



Your Short Cut to Knowledge

The following is an excerpt from a Short Cut published by one of the Pearson Education imprints.

Short Cuts are short, concise, PDF documents designed specifically for busy technical professionals like you.

We've provided this excerpt to help you review the product before you purchase. Please note, the hyperlinks contained within this excerpt have been deactivated.

Tap into learning—NOW!

Visit www.informit.com/shortcuts for a complete list of Short Cuts.



SAMS

Cisco Press

**IBM
Press™**

que®



FileMaker® 8

Calculation Functions

Steve Lane
Scott Love
Bob Bowers

The calculation dialog in FileMaker serves as a fundamental element in nearly all development activities. The calculation dialog allows developers easy access to the data fields in their solutions and to a complete function list. This shortcut presents a complete description of each calculation function, lists examples, and in many cases offers additional comments on usage. In this shortcut you will also find references to a companion book, **Special Edition Using FileMaker 8**, where you can find additional information on functions.

Calculation Functions

Abs()

Category: **Number**

Syntax: **Abs (number)**

Parameters:

number—Any expression that resolves to a numeric value.

Data type returned: **Number, Time**

Description:

Returns the absolute value of number; absolute value is always a positive number.

Examples:

Function	Results
Abs (-92)	Returns 92.
Abs (Get (CurrentPlatform))	Returns 1 for MacOS and 2 for Windows.
Abs (RetailPrice - WholeSalePrice)	Returns the difference between the two prices, regardless of which one is larger.
Abs (2:15:00 – 3:30:00)	Returns 1:15:00.

Atan()

Category: **Trigonometric**

Syntax: **Atan (number)**

Atan()

Parameters:

number—Any expression that resolves to a numeric value.

Data type returned: **Number**

Description:

The arc tangent of a number is the angle (measured in radians) whose tangent is the specified number. The range of values returned by the Atan function is $-(\pi/2)$ to $\pi/2$.

If $\text{Atan}(x) = y$, then $\text{Tan}(y) = x$.

$\text{Atan}(x) = \text{Atan}(-x)$.

Examples:

$\text{Atan}(0) = 0$

$\text{Atan}(1) = .785398163$

which is $\pi/4$ radians, or 45 degrees.

Average()

Category: **Aggregate**

Syntax: **Average (field { ; field... })**

Parameters:

field—Any related field, repeating field, or set of non-repeating fields that represent a collection of numbers. Parameters in curly braces { } are optional and may be repeated as needed, separated by a semicolon.

Data type returned: **Number**

Description:

Returns a numeric value that is the arithmetic mean of all non-blank values in the set designated by the parameter list. The arithmetic mean of a set of numbers is the sum of the numbers divided by the size of the set. Blank values are not considered as part of the set.

Average()

When the parameter list consists of two or more repeating fields, `Average()` generates a repeating field in which the corresponding repetitions from the specified fields are averaged separately. So, if a field `Repeater1` has two values, 16 and 20, and another field, `Repeater2`, has two values, 14 and 25, `Average (Repeater1; Repeater2)` would return a repeating field with values 15 and 22.5.

Examples:

Function	Results
<code>Average (field1; field2; field3)</code>	Returns 2 when field1 = 1, field2 = 2, and field3 = 3.
<code>Average (repeatingField) repeatingField[2]; repeatingField[3];)</code>	Returns 2 when repetition1 = 1, repetition2 = 2, and repetition3 = 3.
<code>Average (repeatingField[1];</code>	Returns 2 when repetition1 = 1, repetition2 = 2, and repetition3 = 3.
<code>Average (Customer::InvoiceTotal)</code>	Returns \$450 when a customer has three related invoice records with invoice totals of \$300, \$500, and \$550.

Case()

Category: **Logical**

Syntax: **Case (test1; result1 {; test2; result2; defaultResult...})**

Parameters:

test(n)—An expression that yields a Boolean result.

result(n)—The value to return if corresponding test is true.

defaultResult—The value to return if all tests are false. Parameters in curly braces { } are optional and may be repeated as needed, separated by a semicolon.

