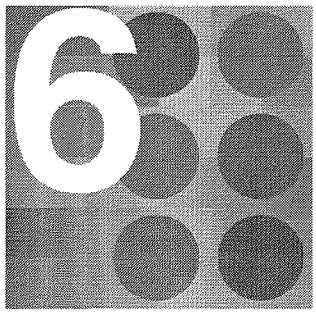
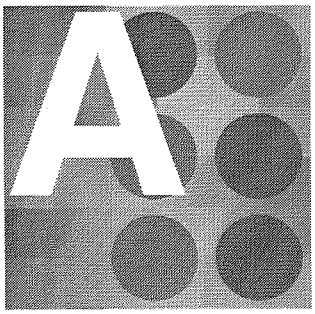


Creating Views

- **Topic A:** Create Views
- **Topic B:** Add View Columns
- **Topic C:** Organize Documents
- **Topic D:** Format Views



Topic A: Create Views

After completing this topic, you should be able to:

- ✓ Identify the characteristics of and uses for views.
- ✓ Describe the process of designing and building a view.
- ✓ List the steps for creating a view.
- ✓ List guidelines for setting initial view properties.
- ✓ Identify selection conditions.

The View Design Element

The purpose of the **View** design element is to create an organized list of documents so that users can find the information they need. Views are one of the entry points to the documents stored in an IBM® Lotus® Domino® database. All databases contain at least one view. However, most databases contain more than one view.

The following figure represents the View design element.

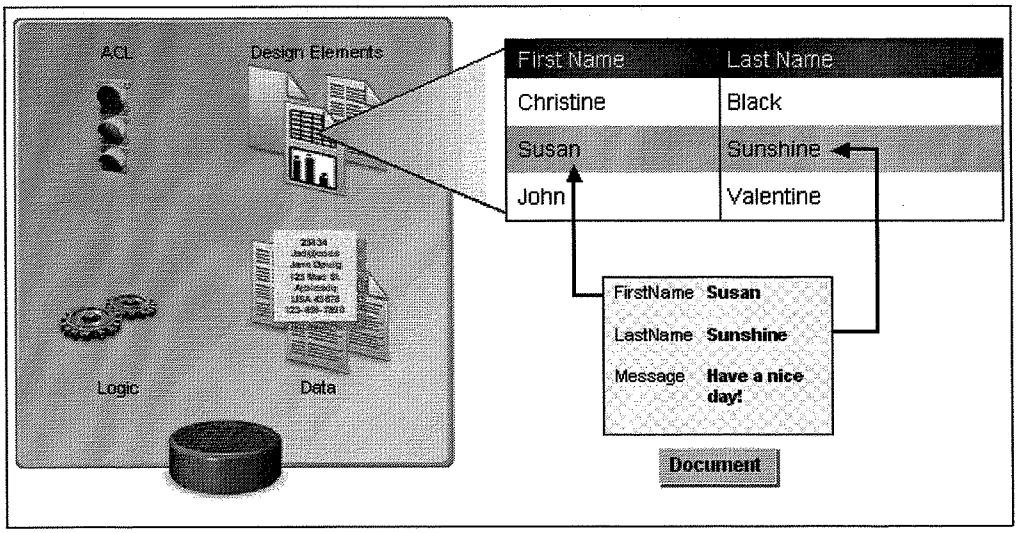


Figure 6-1: The View design element

Characteristics of a view

The following list describes some of the characteristics of views:

- A view is a list of documents in a database.
- Each row in a view represents a document in the database.
- Each column displays information from or about the document in that row.
- A view may include all documents or a subset of documents based on selection criteria.
- Users read the information in the view to locate specific documents.
- Excessive numbers of views and poorly designed views can have a serious impact on database performance.
- Views should be designed early in the application development process.

How Lotus Domino stores a view

Lotus Domino uses an internal filing system called the view index to store the list of documents in a view. End users and designers never work directly with the view index. It is built and maintained by the Lotus Domino application.

The following table describes how Lotus Domino builds a view.

Stage	Description
1	The application developer creates the view design.
2	Lotus Domino pulls information from documents according to the design and builds a view index.
3	Lotus Domino displays the information to users.

The following figure shows how Lotus Domino builds a view.

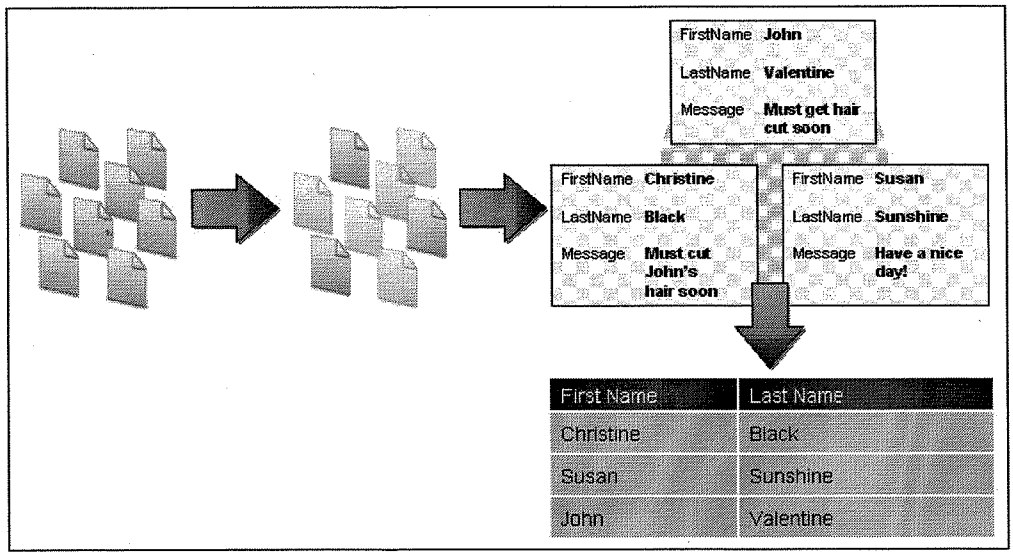


Figure 6-2: How Lotus Domino builds a view

How Lotus Domino builds a view collection and index

The following figure illustrates how Lotus Domino builds a view collection and index.

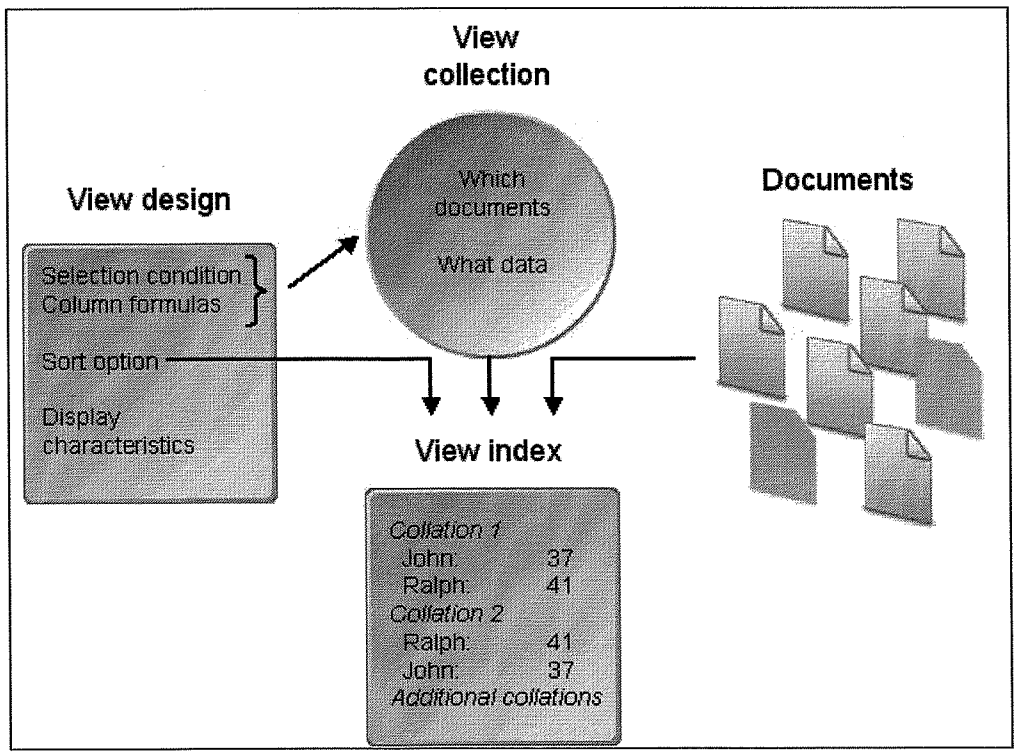


Figure 6-3: How Lotus Domino builds the collection and index

The Notes Index Facility

The Notes Index Facility (NIF) resides within a Lotus Domino database. The NIF is responsible for all tasks that keep documents ordered within each database view. Some of the NIF functions include:

- Open and close collections.
- Update the index.
- Locate notes within an index.
- Locate index entries.
- Update collections.

Minimizing the number of views

Each view builds its own collection and related index, with all the collations or sorts you have specified. When preparing a view for display to a client, keep in mind that it is much faster for Lotus Domino to reorder a collection that is already open (different sorts in same view) than to open and order a different collection (different views).

Use the **Click on column header to sort** property to minimize the number of views and maximize the sorting within those views.

The following figure shows the Click on column header to sort property.

