Autofill tool

> ### 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.

Goal

Autofill answers on a survey page to speed up your survey testing.

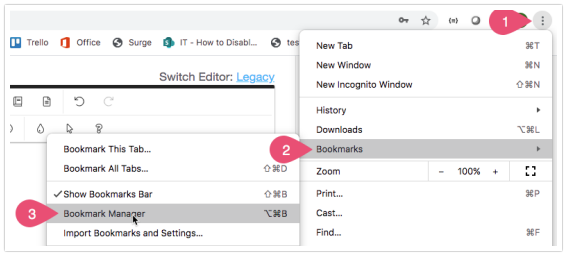
Effort: ✔ ✔ ✔

Solution

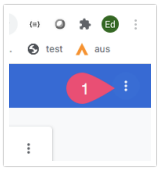
Create a bookmark in Chrome that runs Javascript to intelligently fill in a survey page.

To add the Autofill tool:

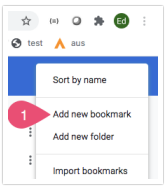
(1) In Chrome, choose the **three vertical dots in the upper right > Bookmarks > Bookmark Manager**



(2) Choose the **three dots in the upper right below the original three dots**

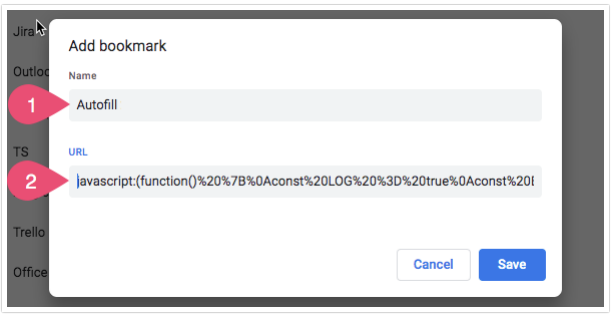


(3) Choose **Add new Bookmark**.



(4) Enter the name **Autofill**.

(5) **Paste** the code below in green for the URL.



To use the Autofill Tool:

When on a survey page click the **Autofill** bookmark and watch the page fill in.

Autofill works with the following question types:

- checkbox (with optional min number of selections or optional write-in)
- checkbox grid (with optional min number of selections)
- conjoint
- continuous sum (with optional required total)
- essay
- dropdown
- dropdown menu grid
- dropdown menu list
- image multi select (but doesn't look at min number...yet)
- image select
- likert
- max diff
- NPS®
- radio button (with optional write-in)
- radio button grid
- ranking grid
- semantic diff
- slider (optional min/max and step)
- slider list (same as above)
- star rating grid
- textbox
  - email
  - date (all three formats but not min/max date)
  - numeric with min/max number or min/max number of characters
  - currency
  - percent
  - phone (based on the regex option)
- textbox grid (same as above)
- textbox list (same as above)
- Custom Groups containing any of the above

