Podio Calculation Tips

In this guide you find some useful tips on how you can use the calculation field in Podio.

We have included some examples of how the calculations should look. When you copy in the string into the calculation field, make sure to reference the right fields, use the “@” symbol and start typing the field name.

When you would like to reference values in a calculation field from other apps, then first you have to have an incoming or outgoing relationship field that connects the two apps.

Please note, you will need to save the app template first when making modifications that require the use of the @reference within the calculation. Once the template is saved, you'll be able to add your calculation and reference the relevant fields using the “@” symbol.

Conditional calculations

If statement

Use: Evaluate a statement as either true or false and then show different outcomes in a calculation field based on selected categories within a category field.

Example: If you use a project and a deliverables app and would like to calculate in your project app the sum of approved deliverables from your deliverables app, you can set the following calculation up in your deliverables app to show only the budgets that have been approved. You would need the following fields in your deliverables app, a number field for the budget, a category field to indicate whether it’s been approved or not, and a calculation field for the calculation that will only show approved budgets. In this example we would like to use and reference the “Approved Budget” calculation field from the deliverables app, within the projects app. To achieve this we need to ensure that the two apps are related with a relationship field. This will allow us to use the “@” symbol (in the Projects app) to pull in the variables (from the Deliverables app). We can then add a calculation field in your projects app and use the @Sum of Approved Budget to show the calculation.

Note: Using the “@Sum of” command will pull in variables from number type fields (money, numbers, calculations that display numbers, durations, percentages, and progress fields).

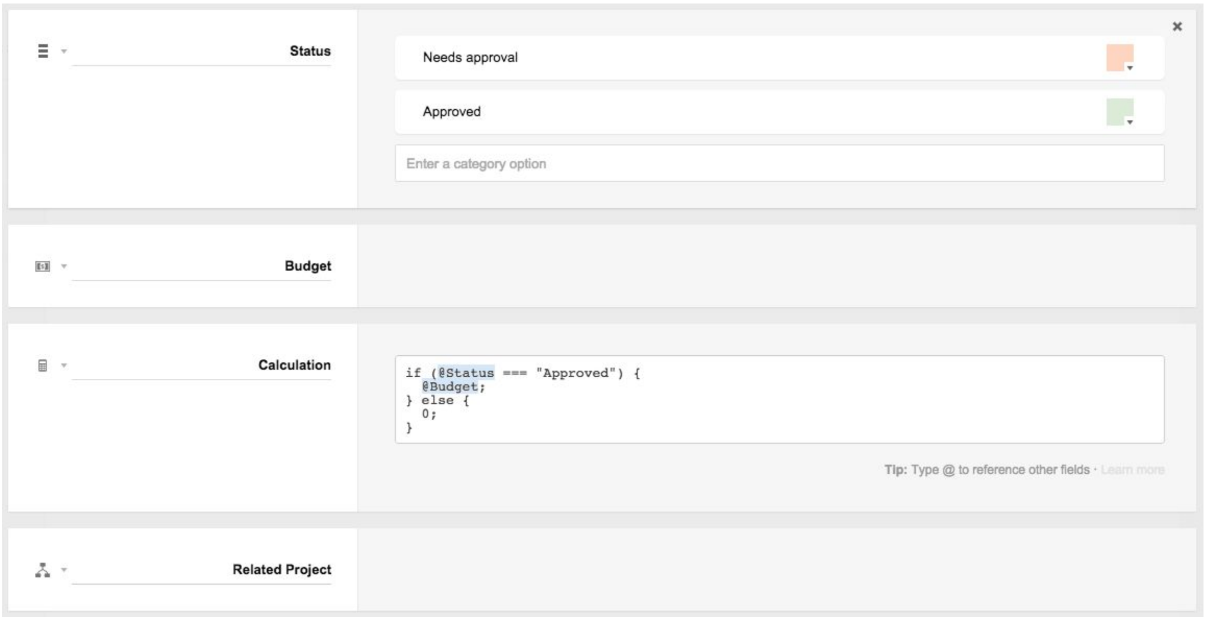
When setting the calculation up take note that “@” will display the name of the number-type field. In our example we’ve named this field Approved Budget.

It is also possible to use the “@” symbol to pull in variables from text fields. Calculations will output data in one of three different types: Number, Date, or Text. The output type is determined the first time you save your calculation field, based on what your calculation returns.