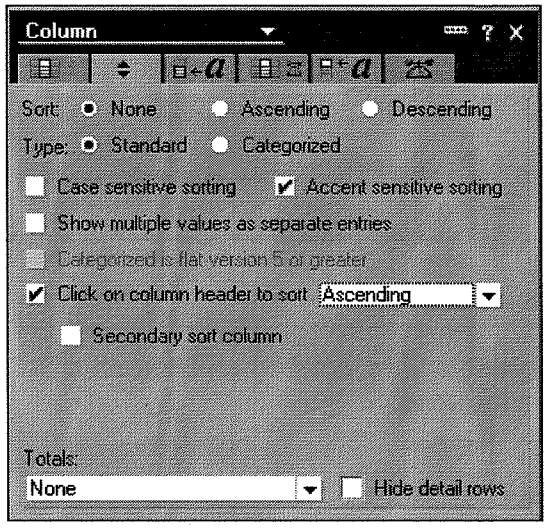


Figure 6-4: The Click on column header to sort property

View Design

Effective views allow users to find information quickly and easily. As a developer, you must understand the needs of the users. When designing a view, consider the following:

- Which documents to include in the view
- What information to display in the columns of the view
- How to sort and/or categorize the documents
- Whether the view will be for shared or private use
- How to format and display the information

Designing a view

Complete these tasks to design and build a view.

Task	Procedure
1	Create the view.
2	Select documents for the view.
3	Add columns to the view.
4	Sort and categorize the documents.
5	Format the view display.  **Note:** These tasks will be discussed in more detail later in the lesson.

View Creation

When designing a view, start by creating an initial view design. The initial view design includes:

- Creating the view
- Naming the view
- Determining the view type
- Controlling how the view will appear in the view list
- Setting the view selection conditions

Creating a View

Follow these steps in Lotus Domino Designer to create a new view.

Task	Procedure
1	In the **Design** pane, click **Views.**
2	In the **Work** pane, click **New View.**
3	In the **Create View** dialog box, set the initial view properties. Click **OK.** **Result:** Lotus Domino creates the view and adds it to the list in the Work pane.

Initial View Properties

The following are guidelines for setting the initial view properties.

Property	Guidelines
View name	Use a short, descriptive name indicating how the view organizes documents.
Type	● Select **Shared,** to create a view many users will access. ● Select **Shared, contains documents not in any folders,** if the user typically stores documents in folders and wants to find all others quickly. ● Select **Shared, contains deleted documents** for users to drag out of the trash to a folder, or undelete, if the soft deletions option has been activated. ● Select **Private** to create a personal view for a single user. ● Select **Shared, private-on-first-use views** to distribute views that begin as shared, but change to private once a user has accessed and saved the view. **Note:** You cannot change the type once you create a view.
Location	● Use up to two levels in a hierarchy. ● Group views that include the same documents together. ● Create a cascaded list of views. **Note:** Entering a backslash (\) after a view's name with a new name accomplishes the same cascaded view menu.
Copy style from	Use the design of another view in the current database as a template.

Property	Guidelines
Selection conditions	• Leave the condition blank to include all documents in the database. • Click **Add Condition** and use the **Search Builder** to specify criteria for including a subset of documents in the view. • Change the selection conditions after you create the view.

Note: It is better to have one well-designed view than a dozen poorly-designed views.

View Aliases

Using an alias is a good habit to practice. Design elements such as views can be referred to by an alias when you reference them in a formula or any kind of programming. If a design element has an alias and that alias is used in any coding, the name of the design element can be changed without the need to also change formulas or code.

Creating an Alias for a View

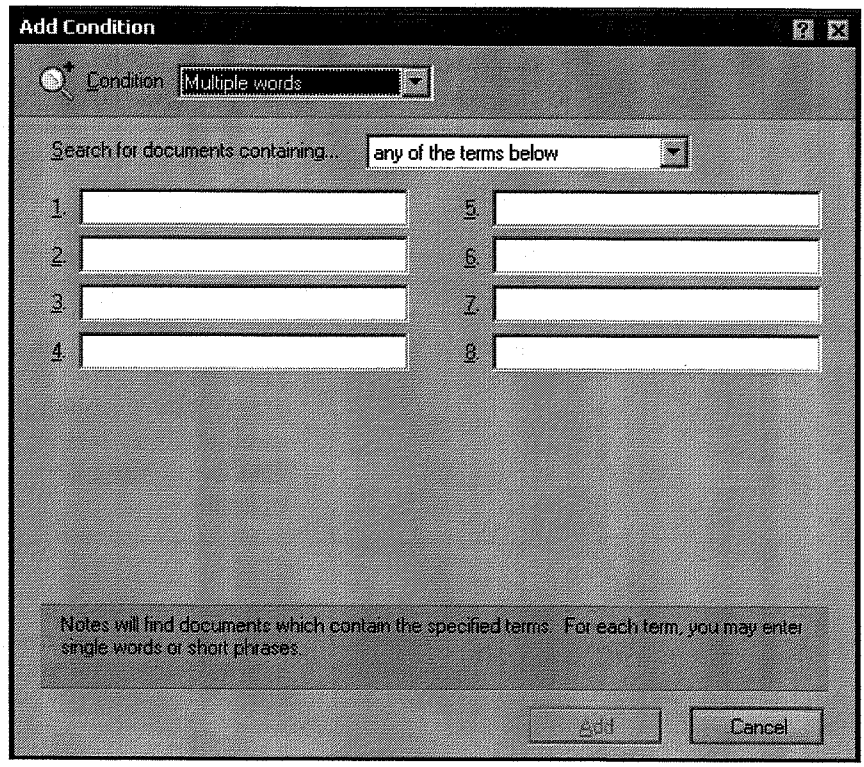
Follow these steps to set properties after creating a new view.

Task	Procedure
1	Open the view in Lotus Domino Designer.
2	Open the **View** Properties box.
3	On the **Basics** tab, enter an **Alias** and **Comment.**

Adding selection conditions to a view

Add selection conditions to control which documents a view includes.

Follow these steps to add selection conditions to a view.

Task	Procedure
1	Open the view in Lotus Domino Designer.
2	In the **Object** list of the Programmer's pane, click **View Selection.**
3	In the **Infolist** of the Programmer's pane, ensure that the default is **Simple Search.** **Result:** An Add Condition button is added to the bottom of the script area in the Programmer's pane.
4	In the Script area, click Add Condition. **Result:** The Add Condition dialog box opens. 
5	In the **Add Condition** dialog box, create a condition. Click **Add.**
6	Refresh the view index in Lotus Domino Designer. **Result:** Lotus Domino Designer rebuilds the view index with the new selection condition.

Specifying conditions in the Add Condition dialog box

The following table describes the types of conditions you can specify in the Add Condition dialog box.

Use the Condition...	To Find Documents...
By Author	Whose author is or is not a specified user or users. Documents must have an Authors field.
By Date	That were created or modified: ● **On, after, before**, or **not on** a specified date ● **In the last, in the next, older than**, or **after the next** specified number of days ● **Between** or **not between** a specified range of dates
By Field	That have a field that **contains** or **does not contain** a specified value.
By Form	Whose form field contains the specified form name.
Fill out example form	Whose form field contains the specified form name and whose fields have matching conditions. Only fill the fields that are relevant to your search.
In Folder	In the specified folder.
Multiple Words	That have any, or all, of up to eight terms, words or short phrases. **Note:** This condition uses significant system resources and can adversely impact the response time of a large database. Use it sparingly, and only if **Fill out example form** is not suitable.

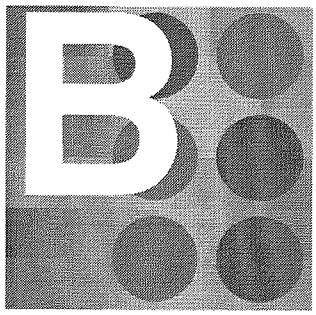
Note: Add Condition is a tool used mainly by end users to simplify setting view conditions. It does not replace formulas. Developers should use the SELECT statement as it is more efficient. Formula building will be discussed later in the course.

Activity 6-1: Create a view with an alias

In this activity, you will first create a view that is limited to documents created using the Employee Information form. Follow these steps to create the view and limit the documents in the view.

Step	Action
1	If necessary, open your copy of the **Practice** database.
2	Click **Views** in the Work pane.
3	Click **New View.** **Result:** The Create View dialog box opens.
4	Set the following characteristics: ● **View Name:** `Employee Information` ● **View Type:** Shared ● **Location:** Views ● **Copy Style from View:** Blank (default) ● **Selection conditions:** Leave blank Click **OK.** **Result:** A new blank view is created and displayed in the listing in the Work pane.
5	Double-click to open the view you just created. **Result:** A blank view opens in the Work pane; and the **View Properties** box opens.
6	In the **View** Properties box, on the **View Info** tab, type the following: ● **Alias:** `ei` ● **Comment:** `This is the first practice view.`
7	Close the **View** Properties box.
8	On the **Objects** tab, click **View Selection.**
9	In the **Infolist** of the Programmer's pane, ensure that **Simple Search** is selected.
10	In the **Script** area, click the **Add Condition** button.
11	In the **Add Condition** dialog box, select the following: ● **Condition:** By form ● **Search for documents which use form:** Employee Information Click **Add.** **Result:** The condition appears in the Programmer's pane.
12	Preview your changes. Click **Yes** to save the view. **Result:** Only documents created using the Employee Information form should appear.

Step	Action
13	Close the view. **Result:** Your new view, its alias and comment appears in the view list in the Lotus Domino Designer window.



Topic B: Add View Columns

After completing this topic, you should be able to:

✓ Describe view columns.

✓ Describe the types of values that view columns can display and what options to use to specify each.

✓ List ways to format a view column.

View Columns

Use columns to display information about the documents in the view.