Code to add as a bookmark (copy everything in green below and paste it in as a new Chrome bookmark):

```
javascript:%2F*%20Alchemer%20v03%0A%0A%20%20%20Autofill%20Chrome%20Bookmarklet%0A%0A%20%20%20Works%20with%20question%20types%3A%0A%20%20%20%20%20%2
in)%0A%20%20%20%20%20%20checkbox%20grid%20(with%20optional%20min%20number%20of%20selections)%0A%20%20%20%20%20%20conjoint%0A%20%20%20%20%20%20continu
in)%0A%20%20%20%20%20%20radio%20button%20grid%0A%20%20%20%20%20%20ranking%20grid%0A%20%20%20%20%20%20semantic%20diff%0A%20%20%20%20%20%20slider%
-%20email%0A%20%20%20%20%20%20--
%20date%20(all%20three%20formats%20but%20not%20min%2Fmax%20date)%0A%20%20%20%20%20%20--
%20numeric%20with%20min%2Fmax%20number%20or%20min%2Fmax%20number%20of%20characters%0A%20%20%20%20%20%20--
%20currency%0A%20%20%20%20%20%20--%20percent%0A%20%20%20%20%20%20--
%20phone%20(based%20on%20the%20regex%20option)%0A%20%20%20%20%20%20textbox%20grid%20(same%20as%20above)%0A%20%20%20%20%20%20textbox%20list%20(same%2
random-whole-numbers-in-javascript-in-a-specific-
range%0A%20*%2F%0Afunction%20getRandomInt(min%2C%20max)%20%7B%0A%20%20%20%20min%20%3D%20Math.ceil(min)%3B%0A%20%20%20%20max%20%3D%20Math.floor(max)
%20min%20%2B%201))%20%2B%20min%3B%0A%7D%0A%0A%2F***%0A%20*%20Shuffle%20array%20in%20place%0A%20*%20Knuth%20shuffle%3A%20https%3A%2F%2Fstackoverflow
to-randomize-shuffle-a-javascript-
array%0A%20*%0A%20*%20array%20(array)%20will%20be%20mutated%0A%20*%20return%20(array)%20the%20original%20array%20after%20being%20shuffled%0A%20*%2F%0Afunction%
%3D%201%3B%0A%0A%20%20%20%20%2F%2F%20And%20swap%20it%20with%20the%20current%20element.%0A%20%20%20%20temporaryValue%20%3D%20array%5BcurrentIndex%5
5901811-28-305-box%0A%20*%20%20%20%20%20sgE-5901811-28-305-10997-
element%0A%20*%20return%20(obj)%20Returns%20object%20of%20the%20constituent%20parts%0A%20*%2F%0Aconst%20parseSgId%20%3D%20(id)%20%3D%3E%20%7B%0A%20%20co
(%5Cd%2B)-(%5Cd%2B)-(%5Cd%2B)(-(%5Cd%2B))%3F-
(%5Cw%2B)%2F%0A%0A%20%20const%20aParsed%20%3D%20id.match(regexID)%0A%0A%20%20if%20(!aParsed%20%7C%7C%20aParsed.length%20!%3D%3D%207)%20%7B%0A%20%2
the-
page%27)%0A%20%20if%20(iFrameElem)%0A%20%20%20%20return%20iFrameElem.contentWindow.SGAPI%0A%20%20return%20sgapi%20%3D%20SGAPI%0A%7D%0A%0A%2F***%0A%2
the-
page%27)%0A%20%20if%20(iFrameElem)%0A%20%20%20%20return%20iFrameElem.contentDocument%20%7C%7C%20iFrameElem.contentWindow.document%0A%20%20return%20documen
filled%2C%20false%20if%20there%20was%20already%20a%20value%0A%20*%2F%0Aconst%20dropdowns%20%3D%20(questionElem)%20%3D%3E%20%7B%0A%20%20let%20autoFilled%
%201)%20%2F%2F%201%20to%20skip%20past%20%22--%20Please%20Select%20--%22%2C%20and%20-1%20since%20this%20is%200-
based%0A%20%20%20%20%7D%0A%20%20%7D)%0A%20%20return%20autoFilled%0A%7D%0A%0A%2F***%0A%20*%20autofill%20star%20rating%20grid%0A%20*%0A%20*%20questic
filled%2C%20false%20if%20there%20was%20already%20a%20value%0A%20*%2F%0Aconst%20starRatingGrid%20%3D%20(questionElem)%20%3D%3E%20%7B%0A%20%20let%20autoFille
%201)%20%2F%2F%201%20to%20skip%20the%20initial%20X%2C%20-1%20since%20it%27s%20zero-
based%0A%20%20%20%20%20%20if%20(LOG)%20console.log(%22--
%20selecting%20stars%20%3D%20%22%2C%20random)%0A%20%20%20%20%20%20for%20(let%20i%20%3D%201%3B%20i%20%3C%3D%20random%3B%20i%2B%2B)%0A%20%20%2
star-
on%27)%0A%20%20%20%20%20%20labelElems%5Brandom%5D.querySelector(%27input%27).checked%20%3D%20true%0A%20%20%20%20%7D%0A%20%20%7D)%0A%20%20return%20
filled%2C%20false%20if%20there%20was%20already%20a%20value%0A%20*%2F%0Aconst%20continuousSum%20%3D%20(questionElem)%20%3D%3E%20%7B%0A%0A%20%20const%20
%201%3B%20i%2B%2B)%20%7B%0A%20%20%20%20%20%20inputElems%5Bi%5D.value%20%3D%20val%0A%20%20%20%20%7D%0A%20%20%20%20inputElems%5BinputElems.length
%201%5D.value%20%3D%20maxTotal%20-%20(val%20*%20(inputElems.length%20-
%201))%0A%20%20%7D%0A%20%20else%20%7B%0A%20%20%20%20inputElems.forEach(inputElem%20%3D%3E%20inputElem.value%20%3D%20getRandomInt(0%2C10))%0A%20%20%7
filled%2C%20false%20if%20there%20was%20already%20a%20value%0A%20*%2F%0Aconst%20slider%20%3D%20(questionElem)%20%3D%3E%20%7B%0A%0A%20%20%2F***%0A%20%
setup%27).value)%0A%20%20%20%20if%20(LOG)%20console.log(%22setupObj%20%3D%20%22%2C%20setupObj)%0A%20%20%20%20const%20steps%20%3D%20Math.floor((setupObj.max
%20setupObj.min)%20%2F%20setupObj.stepval)%0A%20%20%20%20const%20randomValue%20%3D%20setupObj.min%20%2B%20getRandomInt(0%2C%20steps)%20*%20setupObj.stepval%
%20setupObj.min)%20%2F%20(setupObj.max%20-
%20setupObj.min))%20*%20100)%0A%20%20%20%20if%20(LOG)%20console.log(%22random%20val%20%2F%20percent%20%3D%20%22%2C%20randomValue%2C%20%27%20%2F%20%
input%27).value)%0A%20%20%20%20return%20false%0A%0A%20%20const%20%7B%20randomValue%2C%0A%20%20%20%20%20%20%20%20%20%20randomPercent%20%7D%20%3D
input%27).value%20%3D%20randomValue%0A%0A%20%20%2F%2F%20set%20the%20slider%20handle%2C%20this%20must%20be%20on%20a%20timer%20b%2Fc%20of%20how%20the%
slider-
handle%27)%0A%20%20setTimeout(function%20()%20%7B%20sliderHandleElem.style.left%20%3D%20%60%24%7BrandomPercent%7D%25%60%20%7D%2C%20400)%0A%20%20setTimeou
filled%2C%20false%20if%20there%20was%20already%20a%20value%0A%20*%2F%0Aconst%20sliderList%20%3D%20(questionElem)%20%3D%3E%20%7B%0A%20%20let%20autoFilled%2
slider-
row%27).forEach(sliderRowElem%20%3D%3E%0A%20%20%20%20autoFilled%20%3D%20slider(sliderRowElem)%20%7C%7C%20autoFilled)%0A%20%20return%20autoFilled%0A%7D%0A%0
