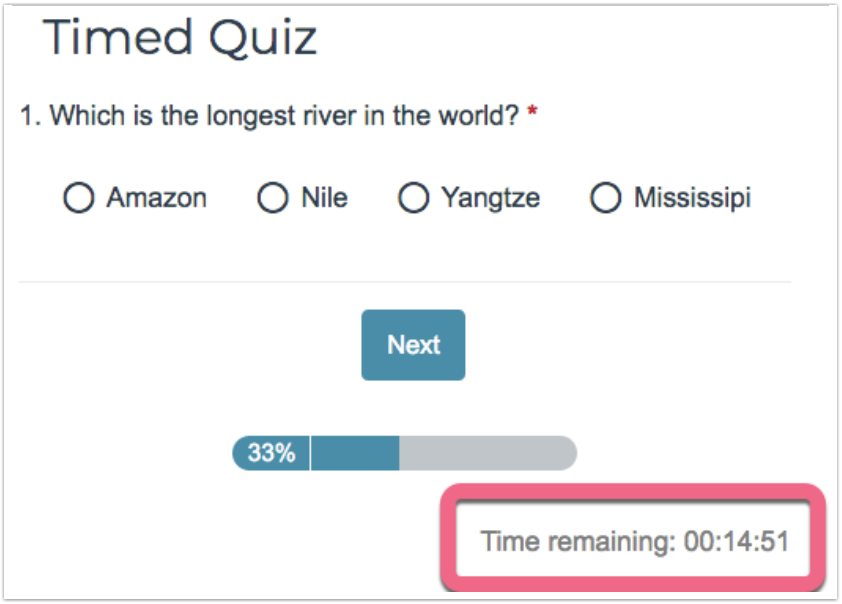
Timed Quiz or Test

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

Present a survey with a timed section that must be completed within a specific amount of hours/minutes after the respondent starts the timed section.



Effort: ✔ ✔ ✔

Solution

This is a more flexible alternative to the built-in Page Timer feature. It also manages a quirk of the Page Timer that can allow a respondent untimed completion of their survey in the case that the respondent starts the survey from an Email Invitation, closes the tab, waits for the Page Timer to expire, and clicks their Invite Link again.

This solution has the following survey structure:

- Introductory untimed pages
- Start Timed Section page (hidden)
- **Timed Pages**
- End Timed Section (hidden)
- Closing untimed pages

Note: If one previews a Timed Page, respondents receive an error that the initial start time was not set. **This is normal and can be ignored**. You can temporarily remove the script to prevent the

error from appearing during testing.

Step 1: Add Introductory Pages

Add as many pages of untimed questions and information as needed.

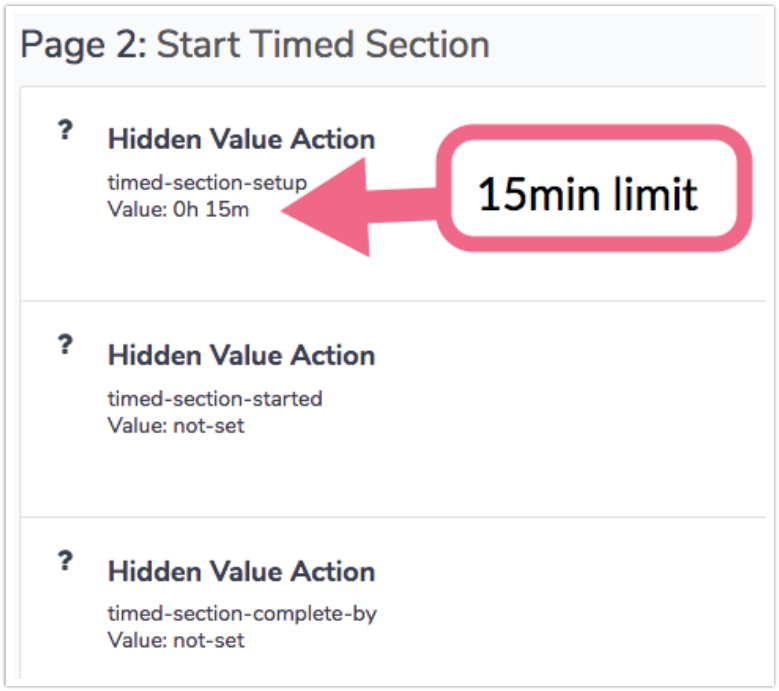
Optional: To change the **Next** button to read **Start Exam** on the last introductory page add the Javascript Action:

```
document.addEventListener("DOMContentLoaded", function() {
  document.querySelector('#sg_NextButton').value = "Start Exam"
})
```