Calculation:

```
if (@Status === "Approved") {
  @Budget;
} else {
  0;
}
```

App template in Deliverables:



Status

Needs approval

Approved

Enter a category option

Budget

Calculation

```
if (@Status === "Approved") {
  @Budget;
} else {
  0;
}
```

Tip: Type @ to reference other fields · Learn more

Related Project

Else if statement

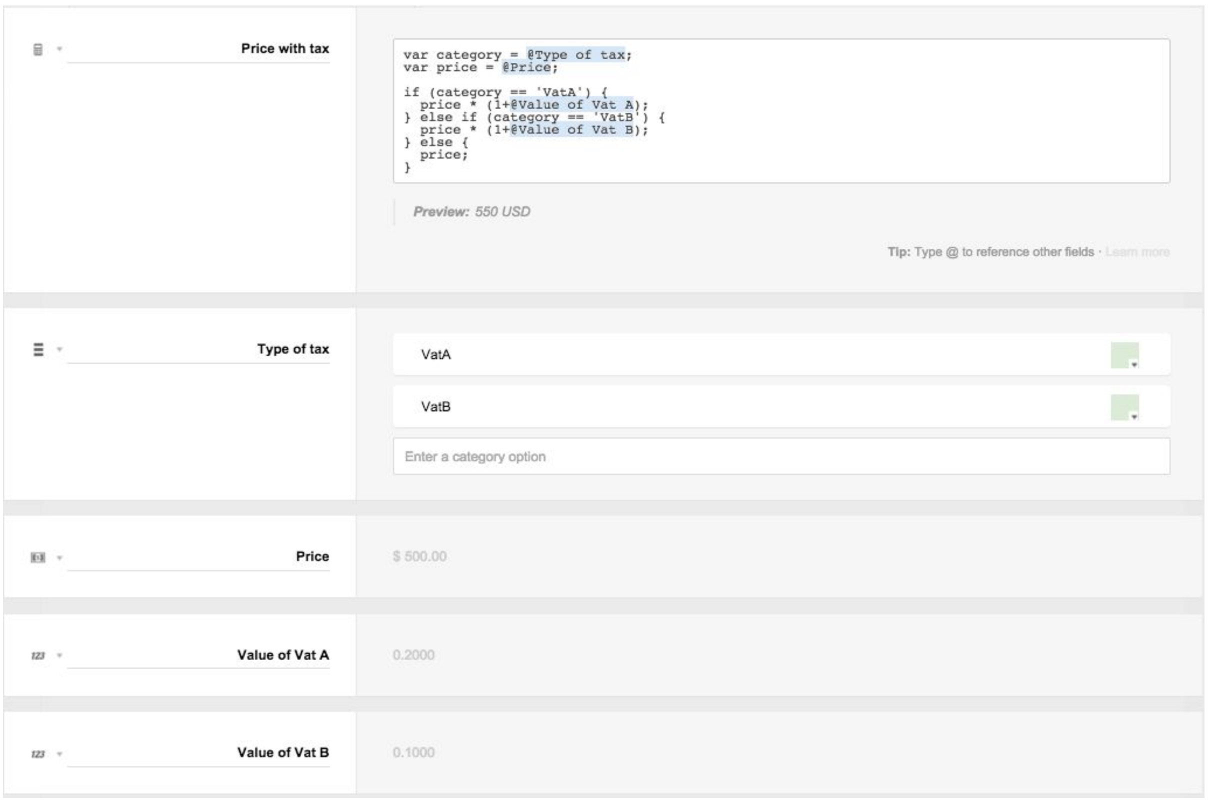
Use: You can specify a new condition if the first condition is false.

Example: In this scenario we would like to calculate the price including VAT where there are two VAT variables, VAT A and VAT B. For this calculation we need five fields: a money field for the price, two number fields for each of the VAT variables, a category field to select which VAT variable we would like to include in the calculation, and finally a calculation field to perform the calculation. When we want to calculate the price including VAT A we can then simply select VAT A in the category field to trigger the calculation and return a price that includes VAT A.

Calculation:

```
var category = @Type of tax;

var price = @Price;

if (category == 'VatA') {

price * (1+@Value of Vat A);

} else if (category == 'VatB') {

price * (1+@Value of Vat B);

} else {

price;

}
```

App template:



Price with tax

```
var category = @Type of tax;
var price = @Price;

if (category == 'VatA') {
  price * (1+@Value of Vat A);
} else if (category == 'VatB') {
  price * (1+@Value of Vat B);
} else {
  price;
}
```