filled%2C%20false%20if%20there%20was%20already%20a%20value%0A%20*%2F%0Aconst%20textboxes%20%3D%20(questionElem)%20%3D%3E%20%7B%0A%0A%20%20const%20qid%
%20min%2Fmax%20chars%20%3D%20%22%2C%20minCharacters%2C%20%27%20%2F%20%27%2C%20maxCharacters)%0A%20%20%20%20return%20%7B%20minCharacters%2C%20max
%20min%2Fmax%20number%20%3D%20%22%2C%20minNumber%2C%20%27%20%2F%20%27%2C%20maxNumber)%0A%20%20%20%20return%20%7B%20minNumber%2C%20maxNumb
validation-
email%27))%0A%20%20%20%20%20%20%20%20inputElem.value%20%3D%20%27test%40test.com%27%0A%0A%20%20%20%20%20%20%20%2F%2F%20DATE%0A%20%20%20%20%20%2
validation-date%27))%20%7B%0A%20%20%20%20%20%20%20%20if%20(classList.contains(%27sg-validation-date-
yyyy%27))%20%0A%20%20%20%20%20%20%20%20%20%20inputElem.value%20%3D%20%272025%2F01%2F01%27%0A%20%20%20%20%20%20%20%20else%0A%20%20%20%20%2
validation-numeric%27)%20%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%7C%7C%20classList.contains(%27sg-
validation-percent%27)%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%7C%7C%20classList.contains(%27sg-validation-
currency%27)%20)%20%7B%0A%20%20%20%20%20%20%20%20if%20(minNumber%20%26%26%20maxNumber)%0A%20%20%20%20%20%20%20%20%20%20inputElem.value%20%3D%
%5C%5Cs%5D))%3F%5C%5Cd%7B3%7D%5B-
%5C%5Cs%5D%5C%5Cd%7B4%7D%24%22)%20%7B%0A%20%20%20%20%20%20%20%20inputElem.value%20%3D%20%27123-456-
7890%27%0A%20%20%20%20%20%20%7D%0A%0A%20%20%20%20%20%20%2F%2F%20OTHERWISE%2C%20normal%20text%0A%20%20%20%20%20else%20%7B%0A%20%20%
filled%2C%20false%20if%20there%20was%20already%20a%20value%0A%20*%2F%0Aconst%20essay%20%3D%20(questionElem)%20%3D%3E%20%7B%0A%0A%20%20const%20textareaE
filled%2C%20false%20if%20there%20was%20already%20a%20value%0A%20*%2F%0Aconst%20radioButton%20%3D%20(questionElem)%20%3D%3E%20%7B%0A%20%20if%20(LOG)%20co
%201)%5D%0A%20%20if%20(LOG)%20console.log(%22clicking%20%22%2C%20radioElem)%0A%0A%20%20%2F%2F%20fire%20display%20logic%20on%20later%20questions%0A%20%20r
```

```
other-li%27))%0A%20%20%20%20radioElem.parentElement.querySelector(%27input%5Btype%3Dtext%5D%27).value%20%3D%20%27write-
in%27%0A%0A%20%20return%20true%0A%7D%0A%0A%2F***%0A%20*%20autofill%20radio%20button%20grid%0A%20*%0A%20*%20questionElem%20(element)%0A%20*%20return%20
filled%2C%20false%20if%20there%20was%20already%20a%20value%0A%20*%2F%0Aconst%20radioButtonGrid%20%3D%20(questionElem)%20%3D%3E%20%7B%0A%20%20let%20autoFil
filled%2C%20false%20if%20there%20was%20already%20a%20value%0A%20*%2F%0Aconst%20imageSelect%20%3D%20(questionElem)%20%3D%3E%20%7B%0A%0A%20%20if%20(questi
image-
selected%27))%0A%20%20%20%20return%20false%0A%0A%20%20const%20imageSelectElems%20%3D%20questionElem.querySelectorAll(%27.sg-
image-
box%20label%27)%0A%20%20const%20imageSelectElem%20%3D%20imageSelectElems%5BgetRandomInt(0%2C%20imageSelectElems.length%20-
%201)%5D%0A%20%20if%20(LOG)%20console.log(%22clicking%20%22%2C%20imageSelectElem)%0A%0A%20%20%2F%2F%20fire%20display%20logic%20on%20later%20questions%0A%2
-%20all%20pages%0A%20*%0A%20*%20questionElem%20(element)%0A%20*%20return%20(t%2Ff)%20true%20if%20auto-
filled%2C%20false%20if%20there%20was%20already%20a%20value%0A%20*%2F%0Aconst%20conjoint%20%3D%20(questionElem)%20%3D%3E%20%7B%0A%0A%20%20if%20(questionEl
conjoint-
set%27)%0A%20%20for%20(let%20i%20%3D%200%3B%20i%20%3C%20conjointSetElems.length%3B%20i%2B%2B)%20%7B%0A%20%20%20%20if%20(LOG)%20console.log(%22%5Cnconj
%201)%0A%20%20%20%20%20%20document.querySelector(%27.sg-next-button.btn-
conjoint%27).click()%0A%20%20%7D%0A%20%20return%20true%0A%7D%0A%0A%2F***%0A%20*%20autofill%20max%20diff%20--
%20all%20pages%0A%20*%0A%20*%20questionElem%20(element)%0A%20*%20return%20(t%2Ff)%20true%20if%20auto-
filled%2C%20false%20if%20there%20was%20already%20a%20value%0A%20*%2F%0Aconst%20maxDiff%20%3D%20(questionElem)%20%3D%3E%20%7B%0A%0A%20%20if%20(questionEl
maxdiff-
set%27)%0A%20%20for%20(let%20i%20%3D%200%3B%20i%20%3C%20maxDiffSetElems.length%3B%20i%2B%2B)%20%7B%0A%20%20%20%20if%20(LOG)%20console.log(%22%5CmaxD
%201)%0A%20%20%20%20%20%20document.querySelector(%27.sg-next-
button%27).click()%0A%20%20%7D%0A%20%20return%20true%0A%7D%0A%0A%2F***%0A%20*%20autofill%20ranking%20grid%0A%20*%0A%20*%20questionElem%20(element)%0A%20
filled%2C%20false%20if%20there%20was%20already%20a%20value%0A%20*%2F%0Aconst%20rankingGrid%20%3D%20(questionElem)%20%3D%3E%20%7B%0A%0A%20%20if%20(questio
1%5D%0A%20%20let%20aRanking%20%3D%20%5B%5D%0A%20%20for%20(let%20i%20%3D%200%3B%20i%20%3C%20trElems.length%3B%20i%2B%2B)%0A%20%20%20%20aRanking
filled%2C%20false%20if%20there%20was%20already%20a%20value%0A%20*%2F%0Aconst%20checkbox%20%3D%20(questionElem%2C%20isCheckboxGridRow%20%3D%20false)%20%3D%
12%27%2C%20where%2012%20is%20the%20QID%0A%20%20%20%20%2F%2Fconst%20row_qid%20%3D%20%5B...questionElem.classList%5D.find(s%20%3D%3E%20s.slice(0%2C%204)%2
%27)%0A%20%20%20%20const%20row_qid%20%3D%20%5B...questionElem.classList%5D.find(s%20%3D%3E%20s.startsWith(%27row-
%27))%0A%20%20%20%20const%20qid%20%3D%20parseInt(row_qid.slice(4))%0A%20%20%20%20minChecks%20%3D%20getProperty(qid%2C%20%27min_answers_per_row%27%2C%20tru
%201)%0A%20%20%20%20if%20(!checkElems%5Brandom%5D.checked)%20%7B%0A%20%20%20%20%20%20const%20checkElem%20%3D%20checkElems%5Brandom%5D%0A%20%20%2
other-
li%27))%0A%20%20%20%20%20%20%20%20checkElem.parentElement.querySelector(%27input%5Btype%3Dtext%5D%27).value%20%3D%20%27write-