When you create a view using the Blank option, Lotus Domino automatically creates one column in the view. This column displays row numbers corresponding to the documents in the view. You can modify this column, as well as insert and append new columns.

When creating a column, do the following:

- Add a column to the view.
- Specify a value for the column to display.
- Format the view column.
- Format the view column values.

Adding a new column to a view

You can insert a column to the left of the current column, or append a column to the right of the last column in the view.

Follow these steps to add a column to a view.

Task	Procedure
1	Click an existing column.
2	To add a column to the **left** of the one selected, choose **Create→Insert New Column.** To add a column to the **right** of the right-most column, choose **Create→ Append New Column** from the menu.

View Column Values

The following table describes the three types of values that a view column can display.

Type of Value	Purpose
Field	Displays the value stored in a specified field for each document in the view. **Note:** View columns cannot display information from computed for display, rich text, and encrypted fields.
Simple Function	Returns information about documents that Lotus Domino stores automatically. For example, simple functions can return: ● A list of attachments in a document ● The size of the attachments ● The name of the document's author ● The creation date of the document
Formula	Displays multiple field values in a single column or creates new values using the Formula language. **Note:** Attempting to access data from computed for display, rich text, or encrypted fields in a view formula will result in a null value.

Specifying view column values

After creating a column, specify the information to display in the column. Refresh the view after selecting the value to update the view index and see the change.

The following table explains the three methods for displaying data in view columns.

To Display...	Select...	And...
A single field value stored in the document	Field	Click a field from the list. **Note:** Fields that cannot be used in a view do not appear in the list
Information about the document not contained in fields	Simple Function	Click a function from the list.
Multiple fields in a single column or new values	Formula	Write a formula.

Displaying multiple fields in a single view column will be discussed later in the course.

View Column Formatting

To format view columns, set properties in the Column Properties box. The Column Properties box has many options available to modify and organize the view columns. You will begin with modifying individual columns.

The following table lists some column formatting options.

In This Tab...	Set Properties to...
Column Info	Specify general display characteristics, including: ● Column title ● Width
Title	Specify text formatting options for the column title, including: ● Font ● Size ● Style Note that you can change all existing column titles by clicking the **Apply to all** button.
Advanced	● Specify a name for the column to use when accessing the column programmatically. This is similar to giving a form or view an alias name. ● Indicate if the values in the column should appear as links when the view appears on the Web.

Formatting view column values

To specify how a column displays its value, set properties in the Column Properties box according to the following table.

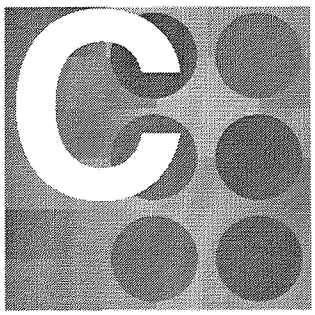
In This Tab...	Set Properties to...
Font	Specify the font, size, and style of text for displaying the column value.
Numbers	Specify special options if the column value is a number.

In This Tab...	Set Properties to...
Date and time format	Specify special options if the column value is a date-time value. **Note:** Date/time values always contain both a date and a time, because of how they are stored internally.

Activity 6-2: Add and format view columns

In this activity, you will modify the Employee Information view to show employees' names, departments, and salaries. Complete the following steps.

Step	Action
1	If necessary, in your copy of the **Practice** database, open the **Employee Information** view in Lotus Domino Designer. **Result:** The view opens in the Work pane.
2	Select the first column header, and select **Display Field** in the Programmer's pane. **Result:** The Programmer's pane displays all the fields available in the Practice database.
3	Choose the **LastName** field.
4	Double-click the column header to open the **Column** Properties box, and type the title: `Last Name` **Result:** The column title changes to Last Name.
5	Insert a column to the left of the **Last Name** field with the following features: ● **Field:** FirstName ● **Column title:** First Name **Result:** The First Name column appears to the left of the Last Name column.
6	Append a column to the right of the **Last Name** column with the following features: ● **Field:** Department ● **Column title:** Department **Result:** The Department column appears to the right of the Last Name column.
7	Append a column to the right of the **Department** column with the following features: ● **Field:** Salary ● **Column title:** Salary **Result:** The Salary column appears to the right of the Department column.
8	In the **Column Properties** box, select the **Title** tab. Set **Center, Bold,** and **Apply to all.** **Result:** All the view column titles are centered and bold.
9	Close the **Column Properties** box.
10	Preview your changes.



Topic C: Organize Documents

After completing this topic, you should be able to:

✓ Identify how documents can be sorted.

✓ Describe document categorization.

Document Organization

Once you have added columns to display relevant information for the users, organize the list of documents so users can locate a specific document quickly. The sorting tab in the Column Properties box provides several different ways to organize documents in a view.

There are two types of sort options you can choose:

- **Standard:** Sorts documents by the column value in ascending or descending order.
- **Categorize:** Groups together documents with the same column values.

When you sort or categorize a column, you are changing the order in which the view displays the documents to users.

Sorted Columns

A sorted column displays documents in a specific order. The following table describes how a view sorts different types of values.

Columns That Display...	Sort Documents...
Text values	Alphabetically
Number values	Numerically
Date/time values	Chronologically

If the column being sorted contains text, there are two other options that could be useful:

- **Case-sensitive sorting:** Sorts lowercase letters before uppercase letters. For example, "ab" sorts before "Aa."
- **Accent-sensitive sorting:** Sorts accented characters after non-accented characters. For example, "ab" sorts before "äa."

Sorting documents by column value

Follow these steps to sort documents based on the contents of a column.

Task	Procedure
1	Select the column you want to sort.
2	Open the **Column** Properties box.
3	Click the **Sorting** tab.
4	Select **None, Ascending,** or **Descending.**
5	Set optional properties to refine the sort: ● Case-sensitive sorting ● Accent-sensitive sorting ● Click column header to sort

User-sorted columns

The sort column feature allows end users to click on the column header to change the column's sort order. This option is available in the Sorting tab of the Column Properties box. The following table describes the different end user sort control options.

Option	Function
Ascending	An upward-pointing triangle appears in the column header. Users click the header to sort in ascending order or revert to the original order.
Descending	A downward-pointing triangle appears in the column header. Users click the header to sort in descending order or revert to the original order.
Both	A two-headed triangle appears in the column header. Users click the header to cycle between ascending, descending, and the original sort order.
Secondary Sort Column	Designate a second column to sort the view further. For example, if the primary sort column is last name, the secondary could be the first name. **Note:** This column can be sorted in either ascending or descending order only.

Activity 6-3: Set sort columns

In this activity, you will modify the Employee Information view so that users can sort the view by last name, then first, if necessary. Follow these steps to create user-sorted columns.

Step	Action
1	If necessary, in your copy of the **Practice** database, open the **Employee Information** view in Designer. **Result:** The view opens in the Work pane.
2	Double-click the **Last Name** column to open the **Column** Properties box. **Result:** The Column Properties box opens.
3	Click the **Sorting** tab and make the following selections: **Click on column header to sort:** Both **Secondary sort column:** First Name **Order:** Ascending **Result:** The Last Name column has a two-headed arrow in the column title header.
4	Preview your changes.

Categorizing

A categorized column groups related documents together in a view based on the column value. The following figure shows a categorized view.

Last Name	First Name
▼ Manufacturing	
Bennett	Lyn
Polanski	Annie
▼ Research and Development	
Smith	Elmer

Figure 6-5: A categorized view

Categorizing documents by column value

Always categorize the left-most columns of the view. Typically, you will insert a new column to group related documents together in the view.

Follow these steps to categorize a view.

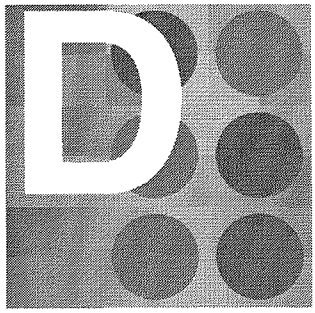
Task	Procedure
1	Select the column whose value will categorize the documents.
2	Open the **Column Properties** box.
3	On the **Sort** tab, select **Categorized.**

Guidelines for formatting categorized columns

A view displays the column value of a categorized column in its own row.

The following table lists some best practice guidelines to follow, and their explanations.

Guideline	Why
Do not use a title for the column.	The column values appear in the view and should be self-explanatory. A column title would be redundant. However, there are instances where a column title is needed, such as when you need to display more than one date.
Set the width of the column to 1.	This helps save room in the view to see more data.
Show twisties when the row is expandable.	This simplifies expanding and collapsing different categories for end users.
Use bold and color to make the categorized column value stand out from other information in the view.	This helps the user differentiate between the category headers and the documents.



Topic D: Format Views

After completing this topic, you should be able to:

✓ List ways to format views.

View Properties

Use the View Properties box to format the view.

The following table describes some of the properties in the View Properties box.

Tab	Set Properties to...
Options	● Open the view by default when the database opens. ● Specify this view as the default for all new views. ● Specify how the view displays the documents when it opens.
Style	● Apply color to the view's background. ● Apply an image to the view's background. ● Display alternate rows in different colors. ● Display totals in the column. ● Set row spacing.
Advanced	● Determine when Domino refreshes the view index. ● Set colors for links in a Web browser.
Security	● List users who will use the view in the database. **Note:** This option does not prevent users from seeing data in the documents displayed in the view. Users could select to display the same documents in another view or folder that they create.

