Validate one number is less than another

Scripting Solutions

Additional scripting solutions will be added in the future. Please reach out to Alchemer with comments and suggestions on solutions you'd like to see via the link here.

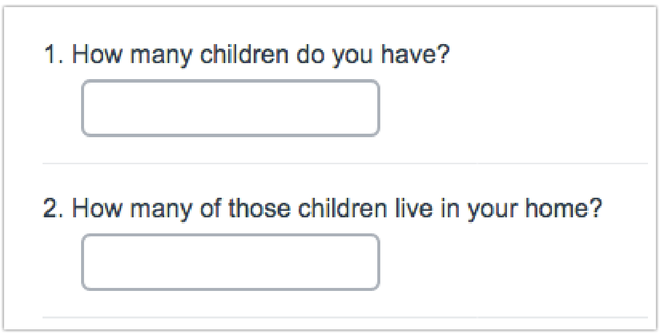
Scripting and Other Custom Solutions

We're always happy to help you debug any documented script that is used as is. That said, we do not have the resources to write scripts on demand or to debug a customized script.

If you have customization ideas that you haven't figured out how to tackle, we're happy to be a sounding board for Alchemer features and functionality ideas that might meet your needs. Beyond this, check out our Professional Services; these folks have the scripting chops to help you to achieve what you are looking for!

Goal

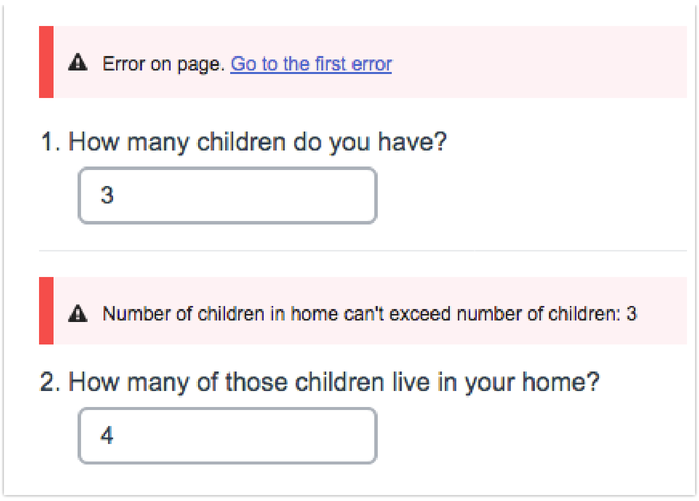
Ensure a value is less than (or equal to) another value, for example:



1. How many children do you have?

2. How many of those children live in your home?

⇩ ⇩ ⇩



⚠ Error on page. Go to the first error

1. How many children do you have?

3

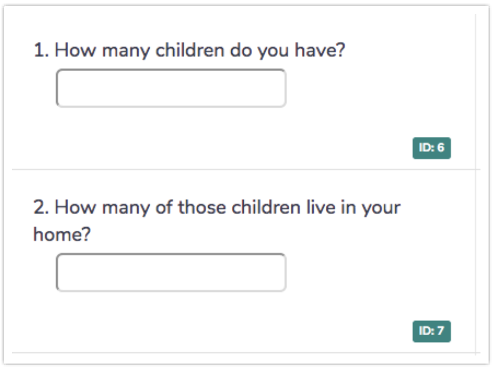
⚠ Number of children in home can't exceed number of children: 3

2. How many of those children live in your home?

4

Solution

Step 1: Add two Textbox questions on the same page or different pages with validation to require numbers or currency.

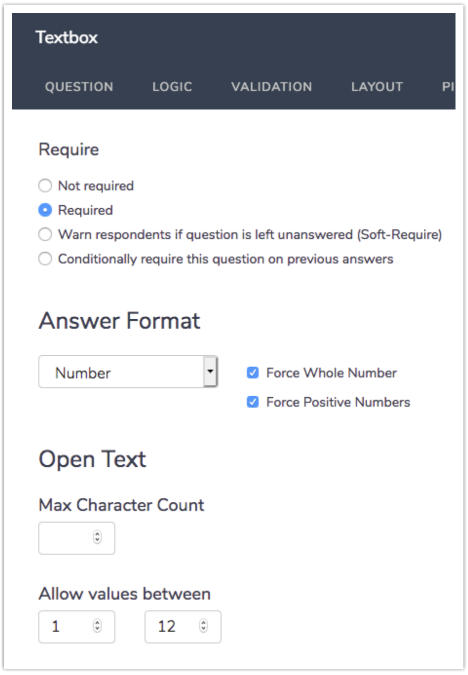


1. How many children do you have?

ID: 6

2. How many of those children live in your home?

ID: 7



Require

- ○ Not required
- ● Required
- ○ Warn respondents if question is left unanswered (Soft-Require)
- ○ Conditionally require this question on previous answers

Answer Format

Number

- ✓ Force Whole Number
- ✓ Force Positive Numbers

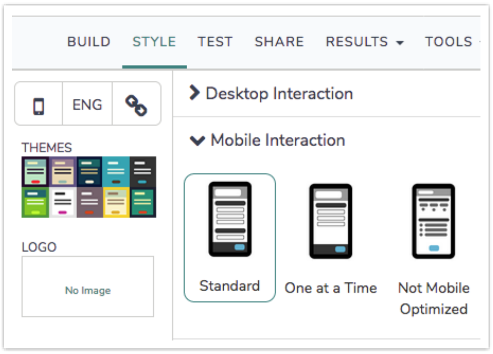
Open Text

Max Character Count

Allow values between

1 12

Step 2: Set Style > Layout > Mobile Interaction > Standard.