in%27%0A%20%20%20%20%7D%0A%20%20%7D%0A%20%20return%20true%0A%7D%0A%0A%2F***%0A%20*%20autofill%20checkbox%20grid%0A%20*%0A%20*%20questionElem%20
filled%2C%20false%20if%20there%20was%20already%20a%20value%0A%20*%2F%0Aconst%20checkboxGrid%20%3D%20(questionElem)%20%3D%3E%20%7B%0A%20%20let%20autoFillec
--------------------
%5CnquestionElem%20%3D%20%22%2C%20questionElem)%0A%0A%20%20%20%20let%20autoFilled%20%3D%20false%0A%0A%20%20%20%20if%20(!questionElem.classList.contains(%27
hide%27))%20%7B%0A%20%20%20%20%20%20if%20(LOG)%20console.log(%22--
%20autopopulating%3A%20%22%2C%20questionElem.id)%0A%0A%20%20%20%20%20%20%2F%2F%20CHECKBOX%0A%20%20%20%20%20%20if%20(questionElem.classList.contains(%27
type-
checkbox%27))%0A%20%20%20%20%20%20%20%20autoFilled%20%3D%20checkbox(questionElem)%0A%20%20%20%20%20%20%2F%2F%20CHECKBOX%20GRID%0A%20%20%20%20%20%
type-table-
checkbox%27))%0A%20%20%20%20%20%20%20%20autoFilled%20%3D%20checkboxGrid(questionElem)%0A%0A%20%20%20%20%20%20%2F%2F%20CONJOINT%0A%20%20%20%20%2
type-
conjoint_new%27))%0A%20%20%20%20%20%20%20%20autoFilled%20%3D%20conjoint(questionElem)%0A%0A%20%20%20%20%20%20%2F%2F%20CONTINUOUS%20SUM%0A%20%20%
type-continuous-
sum%27))%0A%20%20%20%20%20%20%20%20autoFilled%20%3D%20continuousSum(questionElem)%0A%0A%20%20%20%20%20%20%2F%2F%20DROPDOWN%0A%20%20%20%20%2
type-menu%27)%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%7C%7C%20questionElem.classList.contains(%27sg-type-
multimenu%27)%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%7C%7C%20questionElem.classList.contains(%27sg-type-
table-menu-
matrix%27))%0A%20%20%20%20%20%20%20%20autoFilled%20%3D%20dropdowns(questionElem)%0A%0A%20%20%20%20%20%20%2F%2F%20ESSAY%0A%20%20%20%20%20%20els
type-
essay%27))%0A%20%20%20%20%20%20%20%20autoFilled%20%3D%20essay(questionElem)%0A%0A%20%20%20%20%20%20%2F%2F%20IMAGE%20SELECT%0A%20%20%20%20%20%
type-
imageselect%27))%0A%20%20%20%20%20%20%20%20autoFilled%20%3D%20imageSelect(questionElem)%0A%0A%20%20%20%20%20%20%2F%2F%20MAX%20DIFF%0A%20%20%20%2
type-
maxdiff%27))%0A%20%20%20%20%20%20%20%20autoFilled%20%3D%20maxDiff(questionElem)%0A%0A%20%20%20%20%20%20%2F%2F%20RADIO%20BUTTON%0A%20%20%20%20%
type-
radio%27))%0A%20%20%20%20%20%20%20%20autoFilled%20%3D%20radioButton(questionElem)%0A%20%20%20%20%20%20%2F%2F%20RADIO%20BUTTON%20GRID%0A%20%20%2
type-table-
radio%27))%0A%20%20%20%20%20%20%20%20autoFilled%20%3D%20radioButtonGrid(questionElem)%0A%0A%20%20%20%20%20%20%2F%2F%20RANKING%20GRID%0A%20%20%20
type-rank-
table%27))%0A%20%20%20%20%20%20%20%20autoFilled%20%3D%20rankingGrid(questionElem)%0A%0A%20%20%20%20%20%20%2F%2F%20SEMANTIC%20DIFF%0A%20%20%20%20
type-table-
semantic%27))%0A%20%20%20%20%20%20%20%20autoFilled%20%3D%20radioButtonGrid(questionElem)%0A%0A%20%20%20%20%20%20%2F%2F%20SLIDER%0A%20%20%20%20%20
type-
slider%27))%0A%20%20%20%20%20%20%20%20autoFilled%20%3D%20slider(questionElem)%0A%20%20%20%20%20%20%2F%2F%20SLIDER%20LIST%0A%20%20%20%20%20%20else
type-multi-
slider%27))%0A%20%20%20%20%20%20%20%20autoFilled%20%3D%20sliderList(questionElem)%0A%0A%20%20%20%20%20%20%2F%2F%20STAR%20RATING%20GRID%0A%20%20%2
type-table-
stars%27))%0A%20%20%20%20%20%20%20%20autoFilled%20%3D%20starRatingGrid(questionElem)%0A%0A%20%20%20%20%20%20%2F%2F%20TEXTBOX%20%2F%20TEXTBOX%20LI
type-textbox%27)%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%7C%7C%20questionElem.classList.contains(%27sg-
type-multitext%27)%0A%20%20%20%20%20%20%20%20%20%20%20%20%20%20%7C%7C%20questionElem.classList.contains(%27sg-
type-table-
textbox%27))%0A%20%20%20%20%20%20%20%20autoFilled%20%3D%20textboxes(questionElem)%0A%0A%20%20%20%20%20%20%2F%2F%20UNKOWN%2C%20IGNORE%0A%20%20
-
%20question%20type%20not%20recognized%2C%20ignored%22)%0A%20%20%20%20%20%20%7D%0A%20%20%20%20%20%20if%20(LOG%20%26%26%20!autoFilled)%20console.log(%2
-
%20Already%20set%22)%0A%20%20%20%20%20%20autoFilledAnyQuestion%20%3D%20autoFilled%20%7C%7C%20autoFilledAnyQuestion%0A%20%20%20%20%20%20questionElem.scroll
-
%20question%20hidden%2C%20igore%20it%22)%0A%20%20%20%20%7D%0A%20%20%7D)%0A%20%20if%20(autoFilledAnyQuestion)%20%7B%0A%20%20%20%20if%20(LOG)%20consol
--------------------
%5Cn%3E%3E%3E%3E%20Looping%20to%20see%20if%20display%20logic%20popped%20up%20anything%20else%22)%0A%20%20%20%20setTimeout(function()%20%7B%0A%20%20%20
question-
set%27).children%5D%0A%0A%20%20const%20questionElems%20%3D%20%5B%5D%0A%0A%20%20a.forEach(elem%20%3D%3E%20%7B%0A%0A%20%20%20%20%2F%2F%20if%20ele
type-group%27))%20%7B%0A%20%20%20%20%20%20%2F%2F%20the%20questions%20in%20a%20Custom%20Group%20are%20.sg-
queston%27s%20under%20.sg-group-
item%20elements%0A%20%20%20%20%20%20const%20groupItemElems%20%3D%20elem.querySelectorAll(%27.sg-group-
item%27)%0A%20%20%20%20%20%20groupItemElems.forEach(groupItemElem%20%3D%3E%0A%20%20%20%20%20%20%20%20questionElems.push(groupItemElem.querySelector(%27.sg-
question%27)))%0A%20%20%20%20%7D%0A%20%20%20%20%2F%2F%20else%20elem%20is%20a%20question%20itself%0A%20%20%20%20else%20%7B%0A%20%20%20%20%20%20%2
--------------------
%5CnquestionElems%20%3D%20%22%2C%20questionElems%2C%20%27%5Cn%5E%5E%5E%5E%5E%5E%5E%5E%5E%5E%5E%5E%5E%5E%5E%5E%5E%5E%5E%5E%5E%5E%5E%5E%5Cn%27)%
--------------------
%5CnquestionElems%20%3D%20%22%2C%20questionElems%2C%20%27%5Cn%5E%5E%5E%5E%5E%5E%5E%5E%5E%5E%5E%5E%5E%5E%5E%5E%5E%5E%5E%5E%5E%5E%5E%5E%5Cn%27)%
()%0A%0A
```