Case()

Data type returned: **Text, Number, Date, Time, Timestamp, Container**

Description:

The **Case** function returns one of several possible results based on a series of tests.

Each test expression is evaluated in order, and when the first true expression (one that resolves to a Boolean 1) is found, the value specified in the result for that expression is returned. The function stops evaluating as soon as it finds a true test.

The default result at the end of the parameter list is optional. If none of the tests evaluate to True, the function returns the value specified for defaultResult. If no default result is specified, the **Case** function returns an “empty” result. If you believe that one of the tests in the **Case** should always be true, we recommend using an explicit default case, possibly with a value of “default” or “error” to assist in error trapping.

Consider using hard returns in long **Case()** statements to make them more readable, and indent lines with tabs, as shown previously. (Note that this example makes repeated calls to **Get(SystemLanguage)**; in practice it might be better to use **Let()** to make a single call to **Get(SystemLanguage)** so that it needs to be evaluated only once.)

In the last example, while all three tests resolve to true, only the first line is executed and its result returned. Using the **Case()** function, with its “short-circuiting” feature, can help with performance tuning.

Examples:

Function	Results
<code>Case (IsEmpty (Contact_Name) ; 1)</code>	Returns 1 if the Contact_Name field is empty.

Note that a default value is not required, making the usage of **Case()** shorter than **If()**.

```
Case (
  Get(SystemLanguage) = "English"; "Welcome";
  Get(SystemLanguage) = "French"; "Bienvenue";
  Get(SystemLanguage) = "Italian"; "Benvenuto";
  Get(SystemLanguage) = "German"; "Willkommen";
  Get(SystemLanguage) = "Swedish"; "Välkommen";
```

Case()

```
Get(SystemLanguage) = "Spanish "; "Bienvenido";
Get(SystemLanguage) = "Dutch"; "Welkom";
Get(SystemLanguage) = "Japanese"; "Irashaimasu" ;
"Sorry... not sure of your language." // default value
)
```

Returns a welcoming message in the language determined by the Get (SystemLanguage) function.

```
Case (
SalesTotal < 10; .1;
SalesTotal < 50; .2;
SalesTotal < 100; .3;
.35
)
```

Returns .1 when the value in the SalesTotal field is 5, and returns .2 when the value in the SalesTotal field is 12.

Ceiling()

Category: **Number**

Syntax: **Ceiling (number)**

Parameters:

number—Any expression that resolves to a numeric value.

Data type returned: **Number**

Description:

Returns number rounded up to the next integer.

One common use for the Ceiling function is finding out how many pages will be required to print x items if y items fit on a page. The formula for this is $\text{Ceiling} (x / y)$. For instance, if you have 16 items, and 5 can print per page, you would need $\text{Ceiling} (16/5) = \text{Ceiling} (3.2) = 4$ pages.

Ceiling()

Examples:

Ceiling (1.05) = 2

Ceiling (-4.6) = -4

Ceiling (3) = 3

Choose()

Category: **Logical**

Syntax: **Choose (test; result0 {; result1; result2...})**

Parameters:

test—An expression that returns a number greater than or equal to zero.

result(n)—The value returned or the expression that is evaluated based on the result of the test. Parameters in curly braces { } are optional and may be repeated as needed, separated by a semicolon.

Data type returned: **Text, Number, Date, Time, Timestamp, Container**

Description:

Returns one of the result values according to the integer value of test. FileMaker evaluates test to obtain an index number, which is used to then select the corresponding ordinal result.

The Choose function is a 0-based list. Choose (1; "a"; "b"; "c") will return "b".

Any fractional value of test is ignored (as opposed to rounded) when obtaining the index number. Choose (1.9; "a"; "b"; "c") will return "b".

If the index value returned by test exceeds the number of results available, the Choose function will not return any result—The field will be blank as opposed to having a "?" in it. There is no way to define a default value to use when the index value exceeds the number of results available.