Practice Activity 6-4: Create a Categorized View

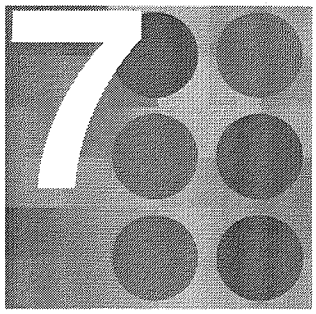
Scenario: Worldwide Corporation wants to provide a new way of organizing its policies and procedures in the Policies and Procedures application. The view should do the following:

- List documents created with the **Policy** form.
- Display the title of the policy and its effective date (in that order).
- Organize the documents by the type of policy (Guidelines, Holidays, and so on).

Perform the following tasks to complete this activity.

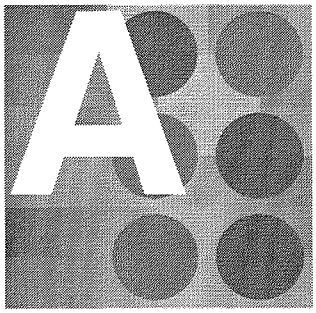
Note: The guided solution to this activity can be found in Appendix A, Solutions to Practice Activities.

1. Build a view according to the requirements stated above.
2. Name the view **Policies.**
3. Test the view in Lotus Notes.
4. Close all open windows.



Introducing Formulas

- **Topic A:** Identify Programming Languages Supported by Lotus Domino
- **Topic B:** Identify Formula Components
- **Topic C:** Identify Event Triggers
- **Topic D:** Compute and Compare Values
- **Topic E:** Work with @Functions
- **Topic F:** Explore Formatting and Comments



Topic A: Identify Programming Languages Supported by Lotus Domino

After completing this topic, you should be able to:

- ✓ List options for programming in the Lotus Domino environment.
- ✓ Identify where to use the Formula language.
- ✓ Identify where to use LotusScript.
- ✓ Identify where to use Java.
- ✓ Identify where to use JavaScript.
- ✓ Identify criteria for choosing a programming language.

Lotus Domino Programming

You can write programs in an IBM® Lotus® Domino® application using the IBM® Lotus® Domino Designer® client. Many design elements can execute programs automatically through the properties and events of the element.

You can also write programs outside the Lotus Domino application using other development tools. These programs can be imported or run in the operating system to access the Lotus Domino application.

Lotus Domino is a flexible development environment

The Lotus Domino development environment supports all leading technologies with tools that cater to varying developer skill levels and degrees of programming experience. Given so many choices, the question that confronts developers at the outset of any application development initiative is which tool to use. The answer is simple: understand the functions that can be performed programmatically in Lotus Domino, then pick the one that meets the demands of the task at hand and is most comfortable for you. Some factors that affect the decision include:

- Does the code run on a server?
- Does the code run on a client? If so, which client?
- Where does the data reside?
- What are the programming skills of the developer?
- What functionality is required?

Just as important as where the code is written is where it is executed. The location of execution, namely, server versus client, can affect how the data is accessed, as well as the security provisions that are taken to protect the data.

Where to program in Lotus Domino

The following figure shows where code can be written for the Lotus Domino environment.

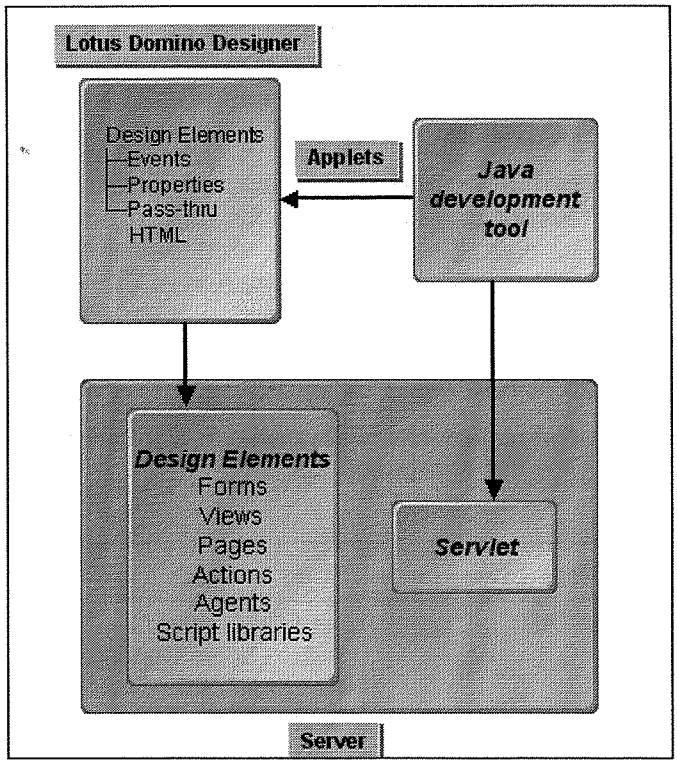


Figure 7-1: Where code can be written for Lotus Domino

Internal programming languages

Lotus Domino Designer provides an Integrated Development Environment (IDE) for writing code in a Lotus Domino application. It supports the following four programming languages:

- Formula
- LotusScript
- Java
- JavaScript

Lotus Domino also supports HTML and XML.

External programming languages

Lotus Domino supports the following external programming languages:

- Java
- C and C++
- Microsoft Visual Basic®

Connecting to Lotus Domino from external applications

One of the new features in Lotus Domino 7 is the ability to store, access, view, and manage application data in a robust, enterprise class DB2 relational data store. When working with the data, the interaction is entirely with the Lotus Domino server within the IBM® Lotus Notes® client. The Lotus Notes user is unaware that the storage is in DB2 and not in an NSF file. There is no need for users to have DB2 connectivity as that is handled by the Lotus Domino server.

Users can query and report on Lotus Domino data using traditional Structured Query Language (SQL) based tools and combine Lotus Domino data with DB2 data. Customers can continue to choose between native Domino NSF or DB2 stores or a combination of the two, depending on the requirements of each application. When working with the data, there is no visible difference between the data stored in a Notes database and the data stored in a DB2 database.

You can also use one of the drivers the following table describes to query Lotus Domino databases via SQL.

This Driver	Is...	Use It When You Require...
NotesSQL	An ODBC driver that makes Lotus Domino databases accessible to an SQL tool or application interface.	Read/write access to Lotus Domino data using any application that supports ODBC. For example, an application that produces reports in Lotus Notes using Business Objects' Crystal Reports® or Microsoft Access.
Lotus Domino Driver for JDBC™	A Type II JDBC driver that makes Lotus Domino databases look like another relational back-end source to an SQL tool or application interface.	Access to Lotus Domino data using any JDBC-enabled application. For example, when your application is a servlet on an HTTP server.

The Formula Language

The **Formula language** is the core language built into Lotus Notes and Lotus Domino from their earliest releases.

Where to use the Formula language

The Formula language provides a straightforward and simple programming interface within Lotus Domino applications. Like other programming languages, the Formula language has rules and syntax. Formula language code is generally best used for working with the element that the user is currently processing.

Use the Formula language to customize the forms, fields, and views of a Lotus Domino application. The following table lists common events and properties that execute formulas.

Element	Property/Event
Form	Window Title WebQuerySave PostOpen (can also execute LotusScript)
Field	Value or Default Value Input Validation Input Translation Hide/when
View	View Selection Form Formula Column Formula

LotusScript

LotusScript is an embedded BASIC scripting language that offers access to Lotus Domino data and services beyond the capabilities of the Formula language. It is much more efficient than Formula language for accessing multiple documents and performing operations on those documents.

Where to use LotusScript

Use LotusScript to program agents and actions or to execute code in a design element event. The following table lists common events that execute LotusScript.

Element	Property/Event
Form	PostOpen (can also execute formulas) Queryclose PostRecalc

Element	Property/Event
Field	Initialize Terminate
Agent	Initialize Terminate

Criteria for selecting LotusScript

The following table summarizes the advantages and limitations of using LotusScript.

Advantages	Limitations
• Embedded BASIC language • Object-based • Sophisticated access to Lotus Domino data • Integrated debugger	• Requires programming skills • More verbose than Formula language • Not executable in Web browsers

Java

Internally, Lotus Domino only supports the use of Java in agents. However, external Java programs can access Lotus Domino applications.

The Lotus Domino Designer IDE provides a basic Java editor in the Agent Builder, which includes color formatting and reference Help for the standard Java language and the Lotus Domino classes.

Where to use Java

Use the Lotus Domino Designer Agent Builder to program Java agents. Java can access the same application data and services as LotusScript.

Use third-party development tools to build applets and standalone applications that access Lotus Domino applications.

Criteria for selecting Java

The following table summarizes the advantages and limitations of using Java.

Advantages	Limitations
• Powerful object-oriented language. • Remote access and networking support. • Extensive class library (beyond Lotus Domino). • Multi-threaded.	• Internally, Lotus Domino only supports its use in agents. • Complex program development requires third-party development tool with debugger.

JavaScript

JavaScript is integrated in Lotus Domino Designer to allow Lotus Domino applications to support users with Web browsers. The Lotus Notes client interprets JavaScript and, therefore, benefits from its unique characteristics. This functionality is critical in building applications for mixed clients.

Where to use JavaScript

Use the Programmer's pane to write JavaScript directly into properties and events of forms, pages, and their design elements. The following table lists common properties and events that execute JavaScript.

Element	Property/Event
Form or page	• JSHeader • onLoad • onUnload
Field or button	• onFocus • onBlur • onChange • onMouseOver

Criteria for selecting JavaScript

The following table summarizes the advantages and limitations of using JavaScript.

Advantages	Limitations
• Object-based language • Web standard • Cross-platform compatibility • Runs in Lotus Notes clients and Web browsers	• Built-in editor, but no debugger • Requires programming skills

Language Selection Criteria

Choosing the best language to use for a given situation can be complex. Many times, the requirements of the situation will narrow the number of choices available.