Unadulterated code and process below is for reference only, if you completed the steps below you have added the Autofill bookmarklet.

To make this code work as a Bookmarklet:

1. Set **BOOKMARK = true**
2. Copy only this part of the code (not the <script> tags): **(function() { . . . })**
3. Run it through a URL Encoder to get the code shown above

To test the code in a single survey:

1. Copy to Style tab > HTML/CSS Editor (lower right of page) > Custom Header

```
<script>

/* Alchemer v03

  Autofill Chrome Bookmarklet

  Works with question types:
    checkbox (with optional min number of selections or optional write-in)
    checkbox grid (with optional min number of selections)
    conjoint
    continuous sum (with optional required total)
    essay
    dropdown
    dropdown menu grid
    dropdown menu list
    image multi select (but doesn't look at min number...yet)
    image select
    likert
    max diff
    nps
    radio button (with optional write-in)
    radio button grid
    ranking grid
    semantic diff
    slider (optional min/max and step)
    slider list (same as above)
    star rating grid
    textbox
    -- email
    -- date (all three formats but not min/max date)
    -- numeric with min/max number or min/max number of characters
    -- currency
    -- percent
    -- phone (based on the regex option)
    textbox grid (same as above)
    textbox list (same as above)

    Custom Groups containing any of the above

  Not working for
    actions
    audo sentiment
    cascading dropdown
    custom table
    drag and drop ranking
    file upload
    grouping open / closed card sort
    image heatmap
    text highlighter
    quick sort
    signature
    video feedback
    video sentiment
*/

(function() {
const LOG = true
const BOOKMARKLET = false // true when setting up as bookmarlet, false to run in the survey Style > HEADER

/**
 * Returns a random integer between min (inclusive) and max (inclusive).
 * https://stackoverflow.com/questions/1527803/generating-random-whole-numbers-in-javascript-in-a-specific-r
ange
 */
function getRandomInt(min, max) {
    min = Math.ceil(min);
    max = Math.floor(max);
    return Math.floor(Math.random() * (max - min + 1)) + min;
}

/***
 * Shuffle array in place
 * Knuth shuffle: https://stackoverflow.com/questions/2450954/how-to-randomize-shuffle-a-javascript-array
 *
 * array (array) will be mutated
 * return (array) the original array after being shuffled
 */
function shuffle(array) {
  var currentIndex = array.length, temporaryValue, randomIndex;

  // While there remain elements to shuffle...
  while (0 !== currentIndex) {

    // Pick a remaining element...
    randomIndex = Math.floor(Math.random() * currentIndex);
    currentIndex -= 1;

    // And swap it with the current element.
    temporaryValue = array[currentIndex];
    array[currentIndex] = array[randomIndex];
    array[randomIndex] = temporaryValue;
  }

  return array;
}

/**
 * Parse an Alchemer element #ID in the form:
 *     sgE-5901811-28-305-box
 *     sgE-5901811-28-305-10997-element
 * 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'
  }
}

/***
 * Get the survey's SGAPI variable
 *
 * return (obj) the SGAPI global vraible
 */
const getSGAPI = () => {
  // Preview is in an iFrame
  const iFrameElem = document.querySelector('iframe#preview-the-page')
  if (iFrameElem)
    return iFrameElem.contentWindow.SGAPI
  return sgapi = SGAPI
}

/***
 * Get the survey's document element
 *
 * return (elem)
 */
const getDocument = () => {
  // Preview is in an iFrame
  const iFrameElem = document.querySelector('iframe#preview-the-page')
  if (iFrameElem)
    return iFrameElem.contentDocument || iFrameElem.contentWindow.document
  return document
}

/***
 * Get a property from SGAPI for the qid
 *
 * qid (int) question ID
 * propertyName (string) name of property
 * isInt (t/f) convert return value to int if true
 * return (int/string) the property or 0 / '' if property doesn't exist
 */
const getProperty = (qid, propertyName, isInt) => {
  const val = getSGAPI().survey.surveyObject.questions[qid].properties[propertyName]
  if (isInt)
    return parseInt(val) || 0
  return val || ''
 }

/***
 * autofill dropdowns
 *
 * questionElem (element)
 * return (t/f) true if auto-filled, false if there was already a value
 */
const dropdowns = (questionElem) => {
  let autoFilled = false
  const selectElems = questionElem.querySelectorAll('select')
  selectElems.forEach(selectElem => {
    if (LOG) console.log("selectElem.value = ", selectElem.value)
    // Don't change if dropdown already has a value
    //  'NoAnswer' for dropdown and dropdown menu list, '' for dropdown menu grid
    if (selectElem.value === 'NoAnswer' || selectElem.value === '') {
      autoFilled = true
      const numOptions = selectElem.querySelectorAll('option').length
      selectElem.selectedIndex = getRandomInt(1, numOptions - 1) // 1 to skip past "-- Please Select --", and -1 sin
ce this is 0-based
    }
  })
  return autoFilled
}

/***
 * autofill star rating grid
 *
 * questionElem (element)
 * return (t/f) true if auto-filled, false if there was already a value
 */
const starRatingGrid = (questionElem) => {
  let autoFilled = false
  questionElem.querySelectorAll('tbody td').forEach(tdElem => {
    if (!tdElem.querySelector('input:checked')) {
      autoFilled = true
      const labelElems = tdElem.querySelectorAll('label')
      const random = getRandomInt(1, labelElems.length - 1) // 1 to skip the initial X, -1 since it's zero-based
      if (LOG) console.log("-- selecting stars = ", random)
      for (let i = 1; i <= random; i++)
        labelElems[i].classList.add('sg-star-on')
      labelElems[random].querySelector('input').checked = true
    }
  })
  return autoFilled
```

```
}

/***
 * autofill continuous sum
 *
 * questionElem (element)
 * return (t/f) true if auto-filled, false if there was already a value
 */
const continuousSum = (questionElem) => {

  const inputElems = questionElem.querySelectorAll('tbody input[type=text]')

  for (let i = 0; i < inputElems.length; i++) {
    if (inputElems[i].value !== '')