Preview: 550 USD

Tip: Type @ to reference other fields · Learn more

Type of tax: VatA, VatB, Enter a category option

Price: $ 500.00

Value of Vat A: 0.2000

Value of Vat B: 0.1000

If and statement

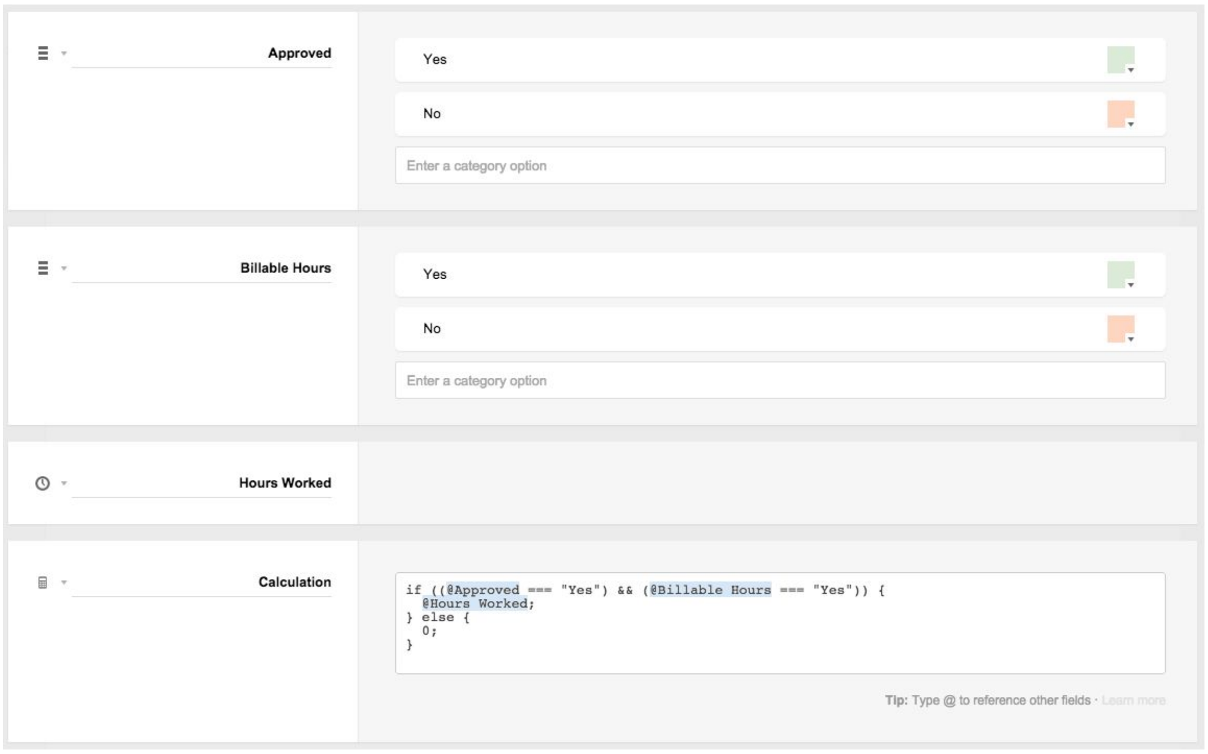
Use: You can specify more conditions that need to be met at the same time.

Example: In this scenario we work on some customer projects and we would like to bill our customers only for those hours that have been approved by our manager. In our app template we have included four fields: a category field to indicate whether the hours worked are considered billable, a duration field to log the hours spent on the task, another category field to indicate whether the task has been approved by a manager or not, and finally a calculation field to show the billable hours. In this example we’ve named these fields “Approved”, “Billable hours”, “Hours worked” and we’ve named our calculation field “Billable hours”. This calculation will only calculate billable hours if both category fields have the “yes” box checked.

Calculation:

```
if ((@Approved === "Yes") && (@Billable hours === "Yes")) {
  @Hours worked;
} else {
  0;
}
```

App template:



Approved	Yes 🟩 No 🟧 Enter a category option
Billable Hours	Yes 🟩 No 🟧 Enter a category option
Hours Worked	
Calculation	see code below

```
if ((@Approved === "Yes") && (@Billable Hours === "Yes")) {
  @Hours Worked;
} else {
  0;
}
```

Tip: Type @ to reference other fields · Learn more

Calculations with text fields

Concatenate two text fields

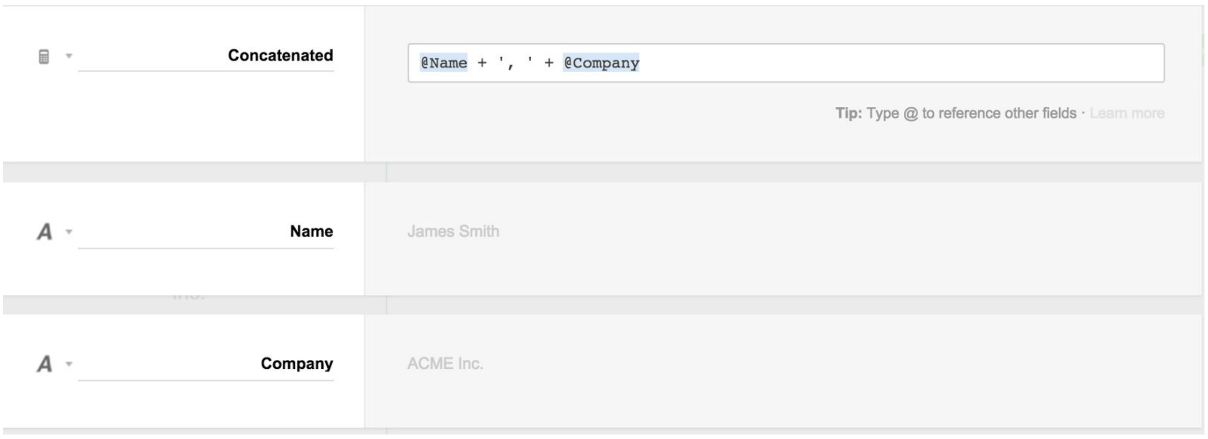
Use: Display data from two text fields in a calculation field.

Example: Show a contact's name and company in one field, divided by a comma. You can of course use a different separator by changing the comma to something else.