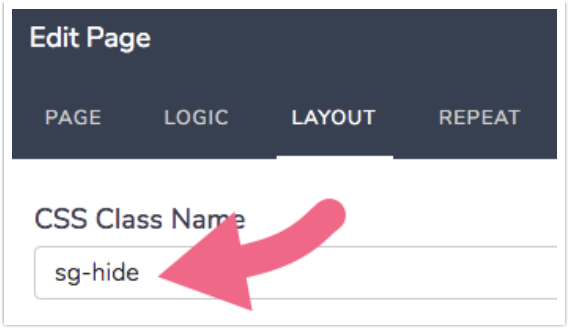
Step 2: Add Hidden Start Timed Section page

1. Add a new page with three Hidden Value Actions with specific titles:

   - **timed-section-setup** -- This defines the hours and minutes allowed for the Timed Section. Enter this in the "**Populate with the following**" field of the Hidden Value Action. The format is: ##h ##m or ##h:##m. Example: 1h 30m or 1h:30m.
   - **timed-section-started** -- Javascript will fill this with the time the respondent started the Timed Section, in their local time.
   - **timed-section-complete-by** -- Javascript will fill this with the time by which the respondent must complete the Time Section, in their local time.

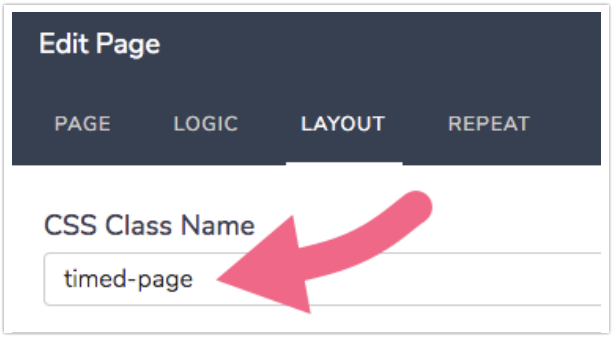


2. Set the page **Layout > Class Name** to **sg-hide** to hide the page



Step 3: Add Timed Pages

Add one or more pages and set the **Layout > Class Name** to timed-page.



These pages ***must*** be completed before the displayed timer expires. If the respondent closes the browser, the timer continues to run. They can return to the timed pages as long as time remains. When the timer expires, the respondent is redirected to the **End Timed Section page,** and any partially answered questions are saved.

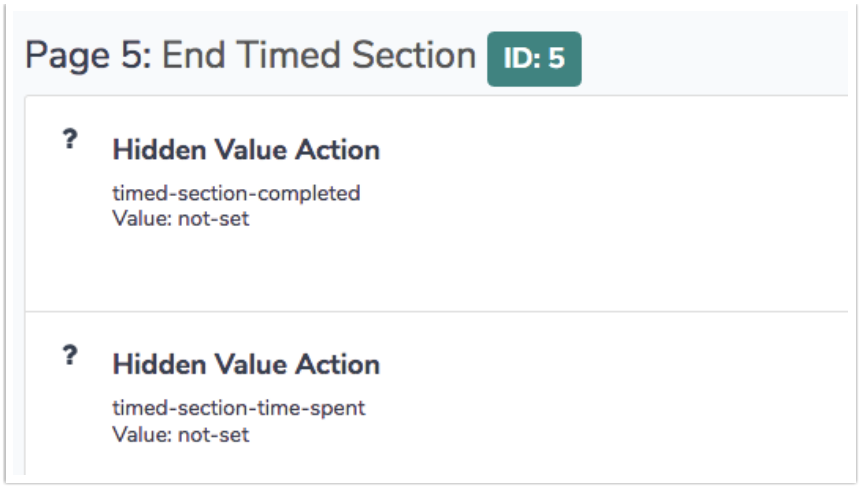
Optional: To remove the **Back button** from the first Timed page add the Javascript Action:

```
document.addEventListener("DOMContentLoaded", function() {
  document.querySelector('#sg_BackButton').classList.add('sg-hide')
})
```

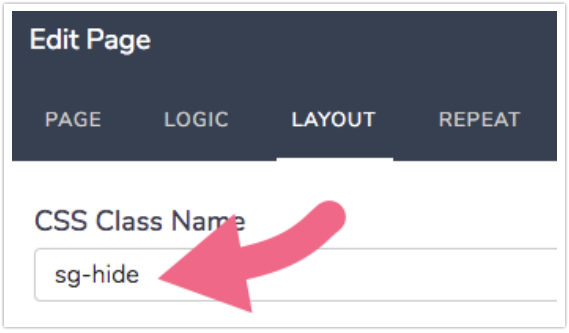
Step 5: Add Hidden End Timed Section page

1. Add a new page with three Hidden Value Actions with specific titles:

    - timed-section-completed -- Javascript will fill this with the time the respondent completed the Timed Section, in their local time.
    - timed-section-time-spent -- Javascript will fill this with the length of time the respondent spent in the Timed Section in hh:mm:ss format.