      return false
  }

  const qid = parseSgId(questionElem.id).qid
  // const maxTotal = parseInt(SGAPI.survey.surveyObject.questions[qid].properties.max_total) || 0
  const maxTotal = getProperty(qid, 'max_total', true)
  if (maxTotal) {
    const val = Math.floor(maxTotal / inputElems.length)
    for (let i = 0; i < inputElems.length - 1; i++) {
      inputElems[i].value = val
    }
    inputElems[inputElems.length - 1].value = maxTotal - (val * (inputElems.length - 1))
  }
  else {
    inputElems.forEach(inputElem => inputElem.value = getRandomInt(0,10))
  }

  // update the total
  inputElems[0].focus()
  inputElems[0].blur()

  return true
}

/***
 * autofill slider
 *
 * questionElem (element) can be a question or a sliderRowElem for a slider list
 * return (t/f) true if auto-filled, false if there was already a value
 */
const slider = (questionElem) => {

  /***
   * Get a random value and percent
   *
   * return (obj of int) { randomValue, randomPercent }
   */
  const getRandomValueAndPercent = () => {

    const setupObj = JSON.parse(questionElem.querySelector('.slider-setup').value)
    if (LOG) console.log("setupObj = ", setupObj)
    const steps = Math.floor((setupObj.max - setupObj.min) / setupObj.stepval)
    const randomValue = setupObj.min + getRandomInt(0, steps) * setupObj.stepval
    const randomPercent = Math.floor(((randomValue - setupObj.min) / (setupObj.max - setupObj.min)) * 100)
    if (LOG) console.log("random val / percent = ", randomValue, ' / ', randomPercent, '%')

    return { randomValue, randomPercent }
  }

  /***
   * main()
   */

  // already set, don't change
  if (questionElem.querySelector('input.sg-input').value)
    return false

  const { randomValue,
          randomPercent } = getRandomValueAndPercent()

  // set slider value
  questionElem.querySelector('input.sg-input').value = randomValue

  // set the slider handle, this must be on a timer b/c of how the slider functions
  const sliderHandleElem = questionElem.querySelector('.ui-slider-handle')
  setTimeout(function () { sliderHandleElem.style.left = `${randomPercent}%` }, 400)
  setTimeout(function () { sliderHandleElem.style.left = `${randomPercent}%` }, 1000) // ensure it worked!
  return true
}

/***
 * autofill sliderList
 *
 * questionElem (element)
 * return (t/f) true if auto-filled, false if there was already a value
 */
const sliderList = (questionElem) => {
  let autoFilled = false
  questionElem.querySelectorAll('.sg-slider-row').forEach(sliderRowElem =>
    autoFilled = slider(sliderRowElem) || autoFilled)
  return autoFilled
}

/***
 * autofill textboxes
 *
 * questionElem (element)
 * return (t/f) true if auto-filled, false if there was already a value
```

```
 */
const textboxes = (questionElem) => {

  const qid = parseSgId(questionElem.id).qid

  /***
   * Get the validation parameters for min/max charachter count. or 0 if not set
   *
   * return (obj of ints) { minCharacters, maxCharacters }
   */
  const getMinMaxCharacters = () => {
    //const qid = parseSgId(questionElem.id).qid
    const minCharacters = getProperty(qid, 'min_characters', true)
    const maxCharacters = getProperty(qid, 'max_characters', true)

    if (LOG) console.log("textboxes(), qid = ", qid)
    if (LOG) console.log("- min/max chars = ", minCharacters, ' / ', maxCharacters)
    return { minCharacters, maxCharacters }
  }

  /***
   * Get the validation parameters for min/max number, or 0 if not set
   *
   * return (obj of ints) { minNumber, maxNumber }
   */
  const getMinMaxNumber = () => {
    //const qid = parseSgId(questionElem.id).qid
    const minNumber = getProperty(qid, 'min_number', true)
    const maxNumber = getProperty(qid, 'max_number', true)

    if (LOG) console.log("textboxes(), qid = ", qid)
    if (LOG) console.log("- min/max number = ", minNumber, ' / ', maxNumber)
    return { minNumber, maxNumber }
  }

  /***
   * Get the input mask (regex validation)
   *
   * reutrn (string) the question property inputmask.MASK or empty string
   */
  const getInputMask = () => {
    const inputMask = getProperty(qid, 'inputmask', false)
    const retval = (inputMask) ? inputMask.MASK : ''
    if (LOG) console.log("getInputMask() = ", retval)
    return retval
  }

  /***
   * main()
   */

  let autoFilled = false

  const { minCharacters,
          maxCharacters } = getMinMaxCharacters()

  let { minNumber,
        maxNumber } = getMinMaxNumber()
  minNumber = Math.ceil(minNumber)
  maxNumber = Math.floor(maxNumber)

  const inputElems = questionElem.querySelectorAll('input[type=text]')
  inputElems.forEach(inputElem => {

    // only fill if there's no value
    if (!inputElem.value) {

      autoFilled = true

      const classList = inputElem.classList

      // EMAIL
      if (classList.contains('sg-validation-email'))
        inputElem.value = 'test@test.com'

      // DATE
      else if (classList.contains('sg-validation-date')) {
        if (classList.contains('sg-validation-date-yyyy'))
          inputElem.value = '2025/01/01'
        else
          inputElem.value = '01/01/2025'
      }

      // NUMERIC
      else if (   classList.contains('sg-validation-numeric')
               || classList.contains('sg-validation-percent')
               || classList.contains('sg-validation-currency') ) {
        if (minNumber && maxNumber)
          inputElem.value = getRandomInt(minNumber, maxNumber)
        else if (minNumber)
          inputElem.value = getRandomInt(minNumber, minNumber + 20)
        else if (maxNumber)
          inputElem.value = getRandomInt(0, maxNumber)
        else if (minCharacters)
          inputElem.value = '1'.repeat(minCharacters)
        else
          inputElem.value = '123'.slice(0, maxCharacters | 3)
      }

      // US PHONE (from the Alchemer regex for a US Phone, note: the backslashes are escaped so they appear dou
bled)
      else if (getInputMask() === "^((\\(\\d{3}\\) ?)|(\\d{3}[-\\s]))?\\d{3}[-\\s]\\d{4}$") {
```

```
        inputElem.value = '123-456-7890'
      }

      // OTHERWISE, normal text
      else {
        if (LOG) console.log("otherwise, normal text")