Calculation:

```
@Name + ', ' + @Company
```

App template:



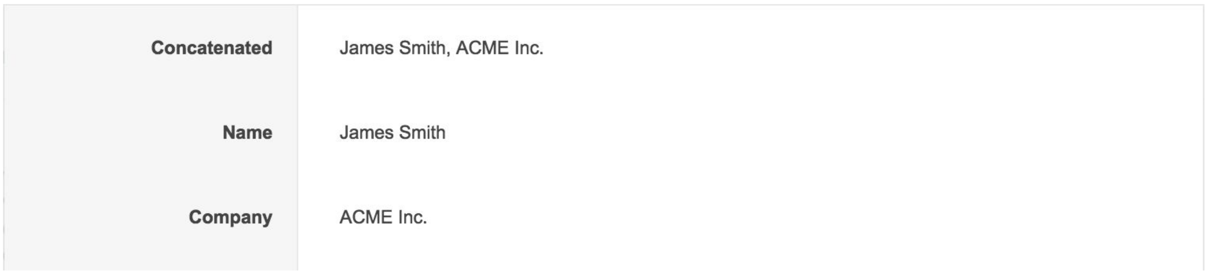
Concatenated

```
@Name + ', ' + @Company
```

Tip: Type @ to reference other fields · Learn more

Name	James Smith
Company	ACME Inc.

Result:



Concatenated	James Smith, ACME Inc.
Name	James Smith
Company	ACME Inc.

Concatenation with null values

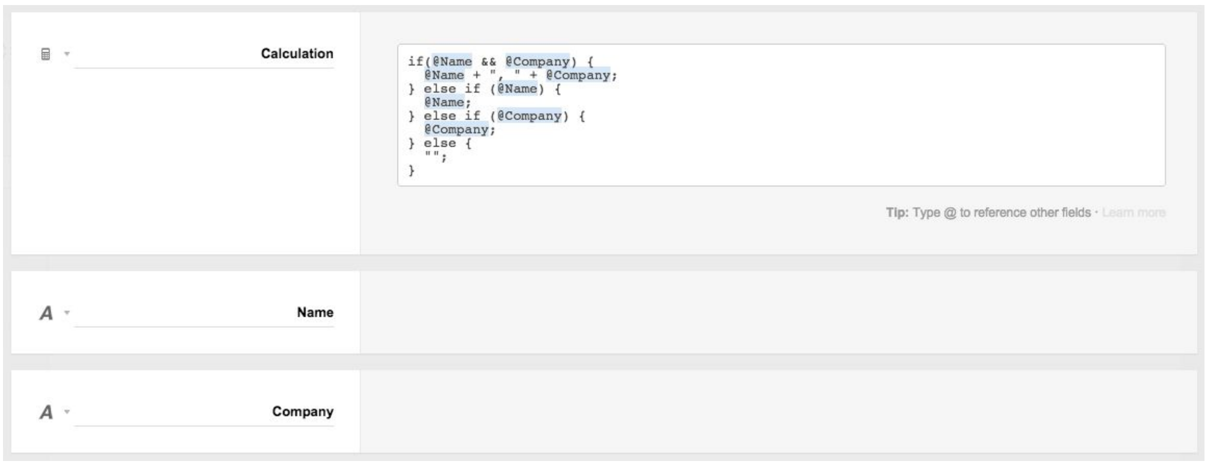
Use: In order to concatenate two strings together where one or both may have a null value, use this calculation.

Example: In this scenario we have two separate text fields (name and company) that we would like to have displayed together in a single field. This can be achieved by using a calculation field. We would also like the calculation to omit any empty fields and not return a "null" value in the concatenation. We set this calculation up to show values only from those fields that contain a value.

Calculation:

```
if(@Name && @Company) {
  @Name + ", " + @Company;
} else if (@Name) {
  @Name;
} else if (@Company) {
  @Company;
} else {
  "";
}
```

App template:



Calculation

```
if(@Name && @Company) {
  @Name + ", " + @Company;
} else if (@Name) {
  @Name;
} else if (@Company) {
  @Company;
} else {
  "";
}
```

Tip: Type @ to reference other fields · Learn more

Name

Company

Calculations with relationships

Number of relationships

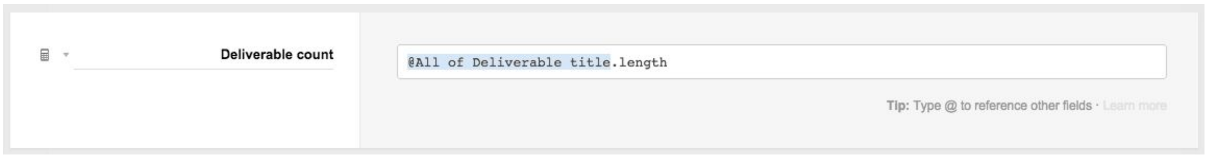
Use: If you want to know how many related items you have attached to an item.

