that implement common features such as event handling, window management, text editing, file management, cutting and pasting between applications, and more. The goal of the Application Kit is to limit your programming task to designing the user interface and coding your application's unique features.

In addition to the Application Kit, NeXTSTEP includes other object-oriented kits and libraries that provide useful features. For example, objects in the Database Kit‰ provide access to industry-

THE APPLICATION KIT: ESSENTIAL FEATURES FOR ANY APPLICATION

To reduce the amount of new code you write for each application, NeXTSTEP's Application Kit implements a number of common features useful to any application.

Graphic User Interface Framework

- Event-handling mechanism for mouse, keyboard, system, and custom events
- Window management
- Buttons, sliders, text fields, and other standard user interface components

Standard Program Features

- Text editing and formatting
- Printing and faxing from any document
- File management
- Spell-checking
- Font selection
- Color selection
- Image handling for TIFF, EPS, RIB, and custom formats

Interapplication Communication Features

- Standard pasteboard for text, fonts, images, paragraph formats and tab settings, and more
- Custom pasteboards for application-specific data types
- Dynamic data links between documents to enable automatic updates as sources are edited
- Mouse-controlled drag and drop of text, images, and other data within and between applications
- Dynamic Services menu that lets applications perform useful operations on one another's data

standard databases from Oracle, Sybase, and other vendors. The Indexing Kit provides objects that can index text and other data for quick access by key values. The 3D Graphics Kit‰ lets you add 3D imaging to an application with minimum effort.

Ready-to-use kits demonstrate the power of object-oriented programming and minimize the time and effort required to code a NeXTSTEP application.

GRAPHICAL DEVELOPMENT ENVIRONMENT

With its object-oriented programming model and standard application framework, NeXTSTEP provides an ideal platform for easy-to-use programming tools. Two tools—Project Builder and Interface Builder‰—are used to create your application and manage its source files. A third tool, Header Viewer, provides quick, object-oriented access to developer documentation and system header files. Together, these tools simplify and expedite application development.

NEXTSTEP KITS AND LIBRARIES: A WIDE RANGE OF ENHANCED FEATURES

In addition to the Application Kit, NeXTSTEP offers other object-oriented kits and libraries to enhance program functionality.

Database Kit

- Fast, efficient development of client-side applications fully integrated with other NeXTSTEP applications
- User interface objects for data display, formatting, and editing
- Includes adaptors for ORACLE‚ and SYBASE‚ servers

Indexing Kit‰

- Fast, indexed access to a variety of data types: text, objects, files, images, and others
- File system management, text parsing, and query processing

Distributed Objects

- Object-oriented model for peer-to-peer and client-server application development.
- Simplifies network-wide interapplication communication

3D Graphics Kit

- Photorealistic and Interactive RenderMan‰ 3D graphics
- Fully integrated with NeXTSTEP for display, printing, and faxing

Project Builder is a graphical tool for project management and source file control. As you create your application, Project Builder keeps track of the required resources. When you need to build your application, Project Builder compiles only those source files that have been updated, using the UNIX-standard make program. And as your project is compiled, Project Builder provides a status update, displays warnings and error messages, and gives instant access to source files for quick error correction.

Project Builder also provides a convenient way to access other application development tools, including Interface Builder, the text editor, the compiler, the debugger, and others.

Interface Builder actually serves three main purposes. First, it's an interactive design tool for composing windows, text fields, buttons, and other objects making up a NeXTSTEP user interface. Second, it's an object editor that lets you specify objects and their behavior. Third, it's an object "connector" that lets you graphically stitch together the network of objects that makes up a NeXTSTEP application. Additionally, Interface Builder provides an interactive Test mode that lets users try out and comment on the user interface as it develops.

Header Viewer is a NeXTSTEP development tool designed to support the specific requirements of object-oriented coding. Header Viewer provides class browsing, query and search for specific classes and methods, and easy access to the system headers and developer documentation for all NeXTSTEP kits.

Thanks to the power provided by these tools—and the power of the underlying programming model—NeXTSTEP offers a more direct route from design to deployment than other environments.