Step 3:

1. Add a new Javascript Action with the code below to the same page as the second question.
2. Set the highlighted values where the MUST_BE comparison operators are:

<	X must be less than Y
<=	X must be less than or equal to Y
=	X must equal Y
<>	X must not equal Y
>	X must be greater than Y
>=	X must be greater than or equal to Y

Option: If you have more complex validation needs Javascript developers can replace functions getX() or getY() to get the values from another question type or a calculation.

The Javascript code

```
/* Alchemer v01

   Client-side error handling, prevent page submission if criteria not met.

   Documentation and updates: https://help.alchemer.com/help/validate-one-number-is-less-than-another
```

```
*/

document.addEventListener("DOMContentLoaded", function() {

  const X_VALUE_OR_QID  = `[question("value"), id="6"]` || `~6` // fill in the Question ID twice
  const MUST_BE = '>=' // <, <=, =, <>, >, >=
  const Y_QID = 7  // Question ID on this page

  const Y_ERROR_MSG = "Number of children in home can't exceed number of children: "

  // ********************************
  // *** no changes needed below ***
  // ********************************

  const LOG = false

  /***
   * Test boolean value, alert() and throw Error if it's false
   *
   * bool (t/f) value to test
   * msg (string) message to alert and throw in new Error
   */
  const assert = (bool, msg) => {
    msg = "Javascript Error:\n\n" + msg
    if (!bool) {
      alert(msg)
      console.error(msg)
      const err = new Error(msg)
      console.error(err)
      throw err
      }
  }

  /***
   * Get an element based on its Question ID
   *
   * qid (int/string) question ID
   * section = "element" (string) the final section of the element id
   * return (element) looks for id's in the form: "sgE-1234567-12-123-element"
   */
  const getElemByQid = (qid, section = "element") => {
    const id = "sgE-" + SGAPI.survey.surveyObject.id + "-" + SGAPI.survey.pageId + "-" + qid + "-" + section
    const elem = document.getElementById(id)
    assert(elem, "Javascript: can't find element with id = " + id + ", section = " + section)
    return elem
  }

  /***
   * Clear all error messages on page
   */
  function clearErrors() {
    let errorElems = document.querySelectorAll(".sg-question-errorlist")
    errorElems.forEach(errorElem => errorElem.remove())


    errorElems = document.querySelectorAll(".sg-error-message")
    errorElems.forEach(errorElem => errorElem.remove())
  }

  /***
   * Set the top of page error message
   *
   * message (string)
   */
  function showTopErrorMessage(message) {
    // insert the error message at top of page
    let errorHTML = '
' + message + ' Go to the first error
'
    let sgContent = document.querySelector(".sg-content")
    sgContent.insertAdjacentHTML("afterbegin", errorHTML);

    // scroll up to show message
    sgContent.scrollIntoView()
  }

  /***
   * Set the question error message
   *
   * qid (int/string) question ID to display message before
   * message (string)
   */
  function showQuestionErrorMessage(qid, message) {
    const elemQuestion = getElemByQid(qid, "box")
    assert(elemQuestion, "Javascript error: missing question with qid: " + qid)

    // insert the error message at the top of the question
    let errorHTML = '
     •  ' + message + '

'
    elemQuestion.insertAdjacentHTML("afterbegin", errorHTML)
  }

  /***
   * showErrorMessages
   *
   * qid (int/string)
   * message (string)
   */
  let showErrorMessages = (qid, message) => {
    clearErrors() // Clear previous error messages
    showTopErrorMessage("Error on page.")
    showQuestionErrorMessage(qid, message)
  }

  /***
   * Get float value from question ID, stripping currency symbol
   *
   * qid (int/string) question ID
   * return (float) value or 0 if qid was blank
   */
  const getValueFromQID = (qid) => {
    // strip leading currency and anything except negative sign, digit, or decimal
    const num = getElemByQid(qid).value.replace(/[^-\d.]/g, "")
    return parseFloat(num) || 0
  }

  /***
   * Get X value
   * You can customize this to do something other than use X_VALUE_OR_QID
   *
   * return (float)
   */
  let getX = () => {
    if (X_VALUE_OR_QID.slice(0,1) === '~')
      return getValueFromQID(X_VALUE_OR_QID.slice(1))
      // value was from previous page and initialized with a merge code
      return parseFloat(X_VALUE_OR_QID) || 0
    }

    /***
     * Get Y value
     * You can customize this to do something other than use Y_QID
     *
     * return (float)
     */
    let getY = () => {
      return getValueFromQID(Y_QID)
    }

    /***
     * test
     */
    const test = (x, mustBe, y) => {
      if (LOG) console.log("test() ", x, " must be", mustBe, y)
      switch (mustBe) {
        case '<':  return x < y
        case '<=': return x <= y
        case '=':  return x === y
        case '<>':  return x !== y
        case '>':  return x > y
        case '>=': return x >= y
        default: assert(false, "Unknown comparison operator: " + mustBe)
      }
    }

    /***
     * main()
     */

    let elemNext = document.getElementById("sg_NextButton") || document.getElementById("sg_SubmitButton")

    // onclick Next/Submit
    elemNext.addEventListener("click", function(e) {

      if (LOG) console.log(`${getX()} must be ${MUST_BE} ${getY()}`)

      // if nothing was entered for X and Y, do nothing (let the validation on those questions generate error messages if
  needed)
      if (!getX() && !getY()) {
        if (LOG) console.log("do nothing")
        return true
      }

      if (!test(getX(), MUST_BE, getY())) {
        showErrorMessages(Y_QID, `${Y_ERROR_MSG}${getX()}`)
        e.preventDefault()
        return false
      }
    })
  })
```

Related Articles