Example: In this scenario we would like to rank projects in our projects app based on the number of deliverables (in our deliverables app) attached to each project. This is possible because we have related these two apps using a relationship field so we can pull in the variables from related apps using the "@" symbol in the calculation. We can then use this calculation to get a figure on which we'll create a report to save to our workspace homepage. Our calculation counts the deliverable titles attached to a project. Here "Deliverable title" is the name of the first field in your related items (in the Deliverables app).

Calculation:

@All of Deliverable title**.length**

App template:

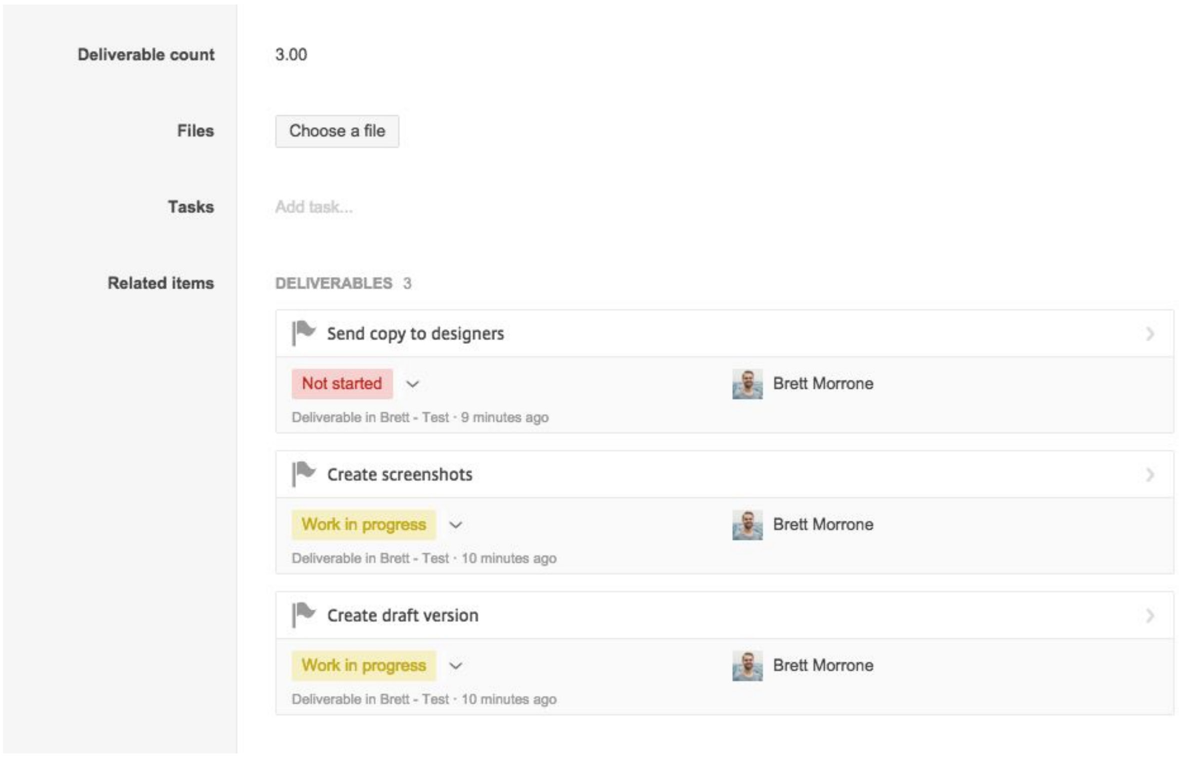


Deliverable count

```
@All of Deliverable title.length
```

Tip: Type @ to reference other fields · Learn more

Result:



Deliverable count	3.00
Files	Choose a file
Tasks	Add task...
Related items	DELIVERABLES 3

Send copy to designers
Not started — Brett Morrone
Deliverable in Brett - Test · 9 minutes ago

Create screenshots
Work in progress — Brett Morrone
Deliverable in Brett - Test · 10 minutes ago

Create draft version
Work in progress — Brett Morrone
Deliverable in Brett - Test · 10 minutes ago

Calculations with dates and numbers

Duration between two dates

Use: You can calculate the duration between two date fields.

Example: In this example we would like to calculate the duration in days of a completed task. We can achieve this by using a calculation field and two separate date fields for the start and end date. We can then subtract the start-time from the end-time. The calculation will return a result in milliseconds so we also need to add a division in the calculation for seconds, minutes, hours, and days.