2. Set the page **Layout > Class Name** to **sg-hide** to hide the page



Step 6: Add Closing Pages

Add as many pages of untimed questions or material as needed, or end with the Thank You page.

If one is not ending with the Thank You page, remove the **Back button** from the first page after the page added in the previous step by adding the Javascript Action:

```
document.addEventListener("DOMContentLoaded", function() {
  document.querySelector('#sg_BackButton').classList.add('sg-hide')
})
```

Step 7: Add CSS

Add the following to **Style tab** > HTML/CSS Editor (lower right of page) > **Custom CSS** to format the countdown timer.

```
/* countdown timer */
.sg-footer-hook-2 {
  text-align: right;
  color:      Gray;
}
```

Step 8: Add Javascript

Add the Javascript below to **Style tab > HTML/CSS Editor (lower right of page) > Custom Header,** and set the highlighted values to the Hidden Value Action IDs from the Start and End Timed Section pages added in Steps 2 and 5 above.

```
<!-- Include standard script that includes $('form').ajaxSubmit() -->
<script src="https://surveygizmolibrary.s3.amazonaws.com/library/664615/jQueryFormPlugin.js"></script>

<script>
// Alchemer v02
// Countdown timer for completing survey, code below functions on Exam Pages in middle of survey
//   and does nothing on non-timer pages

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

 // From the STARTED TIMED SECTION page
 const STARTED_STRING     = `[question('value'), id='32']`
 const COMPLETE_BY_STRING = `[question('value'), id='33']`

 // Page ID of the END TIMED SECTION page
 const FINISH_PID = 5

 // From the END TIMED SECTION page (this is page id: FINISH_PID)
 const COMPLETED_STRING = `[question('value'), id='36']`

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

 // the titles of HVAs on the page BEFORE the timed section
 const SETUP_TITLE       = 'timed-section-setup'
 const STARTED_TITLE     = 'timed-section-started'
 const COMPLETE_BY_TITLE = 'timed-section-complete-by'

 // the page Layout > Class Name for pages IN the timed section
 const TIMED_PAGE_CLASSNAME = 'timed-page'

 // the titles of HVAs on the page AFTER the timed section
 const COMPLETED_TITLE   =  'timed-section-completed'
 const TIME_SPENT_TITLE  =  'timed-section-time-spent'

 const SHORTCUT_TIMER = false // forces the timer to 15 seconds for testing, overriding the SETUP string
 const LOG = true

 /***
  * 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)
     if (LOG) console.error(msg)
     const err = new Error(msg)
     if (LOG) 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
}


/***
 * 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
}


/***
 * getHvaByTitle
 *
 * title (string) title of the HVA
 * return (element) the element for title parameter
 */
const getHvaByTitle = (title) => {
  const qid = getQidByTitle(title)
  assert(qid, `You must have a Hidden Value Action on this page titled: ${title}`)
  return getElemByQid(qid)
}


/***
 * Get HH:MM:SS string from millisec
 *
 * millisec (int) number of milliseconds since 1-1-1970, the birth of Javascript time
 * return (string) in the format hh:mm:ss with leading zeros if needed
 */
const getHHMMSS = (millisec) => {
  // Time calculations for days, hours, minutes and seconds
  let days    = Math.floor(millisec / (1000 * 60 * 60 * 24))
  let hours   = Math.floor((millisec % (1000 * 60 * 60 * 24)) / (1000 * 60 * 60))
  let minutes = Math.floor((millisec % (1000 * 60 * 60)) / (1000 * 60))
  let seconds = Math.floor((millisec % (1000 * 60)) / 1000)


  // build string
  let s = ""
  s += ((hours   < 10)? "0":"") + hours + ":"
  s += ((minutes < 10)? "0":"") + minutes + ":"
  s += ((seconds < 10)? "0":"") + seconds
  return s
```

```
  }


  /***
   * isTimedPage
   *
   * return (t/f)
   */
  const isTimedPage = () => document.querySelector(`.${TIMED_PAGE_CLASSNAME}`)


  /***
   * isOverTimeLimit or have they previously completed the timed section
   *
   * completeByDate (string) the timestamp set in field COMPLETE_BY_STRING on the
   *                      page BEFORE the timed section
   * completedString (string) the timestamp set in field COMPLETED_STRING on the
   *                      page AFTER the timed section or '' if the timed section