Criteria for choosing a language

The following table outlines some of the guidelines you can use to make a programming language choice.

If the Application Requires...	Consider Choosing...
Complex control	LotusScript, Java, or JavaScript
User interaction	Formula language, LotusScript, or JavaScript
Access to Lotus Domino data	LotusScript, Java, or Formula language
Calculations with simple results	Formula language
A series of scripted activities	LotusScript or JavaScript
Data access efficiency	LotusScript, Java, or Formula language

Activity 7-1: Identifying programming languages supported by Lotus Domino

This activity describes several application development scenarios. Select the most appropriate language or languages for each task.

1. **You are designing a form with a number of fields. Some fields on the form require an initial value that is changeable by the user. Other fields require a computed value. Some actions may need to be hidden from certain users.**
   a) LotusScript
   b) JavaScript
   c) Formula language
   d) Java

2. **The application that you are designing requires that you get some of the input from dialog boxes.**
   a) Formula language
   b) JavaScript
   c) LotusScript
   d) Java

3. **The application you are creating requires access to documents in databases other than the one in which it is saved.**
   a) Formula language
   b) LotusScript
   c) JavaScript
   d) Java

4. **You are creating an application that requires some fairly complex interaction with the user. In the past, you have had moderate success at automating a series of tasks, but you do not have extensive experience in programming.**
   a) Formula language
   b) LotusScript
   c) JavaScript
   d) Java

5. **The application that you are developing involves an extensive amount of interaction with the user. You will need to validate field contents, position the cursor, and perform other actions in the user interface.**

   a) Formula language

   b) LotusScript

   c) JavaScript

   d) Java

6. **You are working on a process that involves complex flow control, including loops and multiple conditional branches. You have no idea, as you are writing this process, how many times it will need to be executed.**

   a) Formula language

   b) LotusScript

   c) JavaScript

   d) Java

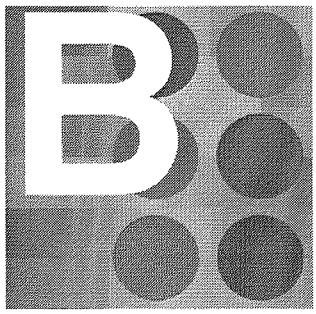
7. **An application is not performing as well as users would like. In analyzing the application, you find that while it is very user friendly, some of the formulas are performing very poorly. Which languages would you use to replace the formulas to improve the performance of the user interface?**

   a) Formula language

   b) LotusScript

   c) JavaScript

   d) Java



Topic B: Identify Formula Components

After completing this topic, you should be able to:

✓ Describe the characteristics of and uses for the Formula language.

✓ List the elements that can be used in statements.

✓ List syntax rules for formulas.

The Formula Language

The Formula language can be used to perform many types of operations. Some of the more common examples are:

- Computing values
- Comparing values
- Automating tasks

Where to use formulas

Formulas can be used in many design elements such as forms, views, pages, fields, buttons, actions, and so on. Code is entered in an event of an object in the Programmer's pane.

Statements

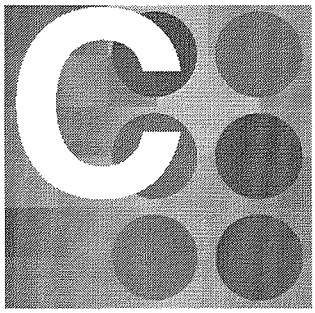
Formulas consist of one or more **statements**, which are made up of a combination of the components described in the following table.

Element	Description
Variables	Can be fields or temporary variables.
Constants	Something whose value does not change. Constants can be text, numeric, or time-date.
Operators	Assign values, modify values, and combine values into new values.
@Functions	Built-in formulas that perform a particular calculation and return a value.
Keywords	Reserved words within the Formula language that perform special functions.

Syntax Rules

All programming languages have rules. The Formula language is no exception. A formula must conform to the general rules described in the following table.

Rule Title	Rule Description
Statement separators	Separate multiple statements using a semicolon; for example: `@IsNewDoc(TrueArgument;FalseArgument)`
Spaces	Any number of spaces, including none, can be used between operators, punctuation, and values. Keywords and functions, however, must be separated by at least one space.
Case	Case is not significant except within text constants. By convention, keywords are usually upper case, and @function and @commands are usually written in mixed upper and lower case.
Operators and values	Two values must be separated by at least one operator.



Topic C: Identify Event Triggers

After completing this topic, you should be able to:

- ✓ Identify the characteristics of objects.
- ✓ Identify the characteristics of events.
- ✓ Describe the order of field evaluation on a form.

Objects

Code is placed in an event of an object in the Programmer's pane. The term **object** refers to the element, such as the form, field, and so on. For instance, you might create a formula to compute the default value of a field. The list of Lotus Domino objects includes, but is not limited to:

- Databases
- Agents
- Actions
- Views
- Forms
- Fields
- Buttons

Events

Code executes in response to an event in a Lotus Domino object. **Events** include opening a database, opening a view, opening a document, moving the cursor into or out of a field, and so on.

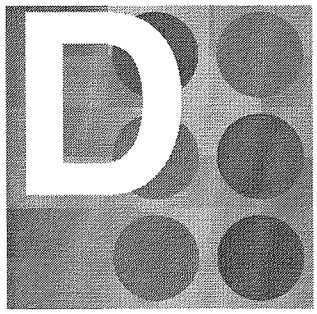
The following table describes several common objects, some of their related events, and what triggers their execution.

Object	Event and Timing
Database	**PostOpen** is triggered after a database, view, or document is opened. **Queryclose** is triggered when a view, document, or database is being closed.
Page or form	**onHelp** is triggered when Help is selected (when F1 is pressed). **onLoad** is triggered when the form or page is loaded. **PostRecalc** is triggered after the object is refreshed (and values are recalculated).

Object	Event and Timing
Field	**Default value** is triggered when a document is created. **Input translation** is triggered when a document is saved, recalculated, or refreshed. **onBlur** is triggered when a field loses focus. **onFocus** is triggered when a field gets focus.
Button	**onClick** is triggered when the button is clicked.

Field Evaluation Order

Fields on a form are evaluated from left to right, and from top to bottom. It is important to keep the order of evaluation in mind when designing the forms in your application. For instance, if a field is computed based on the values in one or more other fields on the form, the computed field must be placed after the field(s) used in the computation.



Topic D: Compute and Compare Values

After completing this topic, you should be able to:

- ✓ Identify the characteristics of constants.
- ✓ Identify the characteristics of variables.
- ✓ Identify the characteristics of operators.
- ✓ Describe how to combine values by using operators.
- ✓ Describe the order of evaluation used in formulas.
- ✓ Describe how to compare values by using comparison operators.

Constants

This section describes how to write formulas to compute and compare values.

Using constants in formulas

Constants represent static information in a formula. The following table shows examples of the three types of constants available in the Formula language.

Constant	Rules to Apply	Examples
Text	● Enclose characters in quotation marks. ● Enclose characters in braces.	● "Hello world." ● {Hello world.}
Numeric	● Specify the sign of the number using a plus or minus sign as the first character, if necessary. ● Use E for scientific notation.	● 10 ● -6.37 ● 37E6
Date-Time	● Enclose a date or time in square brackets. ● Use AM and PM to represent time in a 12-hour format.	● [5:30 PM] ● [13:50] ● [7/11] ● [10/18/71 6:30 AM]

Variables

This section will familiarize you with both field values and temporary variables.

Using field values in formulas

A formula has access to the field values in the document being processed. If a formula refers to a field by name, it will use the value of the field from the current document.

You can use the name of the field as a variable in the formula.

Using temporary variables

Temporary variables exist only within a formula. Use them to make your formulas more modular and readable. You do not need to declare temporary variables before you use them.

Assign a value to a temporary variable using the assignment operator (:=). For example:

```
x := @Power(a; 2) + @Power(b; 2);
```

Then, use that variable in other statements of the same formula. For example:

```
@Sqrt(x);
```

You can also reuse a temporary variable within a formula by assigning a different value to it.

Operators

Operators enable you to manipulate values. Combining and comparing values are common examples of using operators in formulas.

Computed Values

You can only combine values of the same type. For example, you cannot add together text and number values.

The following table describes how to use operators to combine values.

To Combine...	Use the Operators	Example
Number values	+ - / *	10 + 2.37 + 15 6.05 - 2.35 TotalScore / Quantity TotalPrice * DiscountRate
Text values	+	"Welcome, " + FirstName

Order of Evaluation

When creating formulas that compute values, keep the following information in mind.

Order	Description
Parentheses	Using parentheses, you can explicitly force the order of evaluation. Operations within parentheses occur first. Example: (5-3) * (6-4) = 4
Precedence	Operations not in parentheses occur in order of precedence. Example: 5 - 3 * 6 - 4 = -17
Left to right	Operation of equal precedence occur from left to right. Example: 8 / 4 * 2 = 4

Comparison Operators

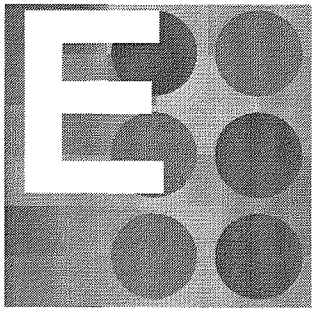
This section describes how to compare values in formulas.

Comparing values using operators

Compare values of the same type using the operators. The following table illustrates how comparison operators work.