/1000 → seconds

/60 → minutes

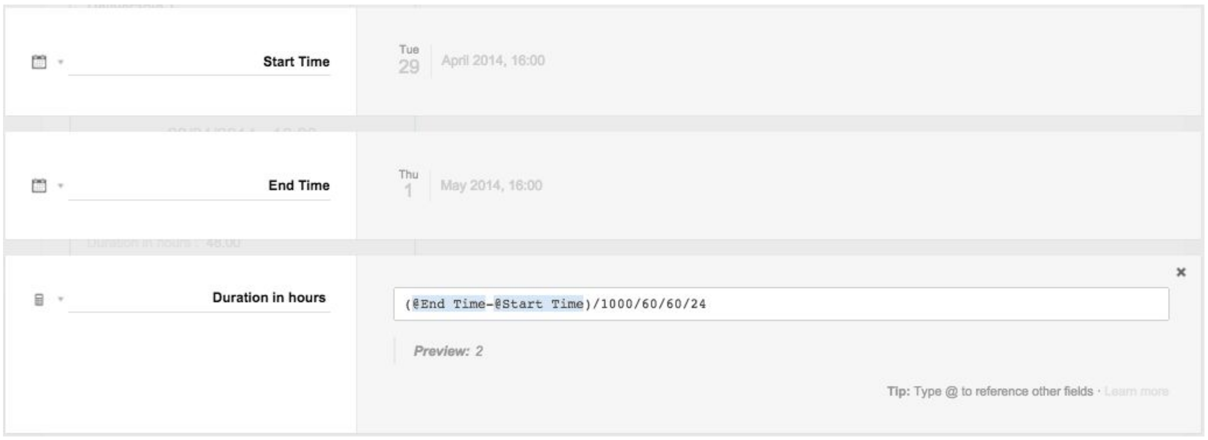
/60 → hours

/24 → days

Calculation:

(@End time - @Start time)/1000/60/60/24

App template:



Start Time — Tue 29 April 2014, 18:00

End Time — Thu 1 May 2014, 16:00

Duration in hours

```
(@End Time-@Start Time)/1000/60/60/24
```

Preview: 2

Tip: Type @ to reference other fields · Learn more

Estimate end date

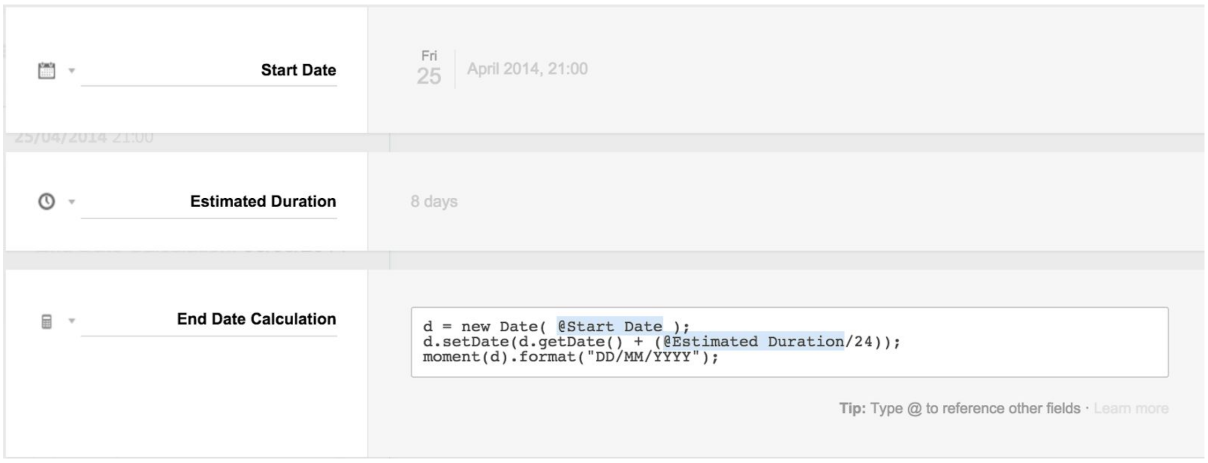
Use: Estimate end date based on a duration.

Example: In this scenario we would like to calculate an expected end date based on a start-date and the expected duration for the project. We need three fields: a date field for the start date, a duration field for the estimated duration, and the calculation field to calculate the estimated end date.

Calculation:

```
d = new Date( @Start Date );

d.setDate(d.getDate() + (@Estimated Duration/24));

moment(d).format("DD/MM/YYYY");
```

App template:



Start Date

Fri 25 April 2014, 21:00

Estimated Duration

8 days

End Date Calculation

```
d = new Date( @Start Date );
d.setDate(d.getDate() + (@Estimated Duration/24));
moment(d).format("DD/MM/YYYY");
```

