Piping is a feature that allows you to display data collected elsewhere in your REDCap project. Piping is only possible within the same project.

Piping can be used to support a wide variety of research activities. With piping, people can view information without having to open additional forms and search data fields. For example, you can pipe information, such as sub-study participation, that is captured at consent onto a subsequent visit form, so a research coordinator will know to perform additional study procedures. Piping can also be used to personalize text in survey invitations.

Piping can be used to insert data in the following locations:

1. Field Labels
2. Field Notes
3. Section Headers
4. Matrix field column headers
5. Option labels for multiple choice fields (radio, drop-down, checkbox)
6. Slider field labels (i.e. text displayed above slider bar)
7. Survey invitations - subject line and body of message
8. Survey instructions and acknowledgements
9. Custom text displayed at top of survey queue
10. Inside the URL for a survey’s 'redirect to a URL' setting
11. Inside the @DEFAULT action tag
12. Custom record locking text (if defined, displayed at bottom of form)

How to Utilize Piping

To pipe data, simply insert the variable name of the field you wish to display inside square brackets [x] in the target location.

**Pro Tip**: When piping from multiple fields, open the Codebook on one tab and the design page on another. This will allow you to view the variable names and prevent errors. The Ctrl+F function can be used to search the Codebook.

For an online demonstration visit: [https://is.gd/PipingTutorial](https://is.gd/PipingTutorial). See examples below for piping scenarios.
Example #1: Piping from a Text Field

If you have a survey question 'What is your first name?' with the variable name 'first_name', you can display the respondent’s name to a follow-up question with piping.

To pipe, insert first_name inside a set of brackets in the field label of the follow-up question. In this example, ‘What is your date of birth?’:

In Example #1 if the respondent entered ‘John’ as their first name, the follow-up question would read ‘John, what is your date of birth?’.

Example #2: Piping from a Multiple-Choice Field (radio, drop down)

If you have a multiple-choice question 'What is your favorite ice cream?' with the response options 'Chocolate', 'Vanilla', and 'Strawberry' (coded as 0, 1, 2) under the variable name ‘fav_icecream’, you can use piping with a follow-up question.

If you are piping from a multiple-choice field, it will display the option label (not the coded data value).
To pipe, insert the variable name `fav_icecream` inside a set of brackets within the field label of the follow-up question.

In this example, “How much do you love [x]?”

After selecting ‘Chocolate’, the follow-up question would be displayed as ‘How much do you love Chocolate ice cream?’
Example #3: Piping a Coded Value of a Multiple-Choice Field

If you wish to pipe the value (not the label) of a multiple-choice field, append `.value` to the variable name inside brackets, e.g. `[option:value]`.

![Diagram of multiple-choice options]

Note: The usage of `.value` must be used if piping inside the @DEFAULT Action Tag if you would like to pipe the value of a multiple-choice field as the default value of a field.

Example #4: Piping to a Different Event in a Longitudinal Project

To pipe data from another event within a longitudinal project, prepend the variable name with the unique event name inside square brackets (as shown below).

Piped field `(piped_v1_date)` is set up as a descriptive field to be displayed in Visit 2 event:

![Diagram of piping to a different event]

Note: For longitudinal projects, unique event names can be found in Project Setup -> Define My Events.
For the example above, the Visit 1 Date is entered on the Visit 1 event:

<table>
<thead>
<tr>
<th>Event Name: Visit 1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Record ID</td>
<td>1</td>
</tr>
<tr>
<td>Visit Date</td>
<td>08-01-2020</td>
</tr>
</tbody>
</table>

With piping, the Visit 1 Date is then displayed on the Visit 2 event:

<table>
<thead>
<tr>
<th>Event Name: Visit 2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Record ID</td>
<td>1</td>
</tr>
<tr>
<td>Visit 1 Date: 08-01-2020</td>
<td></td>
</tr>
<tr>
<td>Visit Date</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** If the unique event name is not prepended, it will pipe the data value from the current event.

**Example #5: Piping from Checkbox Fields**

Piping from checkbox fields differs from other field-types. Checkboxes allow for multiple saved values.

**Options to display a list of checkbox fields include:**

a. **Checked choices**
   i. `[my_checkbox:checked]`
   ii. Appending ':checked' will display a comma-delimited list of choice labels that have been checked - e.g. 'Sunday, Tuesday, Thursday'.
   iii. **Note:** If neither ':checked' nor ':unchecked' is appended to the variable, then it will default to ':checked'.

b. **Unchecked choices**
   i. `[my_checkbox:unchecked]`
   ii. Appending ':unchecked' will display a comma-delimited list of choice labels that have NOT been checked - e.g. 'Monday, Wednesday, Friday, Saturday'.

c. **Specific choices**
   i. `[my_checkbox(code)]`
   ii. If a coded value of the checkbox is included inside parentheses after the variable name - e.g. `[my_checkbox(2)]` - then it will output the word 'Checked' or 'Unchecked'.
Example of piping from check box fields:

Example above will display as follows:

Pro Tip: The checkbox piping options listed above will return the text labels. However, you can append ':value' to the variable to return the raw value.

Example: [my_checkbox:checked:value] and [my_checkbox:unchecked:value] could return '1, 3, 5' and '2, 4, 6, 7', respectively, and [my_checkbox(2):value] will return 1 or 0 if checked or not checked.