This Expression...	Evaluates to True If...	Evaluates to False If...
A = B	The values of A and B are equal.	The values of A and B are not equal.
A < B	The value of A is less than the value of B.	The value of A is greater than or equal to the value of B.
A <= B	The value of A is less than or equal to the value of B.	The value of A is greater than the value of B.
A > B	The value of A is greater than the value of B.	The value of A is less than or equal to the value of B.
A >= B	The value of A is greater than or equal to the value of B.	The value of A is less than the value of B.
A <> B A >< B A != B	The values of A and B are not equal.	The values of A and B are equal.
(A < B) & (B < C) (A < B) | (A < C) !(A < B)	A is less than B and B is less than C. A is less than B or A is less than C. A is not less than B.	B is less than or equal to A or C is less than or equal to B. B and C are both less than or equal to A. B is less than or equal to A.

Note: A and B represent values and can be constants, variables, or combinations of constants, variables, and operators.



Topic E: Work with @Functions

After completing this topic, you should be able to:

- ✓ Identify the characteristics of and uses for @functions.
- ✓ List some @functions that return data about an application.
- ✓ Describe conditional statements.

@Functions

@Functions are built-in formulas that perform calculations and either return a value or perform an action. @Functions are used in Lotus Domino design elements to perform tasks such as:

- Accessing information about the application, its data, and the user.
- Modifying values.
- Acting on a condition.
- Formatting text strings.
- Calculating numeric values or values in a list.

Syntax

@Functions contain the name of the function, followed by any arguments. The general format of an @function is:

```
@FunctionName(argument1; argument2; argument3);
```

Common @Functions

The following table describes some of the @functions that return data about the application.

For Information About the...	Use the Function	Which Returns...
Current document	@Created	The date-time value for when the document was created.
State of the current document	@IsNewDoc	True (1) if the document is new, or False (0) if the document is not new.
View	@ViewTitle	The title of the current view.

For Information About the...	Use the Function	Which Returns...
Database	@DbName	The server and file name of the current database.
User	@UserName	The current user's name.
User environment	@ClientType	"Notes" if the client is Lotus Notes or "Web" if the client is a Web browser.
Web client	@BrowserInfo	Determines the capabilities of a Web client.

Examples of @functions

The following table contains common examples of formulas.

This Formula...	Returns
`@Adjust ([10/11/05]; 0; 1; -4; 0; 0; 0)`	The date-time value, 10/07/05
`@Text (10)`	The text value, 10
`@ProperCase ("hello world.")`	The text value, Hello World.
`@Trim ("  hello world.  ")`	The text value, hello world.

Conditional Statements

The `@If` function executes one statement or another depending on whether a logical value is true or false. The `@If` statement has an odd number of parameters, with a minimum of three. The parameters are:

- Condition, which is the first parameter that is checked.
- True statement, or the action to perform if the condition is true.
- False statement, or the action to perform if the condition is false.

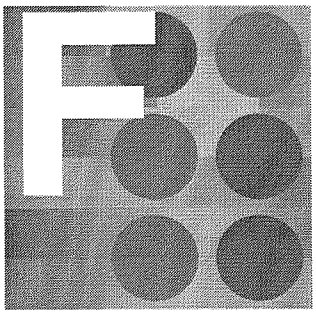
`@If` statements are evaluated from left to right. The first true condition evaluated causes its corresponding True statement to be processed. Subsequent conditions in that `@If` statement are not processed.

If an `@If` statement contains two conditions, the syntax is:

```
@If(condition1; TrueStatement1; condition2; TrueStatement2;
FalseStatement)
```

The following table shows examples of conditional statements.

This Conditional Statement...	Returns...
`@If (3<10; "True"; "False")`	True
`@If (Author = @UserName; "My document"; "Not mine")`	"My document" if the Author field contains the name of the current user.



Topic F: Explore Formatting and Comments

After completing this topic, you should be able to:

✓ Identify formatting that can be added to formulas.

✓ Describe how formulas can be documented.

Formatting

In order to make your code easier to read and debug, it is good practice to include comments and formatting. This helps not only the person who developed the code, but also any developer who may have to modify the code at a later date.

Formatting Formula language

Formulas can get quite lengthy, particularly when using nested `@If` statements. To make complicated formulas easier to read, use line breaks and indenting. For example, even an experienced Lotus Domino developer would have to study the following formula to interpret what it does:

```
@If(Invoiceamount>POamount;@If (Invoiceamount-POamount>1000.00;
@Mailsend(Vendor:Approver;"";"";"Rejection";"Invoice"+
invoiceno+ " has been
rejected");@Mailsend(Approver;"";"";"Invoice
overage";"Invoice"+ invoiceno+"has been
submitted"));@Mailsend(Vendor:Approver;"";"";"Processed";"Invoi
ce"+ invoiceno+" for purchases made under PO#"+Pono+"))
```

Now look at the same formula written following simple formatting rules:

```
@If(Invoiceamount>POamount;
   @If (Invoiceamount-POamount>1000.00;
      @Mailsend(Vendor:Approver;"";"";"Rejection";"Invoice"+
         invoiceno+ " has been rejected");
      @Mailsend(Approver;"";"";"Invoice overage";"Invoice"+
         invoiceno+"has been submitted"));
   @Mailsend(Vendor:Approver;"";"";"Processed";"Invoice"+
      invoiceno+" for purchases made under PO#"+Pono+"))
```

The formula has been formatted so that the conditions and the True and False statements start on separate lines. The statements related to a condition are indented under that condition.

Comments

The formula gets even easier to understand when you add comments.

Commenting Formula language

By adding explanatory text to any code that you write, your applications become much easier to maintain. Documenting code is best done while you are writing it.

Use the REM keyword to include comments in Formula language code. The syntax is:

```
REM { comments };
```

Include a REM statement at the beginning of each line of explanatory text in a formula.

The following example contains several lines of explanatory text before the formula starts. It also contains REM statements inside the formula to further document what the code is doing.

```
REM {An invoice amount is checked against};
REM {its corresponding Purchase Order to see if the invoice };
REM {exceeds the amount of the PO. If it does exceed the PO };
REM {amount by more than $1,000, an email is sent that };
REM {indicates the invoice has been rejected. If the invoice };
REM {amount is over the PO amount but under $1,000, an email };
REM {is sent indicating that an invoice over the PO amount };
REM {has been submitted. Finally, if the invoice is less };
REM {than the PO amount, an email is sent indicating the };
REM {invoice has been processed.};
REM ;
REM {This statement checks to see if the invoice is less than };
REM {the PO amount.};
@If(Invoiceamount>POamount;
   REM {Is the difference greater than $1,000? };
   @If (Invoiceamount-POamount>1000.00;
      REM {Reject the invoice};
      @Mailsend(Vendor:Approver;"";"";"Rejection";"Invoice"+
         invoiceno+ " has been rejected");
      REM {Notify approver that invoice has been submitted};
      @Mailsend(Approver;"";"";"Invoice overage";"Invoice "+
         invoiceno+"has been submitted"));
   REM {Process the invoice.};
   @Mailsend(Vendor:Approver;"";"";"Processed";"Invoice"+
      invoiceno+" for purchases made under PO#"+Pono"))
```

Activity 7-2: Interpreting Formulas

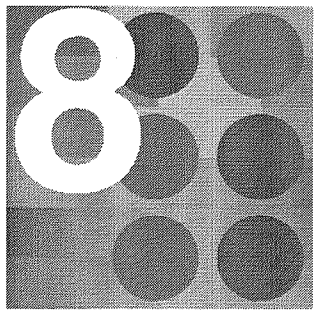
Scenario: A document contains the following fields and values:

```
FirstName: Susan
LastName: Sunshine
Created on 12/9/2005 at 11:59:37 PM
```

Based on the information in the scenario, determine the result for each of the following formulas.

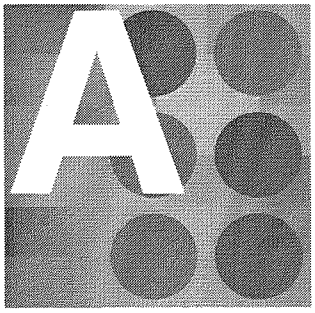
FirstName: _____
"LastName": _____
FirstName + LastName: _____
LastName + ", " + FirstName: _____ _____
FirstName = "Susan": ____
(FirstName != "Larry") & (LastName = "Sunshine"): ____
@Adjust (@Created; 0; 6; 0; -2; 0; 0): _____ _____ ___
@If (@Created < @Today; "Old"; "New"): ___
@Text (10.22): ___ _____ _____
fullName := FirstName +" " + LastName; "Welcome back, " + fullName:

_____ _____ _____ _____



Using Formulas in Forms

- **Topic A:** Create Window Titles
- **Topic B:** Work with Computed Values
- **Topic C:** Work with Field Values
- **Topic D:** Work with Lists
- **Topic E:** Prompt Users
- **Topic F:** Create Reusable Code
- **Topic G:** Implement Error Checking



Topic A: Create Window Titles

After completing this topic, you should be able to:

- ✓ List uses for formulas in forms.
- ✓ Describe where to add code to a form.
- ✓ Describe form events and their triggers.
- ✓ Describe window titles.
- ✓ Identify the characteristics of and uses for the `@IsNewDoc` function.

Common Uses for Formulas

Use formulas in forms to add, calculate, and format data automatically. With formulas, you can:

- Display one title when the document is new, and another title when the document is being edited.
- Calculate ending dates based on start dates and lengths.
- Enter the document creation date.
- Populate a list of options.
- Ensure that users do not save the document with specified fields left empty.
- Format user entries to be consistent in all documents.

Form Programming

Each element of the form, including the form itself, is an object. The Programmer's pane provides access to the programmable properties and events of objects on the form.