Tip: Type @ to reference other fields · Learn more

Decimal values

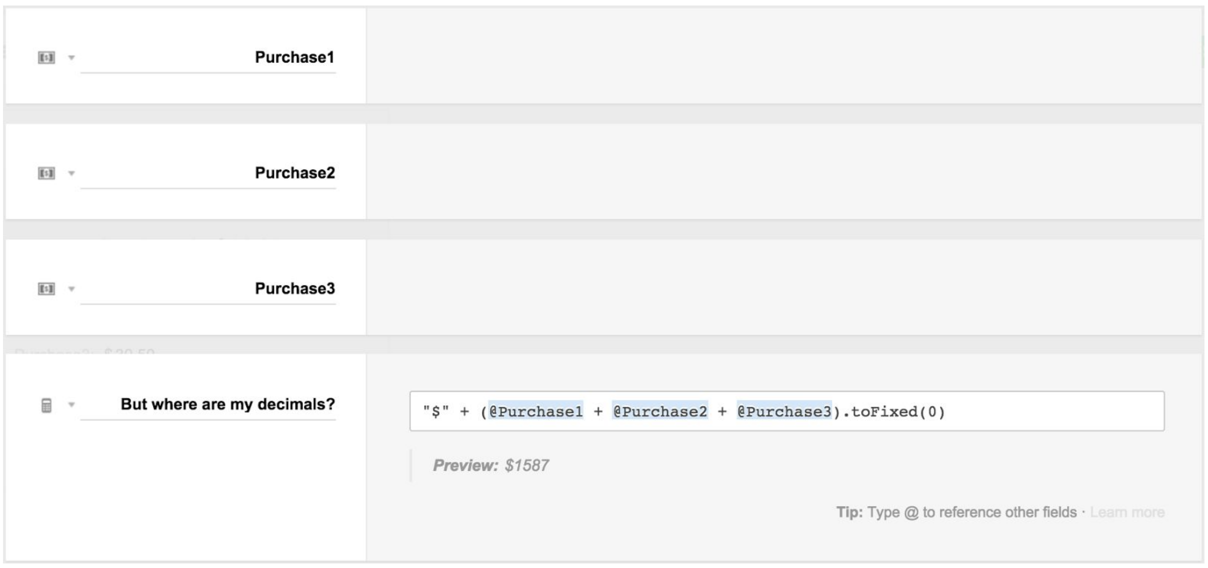
Use: You can determine how many decimals to show after a number.

Example: In this example we would like our calculation to omit decimals. In the calculation we've added .toFixed(0) the "0" in brackets indicates that no decimals will be shown. You can replace the "0" with a number to indicate how many decimals you would like displayed.

Calculation:

"$" + (@Purchase1 + @Purchase2 + @Purchase3)**.toFixed(0)**

App template:


Purchase1
Purchase2
Purchase3
But where are my decimals?

```
"$" + (@Purchase1 + @Purchase2 + @Purchase3).toFixed(0)
```

Preview: $1587

Tip: Type @ to reference other fields · Learn more

Other useful calculations

Section Divider

Use: Create different sections in your item to get a better overview of your data.

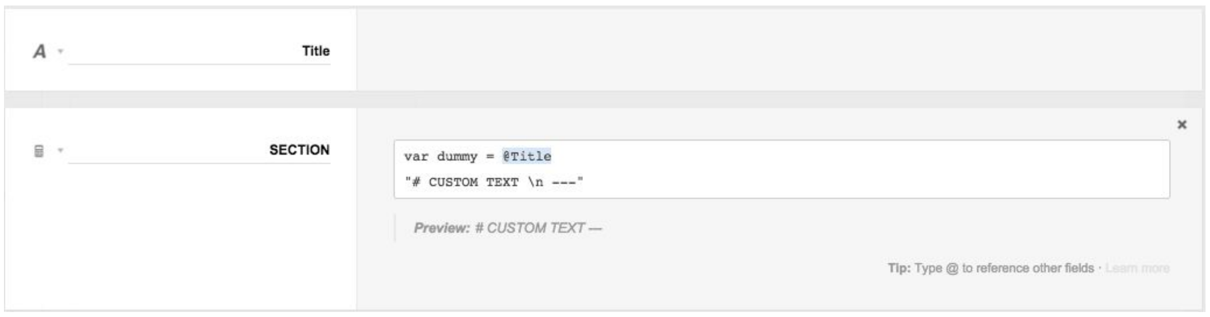
Please note, this is a workaround and therefore it is not supported well on mobile. It's also worth noting that calculation fields do not work in webforms.

Example: In the calculation field, it is a prerequisite that you reference a field in an app. Therefore you will need to use a dummy variable. In the example below we've used our title field as the dummy variable. The dummy variable can be any random field in your template and "CUSTOM TEXT" can be anything you want as the section title. You can also give the calculation field a name, in our example we named it "SECTION".

Calculation:

```
var dummy = @Title

"# CUSTOM TEXT \n ---"
```

App Template:

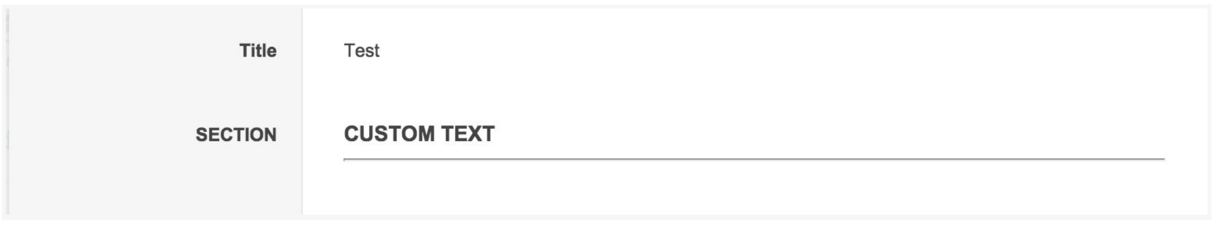


```
var dummy = @Title
"# CUSTOM TEXT \n ---"
```

Preview: # CUSTOM TEXT —

Tip: Type @ to reference other fields · Learn more

Result:



Title	Test
SECTION	CUSTOM TEXT

Synced Unique IDs across apps

Use: Create synced unique IDs across apps (A-001 → B-001). The number digits are the same across the related app items, just the first letter changes according to the name of the app. The 3 digits are basically the unique ID (it's set to have 3 digits, so instead of 1, it's 001, so it can be easily ranked in Excel).

