Javascript snippets

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!

Most of the following snippets are designed to be used in Javascript Actions, **not** as stand-alone scripts. **All scripts should wait** for the page to load before running:

```
document.addEventListener("DOMContentLoaded", function() {
  // your code here...
})
```

Snippet	Description
Auto-submit page	Submit the page after completing a Javascript task
Buttons - change text	Change the text of the Back, Next, or Submit Button on a single page
Buttons - hide	Hide the Back or Next Button on a single page
getCheckedByQid()	Get the Reporting Values for the selected radio button or checkboxes
getElemByTitle()	Get HTML element for a Question or Hidden Value Action title
getElemByQid()	Get HTML element for a Question ID
getPid()	Get current Page ID
getQidByTitle()	Get Question ID for a Question or Hidden Value Action title

getReportingValuesByQid()	Get Option IDs and Reporting Values for a radio button or checkbox question
isMobile()	Is the survey presenting a mobile-optimized view
parseSgId()	Parse an Alchemer element ID, like sgE-5901811-28-305-10997-element
setCheckedByQid()	Check or uncheck Radio Button or Checkbox question options
SGAPI	Survey metadata object

Auto-submit page

Submit the page after completing a Javascript task

The Page Setting "**Automatically submit the page, running all actions and custom scripts**" **does not allow** Javascript to run on the page due to the page being processed on the server and never appearing in the respondent's browser, where Javascript runs.

The Javascript below submits the page after your code completes. Add **sg-hide** to the page's **Layout > CSS Class Name** so the page won't 'flash' to the respondent, instead appearing as a completely blank screen for a brief moment.

> When using **Preview Mode** choose Fire Actions to ensure this Javascript submits the 'blank' page, otherwise the Preview remains on the 'blank' page.

This is a stand-alone script.

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

 /***
  * Check if this page is being shown because user clicked the Back button on the following page.
  *
  * return (t/f) true if moving back
  */
 const isMovingBack = () =>
   SGAPI.surveyData[Object.keys(SGAPI.surveyData)[0]].page_direction === -2

 /***
  * Click Back or Next button
  */
 const back = () => document.querySelector("#sg_BackButton").click()
 const next = () => (document.querySelector("#sg_NextButton") || document.querySelector("#sg_SubmitButton")).clic
k();

 /***
  * main()
  */

 // Moving back?
 if (isMovingBack()) {
   back()
   return
 }

 // your code here...

 next()
})
```

Buttons - Change text

Change the text of the Back, Next, or Submit Button on a single page

This is a stand-alone script. Decide which button text to change and remove the other lines of code. Note: This does not take translations into consideration and will make the change for all languages used in the Text and Translations feature.

```
document.addEventListener("DOMContentLoaded", function() {
 document.querySelector('#sg_BackButton').value   = 'New Back text'
 document.querySelector('#sg_NextButton').value   = 'New Next text'
 document.querySelector('#sg_SubmitButton').value = 'New Submit text'
})
```

Buttons - Hide

Hide the Back or Next Button on a single page

This is a stand-alone script. Decide which button to hide and remove the other line of code.

```
document.addEventListener("DOMContentLoaded", function() {
  document.querySelector('#sg_BackButton').style.display = "none"
  document.querySelector('#sg_NextButton').style.display = "none"
})
```

getCheckedByQid()

Get the Reporting Values for the selected radio button or checkboxes.

Note: This script requires getElemByQid()

```
/***
 * Get the reporting values for the CHECKED options of
 * a radio button or checkbox QID.
 *
 * qid (int / string) question ID
 * return (array of string) array of reporting values
 */
const getCheckedByQid = (qid) => {
  const checkedElems = getElemByQid(qid, "box").querySelectorAll('.sg-question-options input:checked')
  const checkedOptionIDs = [...checkedElems].map(elem => elem.value)
  // get object that maps optionID to reporting value, ex:  { "10014": "reporting value1", "10015": "value2"}
  const optionsObj = SGAPI.survey.surveyObject.questions[qid].options
  // map the object array to array of just the reporting values
  const checkedReportingValues = checkedOptionIDs.map(optionId => optionsObj[optionId].value )
  return checkedReportingValues
}
```

Example, question ID 34 is a checkbox of colors:

```
console.log(getCheckedByQid(34))  // [ "red", "green", "blue" ]
```

getElemByTitle()

Get HTML element for a a Question or Hidden Value Action title

Example makes use of two other snippets:

```
const elem = getElemByQid( getQidByTitle( "What is your favorite color?" ) )
```

getElemByQid()

Get an HTML element based on a question ID

```
/***
 * 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 Assert Error: " + 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)
  return elem
}
```

Example. If you have a Textbox or Hidden Value Action you can get or set its value (assume the Textbox is question ID 34)

```
  const val = getElemByQID(34).value
  getElemByQID(34).value = val + '!!!'
  console.log(getElemByQID(34).value) // old value!!!
```

getReportingValuesByQid()

Get the option ids and reporting values for a radio button or checkbox

```
/***
 * Get the option ids and reporting values for a radio button or
 * checkbox question or grid from the SGAPI object
 *
 * qid (int / string) question ID
 * return (array of obj) array of option objects:
 *           [
 *             { optionId: "10014", reportingValue: "4" },
 *             { optionId: "10015", reportingValue: "3" }
 *           ]
 */
const getReportingValuesByQid = (_qid) => {
  assert(SGAPI.survey.surveyObject.questions[_qid], "Can't find qid on this page: " + _qid)

  let qid = _qid

  // if this is a grid question, get the reporting values from the first row
  if (SGAPI.survey.surveyObject.questions[qid].sub_question_skus)
    qid = SGAPI.survey.surveyObject.questions[qid].sub_question_skus[0]

  const optionsObj = SGAPI.survey.surveyObject.questions[qid].options
  assert(optionsObj, "QID isn't a radio button or checkbox question: " + qid)

  return Object.keys(optionsObj).map(optionId =>
    ({ optionId: optionId, reportingValue: optionsObj[optionId].value }) )
}
```

Example:

```
console.log("The option ids and reporting values for qid 23 are: ", getOptionReportingValues(23))
    // [ {"optionId":"10017","reportingValue":"Home"},
    //   {"optionId":"10018","reportingValue":"Work"},
    //   {"optionId":"10019","reportingValue":"Cell"} ]
```

getPid()

Get the current page ID

```
/***
 * Get the current Page ID
 *
 * return (int) the current page ID
 */
const getPid = () => parseInt(SGAPI.survey.pageId)
```

Example:

```
console.log("This current Page ID is: ", getPid()) // 13
```

getQidByTitle()

Get the Question ID for a Question / Hidden Value Action title

```
/***
 * Get the Question ID based on the question / hidden value title
 *
 * title (string) question title
 * return (int/null) the question ID or null if title not found
 */
const getQidByTitle = (title) => {
  const questionsObj = SGAPI.survey.surveyObject.questions
  const qid = Object.keys(questionsObj).find(key =>
    questionsObj[key].title.toLowerCase() === title.toLowerCase())
  return qid || null
}
```

Example:

```
console.log("QID for first name question:", getQidByTitle("First Name")) // 34
```

isMobile()

Is the survey presenting a mobile-optimized view.

```
/***
 * Determine if this is a mobile presentation
 *
 * return (t/f)
 */
const isMobile = () => document.querySelector('body.sg-mobile-optimized') !== null
```

Example:

```
console.log("Is this survey being presented mobile-optimized?: ", isMobile()) // true or false
```

parseSgId()

Parse an Alchemer element ID, like `sgE-5901811-28-305-10997-element`

```
/**
 * Parse an Alchemer element #ID in the form:
 *     sgE-5901811-28-305-box
 *     sgE-5901811-28-305-10997-element
 *     sgE-6231616-1-21-10021-0  ex: star in a star rating grid
 * return (obj) Returns object of the constituent parts
 */
const parseSgId = (id) => {
  const regexID = /sgE-(\d+)-(\d+)-(\d+)(-(\d+))?-(\w+)/


  const aParsed = id.match(regexID)


  if (!aParsed || aParsed.length !== 7) {
    alert('Javascript error parsing ID = ', id)
  }


  return {
    sid: aParsed[1], // ex: '5901811'
    pid: aParsed[2], // ex: '28'
    qid: aParsed[3], // ex: '305'
    oid: aParsed[5], // ex: '10997' or undefined if this isn't an ID for an option
    type: aParsed[6] // ex: 'box' / 'element'
  }
}
```

Example:

```
  console.log(parseSgId('sgE-5901811-28-305-10997-element'))
  /*
    {
      sid: '5901811',
      pid: '28',
      qid: '305',
      oid: '10997',
      type: 'element'
    }
  */
```

setCheckedByQid()

Check or uncheck Radio Button or Checkbox question options

```
/***
 * Set or clear selections for Radio Button or Checkbox question
 *
 * qid (int/string) the question ID of a radio button or checkbox question to clear
 * checkOptionsIds {int or array of int}
 *        a single OptionID, an array of OptionIDs to be checked, or [] to clear all selections
 */
const setCheckedByQid = (qid, _checkOptionIds) => {

  // convert param to an array if it was a single value
  let checkOptionIds = Array.isArray(_checkOptionIds) ? _checkOptionIds : [_checkOptionIds]

  // ensure array is all integers
  checkOptionIds = checkOptionIds.map(id => parseInt(id))

  // go through all options and check or uncheck them
  getElemByQid(qid, "box").querySelectorAll('.sg-question-options input')
    .forEach(inputElem =>
      inputElem.checked = checkOptionIds.includes(parseInt(inputElem.value)))
}
```

Example, assuming QID 22 is a Radio Button and QID 25 is a Checkbox question:

```
setCheckedByQid(22, 10016)         // check optionId 10016
setCheckedByQid(25, [10017, 10019]) // check optionIds 10017 and 10019, uncheck all others
setCheckedByQid(25, [])            // clear/uncheck all options
```

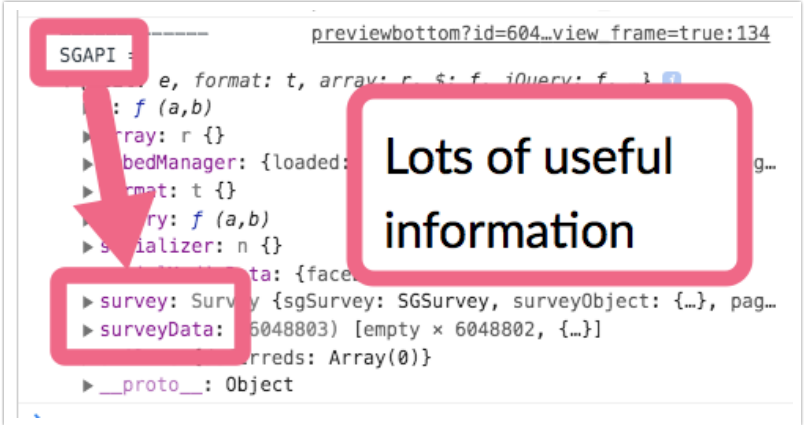
SGAPI

Survey metadata object

Example, type SGAPI in browser Console window to browse that object's survey and surveyData properties.

```
console.log(SGAPI.surveyData[Object.keys(SGAPI.surveyData)[0]].messages.required)
/* This question is required */
```

```
SGAPI
```



Related Articles