Adding a formula in the Programmer's pane

Follow these steps to add a formula to an object in the application.

Task	Procedure
1	Select the object, either: ● On the form in the **Work** pane. ● From the **Objects** tab in the **Programmer's** pane. **Result:** IBM® Lotus® Domino Designer® highlights the object on the **Objects** tab and displays the programmable properties and events of the object.
2	Select the property or event to which you want to add a formula. **Result:** Lotus Domino Designer highlights the property or event.
3	At the top of the **Script** area, select **Client** and **Formula.** **Result:** The **Reference** tab lists the database fields and @functions that can be used in the formula.
4	Enter the formula. **Result:** Lotus Domino Designer highlights different elements of the formula with colors.
5	Save the formula. **Result:** Lotus Domino Designer checks the syntax of the formula.

Form Events

Form events are triggered when a document is opened, saved, refreshed, closed, or switched between read and edit mode. You can use formulas in form events to set fields, prompt users, or perform background processing.

The following table lists most of the form events and when they evaluate.

Form Event	When the Event Is Triggered
`QueryOpen`	Before a document opens.
`PostOpen`	After a document opens and is drawn onto the screen.
`QueryModeChange`	Before the document is changed into either edit or read mode.
`PostModeChange`	After the document is changed into either read or edit mode.
`PostRecalc`	After the document has recalculated, usually after F9 is pressed.

Form Event	When the Event Is Triggered
`QuerySave`	Before the document is saved.
`PostSave`	After the document is saved.
`QueryClose`	Before the form is closed.

Window Titles

Both forms and pages contain a property called the **window title.** You can write a formula to compute a text string to display in the upper-left corner of the client when users open the element. It is good practice to always set the window title when designing forms and pages. If the form or page opens in a Web browser, the window title will not display.

Setting the window title

Follow these steps to set the window title of a form or page.

Task	Procedure
1	In the **Object** tab of the form or page, select the window title.
2	Enter a formula that results in a text value.
3	Save the formula.
4	Preview the page or form.

The @IsNewDoc Function

The `@IsNewDoc` function is particularly useful when creating a form window title. Window titles that incorporate a field from the document will not display the window title correctly when the document is new and there is no data yet. By combining `@IsNewDoc` and `@If`, you can create a formula that displays one text phrase if the document is new, and another if the document has been saved.

The syntax for `@IsNewDoc` is simply:

```
@IsNewDoc
```

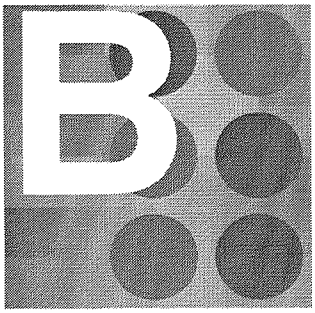
The function returns a value of 1 for True if the document has not yet been saved. It returns a value of 0 for False if the document has been saved.

Example of writing a formula using @If and @IsNewDoc

The following example illustrates how to combine `@If` and `@IsNewDoc` to create a meaningful window title:

```
@If(@IsNewDoc;"New Document";Subject)
```

In this case, the phrase "New Document" displays if the document is new. If the document has been saved, the window title changes to the value in the Subject field of the document.



Topic B: Work with Computed Values

After completing this topic, you should be able to:

- ✓ Identify the characteristics of Computed for display fields.
- ✓ Identify the characteristics of and uses for computed text.
- ✓ List the types of fields that can be used to store values in a document.

Computed for Display Fields

One way to compute values that display on a form is to use **Computed for display** fields. Computed for display fields are recalculated every time a user opens or saves a document, or when the document is refreshed.

The contents of a Computed for display field are not stored in the document. The following table describes examples of using Computed for display fields.

Use a Computed for Display Field to Display...	Example
The current time	`@Now`
The user's name	`@UserName`
The creation date of a document	`@Created`
The value of a list field when the document is open	`WorkType`

Computed for display fields require a formula. The formula must evaluate to a value suitable for storage in the field. For example, if the type of field is date-time, the formula must result in a date-time value.

Computed Text

Computed text is another element you can add to a form or page to display computed values. It is similar to a Computed for display field, except that the result of the formula must be a text value and, unlike fields, you may use it on a page. You can format computed text like other text elements using the Computed Text Properties box.

The following table describes examples of using computed text.

Use Computed Text to Display...	Example
Different text based on a condition	`@If ((DueDate < @Today) & (Status != "Complete"); "This is late!"; "")`
Dynamic messages	`"Welcome, " + @Name([CN]; @UserName)`

Adding computed text

Follow these steps to add computed text to a form.

Task	Procedure
1	Open the form or page in Lotus Domino Designer. **Result:** The form appears in the Work pane and the Programmer's pane opens.
2	Place the cursor where you want the result of the computed text to appear.
3	Choose **Create→Computed Text.** **Result:** The **Computed Text** Properties box opens. The value property of the new computed text object is highlighted in the Object list.
4	Enter a formula that evaluates to a text value in the Script area of the Programmer's pane. **Result:** Lotus Domino Designer checks for formula syntax errors.
5	Set properties to format the text. **Result:** The result of the formula in Step 4 will display according to the properties you set.
6	Preview the page or form. **Result:** The result of the formula displays in the form or view.

Practice Activity 8-1: Create a Dynamic Title

Scenario: Worldwide Corporation would like to customize the titles displayed to users when they open certain documents in the Policies and Procedures database.

Note: The guided solution to this activity can be found in Appendix A, Solutions to Practice Activities.

Perform the following task to complete this activity.

1. When a Policy document is opened, check to see if it is new or not. Display either the title of the Policy or "New Policy", depending on the state of the document.

Computed Fields that Store Values

To store values in a document, you must use one of the following types of fields:

- Computed
- Computed when composed
- Editable

This section describes how to use these field types and set their values. The following table describes Computed and Computed when composed fields.

Field	Calculates a Value...	Example
Computed	Each time users create, refresh, or save the document.	The result of a calculation involving other fields, such as calculating a total.
Computed when composed	When the users originally create the document, or the first time that particular field appears on a form with which that document is opened.	• The original author • A value that never changes • The date the document was created

Setting the value of a Computed field

Follow these steps to add a formula to calculate the value of a Computed field.

Task	Procedure
1	Open the form in Lotus Domino Designer.
2	Select the field.
3	Under the field in the **Objects** tab, select **Value.**
4	Enter the formula to compute a value suitable for storage in the field.
5	Test the formula by previewing the form.

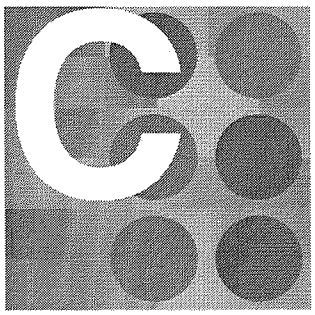
Using formulas in editable fields

Gather information from users through editable fields. To improve the usability of editable fields, you can use formulas to:

- Compute a default value when users create documents.
- Translate or reformat the value when users save documents.
- Validate or verify the value when users save documents.

Specifying a default value

Specify an initial value for an editable field in the Programmer's pane. Add the formula to the field's Default value property. The formula executes only when users create documents.



Topic C: Work with Field Values

After completing this topic, you should be able to:

- ✓ Identify the characteristics of and uses for input translation formulas.
- ✓ Define @functions that can be used to format text.
- ✓ Identify the characteristics of and uses for input validation formulas.
- ✓ Define @functions that can be used to validate field values.

Input Translation Formulas

You can automatically format data by specifying an input translation formula for editable fields. Using an input translation formula, you can format such things as:

- Telephone numbers, by adding parentheses, hyphens, or periods.
- Proper nouns, by capitalizing only the first letter.

Working with input translation formulas

The input translation formula is triggered when the document is saved or refreshed. The input translation formula adjusts or reformats the value.

The result of an input translation formula must evaluate to the same type of value specified by the field. For example, if you are translating a text field, the formula must evaluate to a text value.

Note: Input translation formulas do not apply to rich text fields.

Text Formatting Functions

There are several @functions available for formatting text. The following table describes some of these functions.

Function	Description
`@Trim`	Removes leading, trailing, and redundant spaces in a text string.
`@ProperCase`	Capitalizes the first letter of each word in a text string.
`@UpperCase`	Converts all letters in a text string to uppercase.

Function	Description
`@LowerCase`	Converts all letters in a text string to lowercase.

Writing an input translation formula

Follow these steps to write an input translation formula.

Task	Procedure
1	Select the field.
2	On the **Objects** tab of the **Programmer's** pane, select the field's Input Translation event.
3	In the **Script** area, write the translation formula.
4	Save the form.
5	Test the formula: ● Preview the form. ● Enter a value and refresh the document.

Activity 8-2: Interpret input translation formulas

Follow these steps to complete the activity.

Step	Action
1	Open the your copy of the **Practice** database in the Lotus Notes client.
2	Choose **Create→Other.** Select **Translation Demo** and click **OK.**
3	Work through the translation formulas.
4	Click **Mark Now** to see your results.
5	In the **Execution Security Alert** dialog box, select **Start trusting the signer to execute this action,** and click **OK.**

Input Validation Formulas

You can stop users from saving documents with incorrect or incomplete values by specifying an input validation formula when designing editable fields.

Working with input validation formulas

When users save or refresh documents, the input validation formulas are triggered. The input validation formula is a conditional statement that:

- Determines if the value is acceptable or not.
- If the value is acceptable, continues saving the document.
- If the value is not acceptable, displays a message to users indicating that the value is not acceptable and aborts the save or refresh operation.