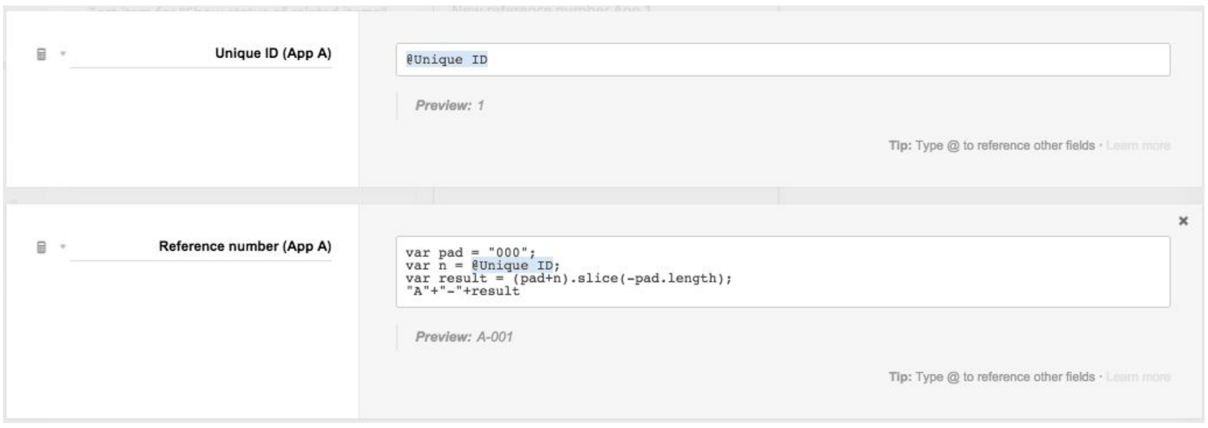
Example: A quote turns into a job, that turns into a deliverable etc. and we would like to keep the same reference number across the apps, just with a different prefix, so we can easily track the different items. We have to create a calculation field in App A to display the Unique ID, so we will be able to reference this in the calculation in App B.

Calculation:

App “A” reference number:

```
var pad = "000";

var n = @Unique ID;

var result = (pad+n).slice(-pad.length);

"A"+"-"+result
```

App template:


Unique ID (App A)

```
@Unique ID
```

Preview: 1

Tip: Type @ to reference other fields · Learn more

Reference number (App A)

```
var pad = "000";
var n = @Unique ID;
var result = (pad+n).slice(-pad.length);
"A"+"-"+result
```

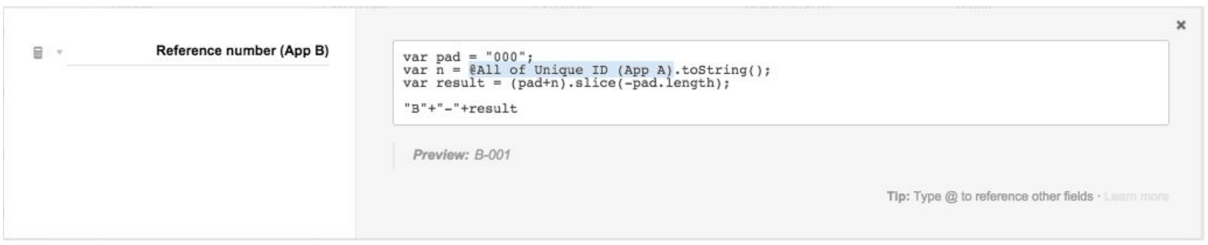
Preview: A-001

Tip: Type @ to reference other fields · Learn more

App “B” reference number:

```
var pad = "000";

var n = @All of Unique ID (App A).toString();

var result = (pad+n).slice(-pad.length);

"B"+"-"+result
```

App template:


Reference number (App B)

```
var pad = "000";
var n = @All of Unique ID (App A).toString();
var result = (pad+n).slice(-pad.length);
"B"+"-"+result
```

Preview: B-001

Tip: Type @ to reference other fields · Learn more

Limitations

Calculation fields are only updated when data in the fields they reference is altered.

As an example, the calculation x + y will be updated every time x or y changes. Now consider the calculation: Current hour + x. The intent of this calculation is to show 9 + x when the time is between 9:00 and 9:59, 10 + x when the time is between 10:00 and 10:59, etc. The problem is that the calculation will only be updated when x changes. If x does not change, then the calculation field will show whatever the hour was when x last changed + x.

The reason for this is to avoid updating calculation fields all the time. Keeping such calculations up to date would require an enormous amount of computing resources.

Alternatively, you can set it up through the API or set up a "Current-time" app where you can update the current time that updates all the referenced calculations.