        if (minCharacters) {
          if (minCharacters === 5) // special case for zip
            inputElem.value = '12345'
          else
            inputElem.value = 'x'.repeat(minCharacters)
        }
        else  {
          inputElem.value = 'test'.slice(0, maxCharacters | 4)
        }
      }

      // fire display logic on later questions
      inputElem.focus()
      inputElem.blur()
    }
  })
  return autoFilled
}

/***
 * autofill essay
 *
 * questionElem (element)
 * return (t/f) true if auto-filled, false if there was already a value
 */
const essay = (questionElem) => {

  const textareaElem = questionElem.querySelector('textarea')

  // if already has a value, do nothing
  if (textareaElem.value)
    return false

  textareaElem.value = 'test'

  // fire display logic on later questions
  textareaElem.focus()
  textareaElem.blur()
  return true
}

/***
 * autofill radio button
 *
 * questionElem (element) question or other elem type for radio button grid or conjoint
 * return (t/f) true if auto-filled, false if there was already a value
 */
const radioButton = (questionElem) => {
  if (LOG) console.log("radioButton = ", questionElem)
  // if already selected, do nothing
  if (questionElem.querySelectorAll('input[type=radio]:checked').length)
    return false

  const radioElems = questionElem.querySelectorAll('input[type=radio]')

  const radioElem = radioElems[getRandomInt(0, radioElems.length - 1)]
  if (LOG) console.log("clicking ", radioElem)

  // fire display logic on later questions
  radioElem.click()

  // check for Other Write In
  if (radioElem.parentElement.classList.contains('sg-other-li'))
    radioElem.parentElement.querySelector('input[type=text]').value = 'write-in'

  return true
}

/***
 * autofill radio button grid
 *
 * questionElem (element)
 * return (t/f) true if auto-filled, false if there was already a value
 */
const radioButtonGrid = (questionElem) => {
  let autoFilled = false
  const trElems = questionElem.querySelectorAll('tbody tr')
  trElems.forEach(trElem =>
    autoFilled = radioButton(trElem) || autoFilled)
  return autoFilled
}

/***
 * autofill image select and image multi select
 *
 * questionElem (element)
 * return (t/f) true if auto-filled, false if there was already a value
 */
const imageSelect = (questionElem) => {

  if (questionElem.querySelector('.sg-image-selected'))
    return false

  const imageSelectElems = questionElem.querySelectorAll('.sg-image-box label')
  const imageSelectElem = imageSelectElems[getRandomInt(0, imageSelectElems.length - 1)]
  if (LOG) console.log("clicking ", imageSelectElem)
```

```
  // fire display logic on later questions
  imageSelectElem.click()

  return true
}

/***
 * autofill conjoint -- all pages
 *
 * questionElem (element)
 * return (t/f) true if auto-filled, false if there was already a value
 */
const conjoint = (questionElem) => {

  if (questionElem.querySelector('input[type=radio]:checked'))
    return false

  const conjointSetElems = questionElem.querySelectorAll('.sg-conjoint-set')
  for (let i = 0; i < conjointSetElems.length; i++) {
    if (LOG) console.log("\nconjoint set = ", conjointSetElems[i])
    radioButton(conjointSetElems[i])

    if (i !== conjointSetElems.length - 1)
      document.querySelector('.sg-next-button.btn-conjoint').click()
  }
  return true
}

/***
 * autofill max diff -- all pages
 *
 * questionElem (element)
 * return (t/f) true if auto-filled, false if there was already a value
 */
const maxDiff = (questionElem) => {

  if (questionElem.querySelectorAll('input[type=radio]:checked').length)
    return false

  const maxDiffSetElems = questionElem.querySelectorAll('.sg-maxdiff-set')
  for (let i = 0; i < maxDiffSetElems.length; i++) {
    if (LOG) console.log("\maxDiff set = ", maxDiffSetElems[i])

    const trElems = shuffle([...maxDiffSetElems[i].querySelectorAll('tbody tr')])
    trElems[0].querySelectorAll('input[type=radio]')[0].click()
    trElems[1].querySelectorAll('input[type=radio]')[1].click()

    if (i !== maxDiffSetElems.length - 1)
      document.querySelector('.sg-next-button').click()
  }
  return true
}

/***
 * autofill ranking grid
 *
 * questionElem (element)
 * return (t/f) true if auto-filled, false if there was already a value
 */
const rankingGrid = (questionElem) => {

  if (questionElem.querySelectorAll('tbody tr input[type=radio]:checked').length)
    return false

  const trElems = questionElem.querySelectorAll('tbody tr')

  // get a randomized array of ints [0..trElems.length-1]
  let aRanking = []
  for (let i = 0; i < trElems.length; i++)
    aRanking.push(i)
  aRanking = shuffle(aRanking)

  trElems.forEach((trElem, idx) => {
    const inputElems = trElem.querySelectorAll('input[type=radio]')
    inputElems[aRanking[idx]].click()
  })

  return true
}

/***
 * autofill checkbox
 *
 * checkboxElem (element) a question for a checkbox OR a TR for a checkbox grid row
 * return (t/f) true if auto-filled, false if there was already a value
 */
const checkbox = (questionElem, isCheckboxGridRow = false) => {

  console.log("checkbox() questionElem = ", questionElem)

  // if already checked, do nothing
  if (questionElem.querySelectorAll('input[type=checkbox]:checked').length)
    return false

  // checkboxes
  const checkElems = questionElem.querySelectorAll('input[type=checkbox]')

  // the minimum number of checks based on the Validation for the checkbox question or checkbox grid
  let minChecks = undefined
  if (isCheckboxGridRow) {
    // checkbox grid TRs have a class name in the form 'row-12', where 12 is the QID
    //const row_qid = [...questionElem.classList].find(s => s.slice(0, 4) === 'row-')
```

```
    const row_qid = [...questionElem.classList].find(s => s.startsWith('row-'))
    const qid = parseInt(row_qid.slice(4))
    minChecks = getProperty(qid, 'min_answers_per_row', true)
  }
  else {
    const qid = parseSgId(questionElem.id).qid
    minChecks = getProperty(qid, 'minimum_response', true)
  }
  minChecks = Math.min(minChecks, checkElems.length) || 1
  console.log("minChecks = ", minChecks)