Note: The input validation formula runs after the input translation formula.

Note: Input validation formulas do not apply to rich text fields.

Validation Functions

Use the functions `@Success` and `@Failure` as the actions in the input validation formula.

The following table describes how to use the two functions when validating field values.

If the Field Value Is...	Use This Function...	To...
Acceptable	`@Success`	Continue saving or refreshing the document.
Not acceptable	`@Failure`	Display a pop-up message box directing users to fix the problem, and abort saving or refreshing the document.

Examples of input validation formulas

The following code verifies that a new number entered by a user is within a specific range:

```
@If (NewNumber >= LowValue & NewNumber <= HighValue;
   @Success;
   @Failure ("Please enter a value between " + @Text(LowNumber)
+ " and " + @Text (HighNumber))
```

The following code verifies that the field contains a value.

```
@If(Title="";
@Failure("Please enter a value.");
@Success)
```

Writing a field input validation formula

Follow these steps to write a field input validation formula.

Task	Procedure
1	Select the field.
2	On the **Objects** tab under the field, select **Input Validation.**
3	In the **Script** area, enter a formula to test the value.
4	Save the form.
5	Test the input validation formula: ● Preview the form. ● Enter appropriate and inappropriate values and refresh the document.

Practice Activity 8-3: Automate Data Entry and Formatting

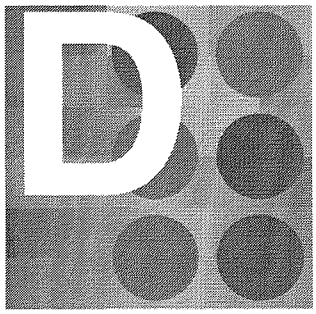
Scenario: During this activity, you will automate data entry and formatting in the Policy form. Here is a list of the functionality that Worldwide would like to incorporate into the application:

- The policy title should be formatted using proper case.
- Create a field to capture the date the policy is created. This date should be displayed without the time element. Name it `PolicyCreatedDate`
- The effective date should calculate to one month from the date the policy is created in order to account for an internal review process. This date should also be displayed without the time.
- When a policy is created, the document should not be saved unless it contains a title and category.

Perform the following tasks to complete this activity:

1. Implement the above specifications.
2. Preview the form and test your formulas.

Note: The guided solution to this activity can be found in Appendix A, Solutions to Practice Activities.



Topic D: Work with Lists

After completing this topic, you should be able to:

- ✓ Identify the characteristics of lists.
- ✓ Define @functions that can be used with lists.
- ✓ List ways to define the list of choices in a field.
- ✓ Identify the characteristics of and uses for the @DbColumn function.

Lists

In Lotus Domino, a **list** is defined as a named entity that contains multiple values of the same data type. The return value of some @functions is a list. Fields that allow multiple values contain lists. Constants and variables in formulas can contain a list. You can also present lists to a user using a prompt box.

Multiple elements in a list are separated by a colon (:). For example, the following code defines a temporary variable and assigns a list of text elements to it:

```
cities := {New York} : {London} : {Paris} : {Boston};
```

List Functions

The following table contains @functions useful for working with lists.

@Function	Usage	Example
@Elements(list)	Returns the number of elements in a list.	`@Elements(Colors)` **Returns:** The number of elements in the Colors field.
@Explode(String; separator)	Splits a string into a list of values using the specified separator.	`@Explode("Red Green Blue"; " ")` **Returns:** "Red":"Green":"Blue"
@Implode(list;separator)	Converts a text list into a string using the specified separator.	`@Implode("Red":"Green":"Blue";"*")` **Returns:** Red*Green*Blue

@Function	Usage	Example
@IsMember(value; list)	Returns true if the value is a member of the list.	`@IsMember("Orange"; "Red":"Green":"Blue")` **Returns:** 0 (False)
@Member(value;list)	Returns the number of the element in the list.	`@Member("I";"A":"E":"I":"O":"U")` **Returns:** 3
@Subset(list;number)	Returns a portion of the list containing the number of elements specified by number.	`@Subset({A}:{E}:{I}:{O}:{U};2)` **Returns:** {A}:{E}
@Unique(list)	Removes duplicates from a list.	`@Unique({A}:{E}:{I}:{E})` **Returns:** {A}:{E}:{I}

Activity 8-4: Evaluate list formulas

Scenario: Use Lotus Domino Designer Help to help evaluate the formulas.

Follow these steps to complete this activity.

Step	Action
1	Open your copy of the **Practice** database in the Lotus Notes client.
2	Choose **Create→Lists Demo.**
3	Evaluate the list formulas.
4	Enter the result in the appropriate field.
5	When you have evaluated all of the formulas, click **Mark Now** to see your results.

List Choices

Previously, you learned about the different types of fields, which include fields that offer a pre-defined list of choices to the user. To define the list of choices, you can do either of the following:

- Enter each choice.
- Write a formula to calculate the choices.

Adding a formula to compute the list of choices in a field

Follow these steps to create a list of choices using a formula.

Task	Procedure
1	In the Properties box of the **List** field, click the **Control** tab.
2	Select **Use formula for choices.**
3	Enter the formula to create the list of choices.
4	Save the formula.
5	Test the formula by previewing the form.

The @DbColumn Function

The @DbColumn function returns an entire column of values from a view. By writing a formula with @DbColumn to compute the choices of a field, you can avoid updating the list manually as data changes.

The syntax of @DbColumn is:

```
@DbColumn( class : cache ; server : database ; view ;
columnNumber )
```

The following table describes each parameter in the formula.

Argument	Data Type	Description
`class : cache`	Text list	● Indicates the type of database (class) you are accessing and whether to cache the results. ● For example, `"":""`, accesses a Lotus Domino database and caches the results.
`server : database`	Text list	● Specifies the location and file name of the database ● For example, `"":""`, refers to the current database.
`view`	Text	● Specifies the name of the view.
`columnNumber`	Number	● Specifies the column number within the view.

Example of using @DbColumn

The example formula below is based on the following characteristics:

- The view name is In Progress.
- The view is located in the Tracking database on Server01.
- The first column contains the title of all projects currently in progress.

The formula returns a list of all the projects currently in progress (note the use of temporary variables to make the formula more readable):

```
REM {This formula returns the values in the first column};
REM {of the In Progress view in Tracking on Server01};
source := "Notes": "";
location := "Server01":"Tracking.NSF";
view := "In Progress";
colNumber := 1;
@DbColumn (source; location; view; colNumber)
```

Practice Activity 8-5: Compute a List of Choices for a Field

Scenario: To make the Policies and Procedures application easier to maintain over time, create a view that contains the policy categories in the first column. When a policy is created or edited, the categories should be displayed from the Categories view. In order to accomplish this:

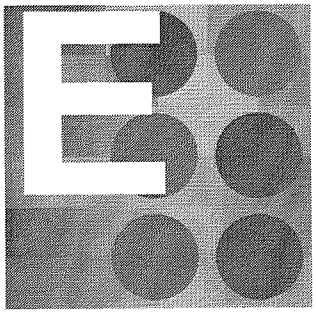
- Create a Category form containing a field called Category.
- Create Category documents with the following categories:
  - Office Guidelines
  - Benefits
  - Holidays
  - Grievance Procedures
  - Security
  - E-mail Etiquette
  - Diversity
- Create a view called Categories that only displays the Category documents.
- Modify the Policy form to perform a lookup to the Categories view to produce the list for the Category field.

Hint: The view should only display documents created with the Category form.

Note: The guided solution to this activity can be found in Appendix A, Solutions to Practice Activities.

Perform the following tasks to complete this activity:

1. Implement the above specifications.
2. Preview the form and test your formulas.



Topic E: Prompt Users

After completing this topic, you should be able to:

✓ Identify characteristics of and uses for the @Prompt function.

The @Prompt Function

What do you do when you need to prompt a user for information in your application? If you are working with the Formula language, you use the @Prompt function. @Prompt allows you to create a more interactive interface for Notes applications. The @Prompt function allows users to enter values, respond to questions, or choose from a list of choices.

There are many different styles of prompt boxes, and the prompt boxes can be triggered in different ways. @Prompt can be used in fields, hotspots (buttons), manual agents, and toolbar buttons.

Caution: @Prompt does not set the modified flag of the document. Therefore, if a form uses only @Prompts to collect data and set fields, the user could close the document and not be prompted to save it.

Syntax for @Prompt

The syntax for @Prompt is:

```
@Prompt( [style] : [ NOSORT ] ; title ; prompt ; defaultChoice ;
choiceList ; filetype )
```

The following table describes the parameters for @Prompt.

Argument	Data Type	Description
style	Keyword	Indicates the type of dialog box to display.
NOSORT	Keyword	If present, the list of choices is not automatically sorted.
title	Text	The title for the dialog box.
prompt	Text	The text you want to display in the dialog box.
defaultChoice	Text	The value to use as a default value.
choiceList	Text list	The values to display in the dialog box's list.

Argument	Data Type	Description
filetype	Text	The types of files to display when using the LOCALBROWSE style.

The following example shows a simple @Prompt formula:

```
@Prompt([OK];"Prompt box";"You did it! Click OK to close.")
```

Triggering @Prompts

Prompt dialog boxes can be triggered in the ways described in the following table.

Trigger	Description
Field formulas	Use @Prompt to prompt the user to enter a value in a field.
Button formulas	@Prompt can be used in buttons to take input from a user.
Form and View action buttons or manual agents	@Prompts are useful in Action buttons and agents. For example, you could prompt the user for a value and set the value of a field based on what the user enters.
Password field formulas	In combination with @Password, @Prompt can be used to prompt for a password on a form.