**[Built-in Functionality](https://docs.opendatakit.org/odk2/xlsx-converter-reference/" \l "id39)**

The **jquery** and **underscore** libraries are available when defining calculates expressions or writing statements for the **condition** column or the **required** column.

ODK Survey exposes built-in functionality through formula functions to decrease form development time.

[**Formula Functions**](https://docs.opendatakit.org/odk2/xlsx-converter-reference/#id40)

The following formula functions can be used to simplify calculations or expressions.

| *Built in formula functions* | | |
| --- | --- | --- |
| **Name** | **Description** | **Example** |
| assign | Assignment operator that will assign the value  to the field and return the value | assign('fieldname',value) |
| countSelected | Returns the number of items selected from a  select\_multiple prompt | countSelected(data(‘options’)) |
| data | Returns the value of a field or session variable. | data(‘options’) |
| equivalent | Check to see if two values are equivalent | equivalent(data(‘option1’), data(‘option2’)) |
| isFinalized | Returns true if this submission is finalized | isFinalized() |
| localize | Localizes the text passed in. | localize(data('options')) |
| metadata | Returns a metadata field of this row | metadata(‘\_group\_read\_only’) |
| not | Negates the argument passed in. | not(selected(data('examples'), 'label\_features')) |
| now | Returns the current date | now().getDay() |
| selected | Returns true if the value selected from a select  prompt is equal to the second argument passed  into the function. | selected(data('visited\_continents'), 'NorthAmerica') |

And, additionally, the *opendatakit* object is also available for use in calculates expressions.

JavaScript Operators

The built in formula functions can be combined in advanced ways using any valid JavaScript expression. This is particularly useful for creating complex condition statements to implement skip patterns or conditional statements for required variables. JavaScript operators will allow the expressions to involve more than one variable or more than one response from a single variable. Parentheses can be used in creating particularly complex conditions. A few basic JavaScript operators:

| **Name** | **Description** | **Example** |
| --- | --- | --- |
| && | And | data('person\_age')>=18 && selected(data('pizza\_type'), 'mushroom') |
| || | Or | (selected(data('pizza\_type'), 'mushroom') || selected(data('pizza\_type'), 'onions') |
| == | Equal | data('person\_number') == 1 |
| === | Strict equal of the same type | data('consent')==="yes" |
| >= | Greater than or equal to | data('age') >=18 |
| <= | Less than or equal to | data('age') <=17 |

**Tip**

Make sure that statements using && and || operators for variables that were select\_one type are logical and that they work as intended. For example, if the variable pizza\_type had been a select\_one, the statement (selected(data('pizza\_type'), 'mushroom') && selected(data('pizza\_type'), 'onions') could never be valid, because the respondent could only have selected one or the other or neither, not both. Therefore, the example instead uses an ||statement.