  // check the min number of checkboxes and fire display logic on later questions
  let checked = 0
  while (checked < minChecks) {
    const random = getRandomInt(0, checkElems.length - 1)
    if (!checkElems[random].checked) {
      const checkElem = checkElems[random]
      checkElem.click()
      checked++

      // check for Other Write In
      if (checkElem.parentElement.classList.contains('sg-other-li'))
        checkElem.parentElement.querySelector('input[type=text]').value = 'write-in'
    }
  }
  return true
}

/***
 * autofill checkbox grid
 *
 * questionElem (element)
 * return (t/f) true if auto-filled, false if there was already a value
 */
const checkboxGrid = (questionElem) => {
  let autoFilled = false
  const trElems = questionElem.querySelectorAll('tbody tr')
  trElems.forEach(trElem =>
    autoFilled = checkbox(trElem, true) || autoFilled)
  return autoFilled
}

/***
 * autofill the page
 *
 * This function uses a Timeout to recurse.  The Timeout allows the survey's
 * display logic engine to run and we go through the questions again to
 * fill any new ones that were displayed.
 *
 * questionElems (arr of elems) all question on the page including hidden
 */
const autofill = (questionElems) => {

  let autoFilledAnyQuestion = false

  questionElems.forEach(questionElem => {

    if (LOG) console.log("\n----------------------\nquestionElem = ", questionElem)

    let autoFilled = false

    if (!questionElem.classList.contains('sg-hide')) {
      if (LOG) console.log("-- autopopulating: ", questionElem.id)

      // CHECKBOX
      if (questionElem.classList.contains('sg-type-checkbox'))
        autoFilled = checkbox(questionElem)
      // CHECKBOX GRID
      else if (questionElem.classList.contains('sg-type-table-checkbox'))
        autoFilled = checkboxGrid(questionElem)

      // CONJOINT
      else if (questionElem.classList.contains('sg-type-conjoint_new'))
        autoFilled = conjoint(questionElem)

      // CONTINUOUS SUM
      else if (questionElem.classList.contains('sg-type-continuous-sum'))
        autoFilled = continuousSum(questionElem)

      // DROPDOWN
      // DROPDOWN MENU LIST
      // DROPDOWN MENU GRID
      else if (   questionElem.classList.contains('sg-type-menu')
              || questionElem.classList.contains('sg-type-multimenu')
              || questionElem.classList.contains('sg-type-table-menu-matrix'))
        autoFilled = dropdowns(questionElem)

      // ESSAY
      else if (questionElem.classList.contains('sg-type-essay'))
        autoFilled = essay(questionElem)

      // IMAGE SELECT
      // IMAGE MULTI SELECT
      else if (questionElem.classList.contains('sg-type-imageselect'))
        autoFilled = imageSelect(questionElem)

      // MAX DIFF
      else if (questionElem.classList.contains('sg-type-maxdiff'))
        autoFilled = maxDiff(questionElem)

      // RADIO BUTTON
      else if (questionElem.classList.contains('sg-type-radio'))
```

```
      autoFilled = radioButton(questionElem)
    // RADIO BUTTON GRID
    else if (questionElem.classList.contains('sg-type-table-radio'))
      autoFilled = radioButtonGrid(questionElem)

    // RANKING GRID
    else if (questionElem.classList.contains('sg-type-rank-table'))
      autoFilled = rankingGrid(questionElem)

    // SEMANTIC DIFF
    else if (questionElem.classList.contains('sg-type-table-semantic'))
      autoFilled = radioButtonGrid(questionElem)

    // SLIDER
    else if (questionElem.classList.contains('sg-type-slider'))
      autoFilled = slider(questionElem)
    // SLIDER LIST
    else if (questionElem.classList.contains('sg-type-multi-slider'))
      autoFilled = sliderList(questionElem)

    // STAR RATING GRID
    else if (questionElem.classList.contains('sg-type-table-stars'))
      autoFilled = starRatingGrid(questionElem)

    // TEXTBOX / TEXTBOX LIST / TEXTBOX GRID
    else if (   questionElem.classList.contains('sg-type-textbox')
            || questionElem.classList.contains('sg-type-multitext')
            || questionElem.classList.contains('sg-type-table-textbox'))
      autoFilled = textboxes(questionElem)

    // UNKOWN, IGNORE
    else {
      console.log("-- question type not recognized, ignored")
    }
    if (LOG && !autoFilled) console.log("-- Already set")
    autoFilledAnyQuestion = autoFilled || autoFilledAnyQuestion
    questionElem.scrollIntoView()
   }
   else {
    console.log("-- question hidden, igore it")
   }
  })
  if (autoFilledAnyQuestion) {
   if (LOG) console.log("\n---------------------\n>>>> Looping to see if display logic popped up anything else")
   setTimeout(function() {
    autofill(questionElems)
   }, 500) // wait to allow the survey engine's display logic to fire
  }
  else {
   if (LOG) console.log("\nDONE!")
  }
}

/***
 * Get array of questions on the page (pulling up questions in a Custom Group)
 */
const getQuestionElems = () => {

  let docElem = getDocument()

  if (LOG) console.log("docElem = ", docElem)

  const a = [...docElem.querySelector('.sg-question-set').children]

  const questionElems = []

  a.forEach(elem => {

    // if elem is a Custom Group, look for the question elements in it
    if (elem.classList.contains('sg-type-group')) {
     // the questions in a Custom Group are .sg-queston's under .sg-group-item elements
     const groupItemElems = elem.querySelectorAll('.sg-group-item')
     groupItemElems.forEach(groupItemElem =>
      questionElems.push(groupItemElem.querySelector('.sg-question')))
    }
    // else elem is a question itself
    else {
     questionElems.push(elem)
    }
   })

   return questionElems
 }

/***
 * main()
 */

if (BOOKMARKLET) {
  const questionElems = getQuestionElems()
  console.log("\n---------------------\nquestionElems = ", questionElems, '\n^^^^^^^^^^^^^^^^^^^^^^^^\n')
  autofill(questionElems)
}
else {
  document.addEventListener("DOMContentLoaded", function() {
    const questionElems = getQuestionElems()
    console.log("\n---------------------\nquestionElems = ", questionElems, '\n^^^^^^^^^^^^^^^^^^^^^^^^\n')
    autofill(questionElems)
  })
}
})()
```

```
</script>
```


Related Articles