   *                      hasn't been completed yet
   * return (t/f)
   */
  const isOverTimeLinit = (completeByDate, completedString) => {
    if (LOG) console.log("isOverTimeLimit, completeByDate = ", completeByDate, ", completedString = ", completed
String, ", (completeByDate - new Date()) = ", completeByDate - new Date())
    // test if over time or the complete field has already been set (meaning the timed section has already been comp
leted)
    return    (completeByDate - new Date()) < 0
          || (completedString && completedString !== 'not-set')
  }


  /***
   * Force submit and move to page AFTER timed section
   *
   * finishPid (int) page ID of the page AFTER timed section
   */
  const forceEndTimedSection = (finishPid) => {
    if (LOG) console.log("$('form') = ", $('form'))
    $('form').ajaxSubmit()
    const url = '[survey("edit link")]&__sgtarget=' + finishPid
    // Allow the event loop and JQuery to process the ajaxSubmit().
    //   Not sure what's up with JQuery since a timeout of '0' doesn't
    //   work despite allowing the even loop to run.  So JQuery's doing
    //   other queueing behind the scenes.  So is 1000 enough?  10 wasn't
    //   but 200 was for a simple page.
    setTimeout(function() {
      window.location.href = url
    },1000)
  }


  /***
   * Display the timer until over time limit (then move to page AFTER timed section)
   *
   * timer (timerId) the id of the 1-second interval timer
   * countDownElem (element) the DIV to display the countdown timer
   * completeByDate (Date) Date to complete the timed section by
   * completedString (string) the timestamp set in field COMPLETED_STRING on the
   *                      page AFTER the timed section or '' if the timed section
   *                      hasn't been completed yet
   */
  const countdownOrExpire = (timer, countdownElem, completeByDate, completedString, finishPid) => {
    if (LOG) console.log("countdownOrExpire()")
```

```
  if (isOverTimeLinit(completeByDate, completedString)) {
    if (LOG) console.log("-- timer expired, leave timed section")
    countdownElem.innerText = "TIME EXPIRED"
    if (LOG) console.log("clearing timer")
    clearInterval(timer)
    forceEndTimedSection(finishPid)
    return
  }


  // Find the timeDiff the 'complete-by' date and now in milliseconds
  let timeDiff = completeByDate - new Date()
  if (LOG) console.log("-- timeDiff = ", timeDiff)


  countdownElem.innerText = "Time remaining: " + getHHMMSS(timeDiff)
}


/***
 * isStartPage()
 *
 * return (t/f) true if this is the page BEFORE the timed section
 */
const isStartPage = () =>
     getQidByTitle(SETUP_TITLE)
  || getQidByTitle(STARTED_TITLE)
  || getQidByTitle(COMPLETE_BY_TITLE)


/***
 * isEndPage()
 *
 * return (t/f) true if this is the page AFTER the timed section
 */
const isEndPage = () =>
     getQidByTitle(COMPLETED_TITLE)
  || getQidByTitle(TIME_SPENT_TITLE)


/***
 * Initialize HVAs on the page BEFORE the timed section and click Next
 *
 * setupTitle     (string) HVA title to get the setup string from in format '3h 15m' or '3h:15m'
 * startedTitle   (string) HVA title to save the started timestamp
 * completeByTitle (string) HVA title to save complete-by timestamp
 */
const initStartTimedSection = (setupTitle, startedTitle, completeByTitle) => {


  /***
   * parse the setup string
   *
   * s (string) Setup string in the form '3h 15m' or '3h:15m'
   * return (obj) in the form { hours: 3, minutes: 15 }
   */
  const parseSetup = (s) => {
    if (SHORTCUT_TIMER)
      return { hours: 0, minutes: 0.25 }
    const regex = /(\d+)h( *| *: *)(\d+)m/ // 3h 15m OR 3h:15m
    const aParsed = s.match(regex)
    assert(aParsed && aParsed.length === 4, `The setup string isn't in the form '3h 15m': ${s}`)
    const h = parseInt(aParsed[1])
    const m = parseInt(aParsed[3])
    if (LOG) console.log(`${h} : ${m}`)
```

```
    return { hours: h, minutes: m }
  }


  /***
   * main()
   */


  if (LOG) console.log("initStartTimedSection()")


  const setupObj       = parseSetup(getHvaByTitle(setupTitle).value)
  const elemStarted    = getHvaByTitle(startedTitle)
  const elemCompleteBy = getHvaByTitle(completeByTitle)


  // first time through survey, set the started and complete-by Hidden Values
  if (!elemStarted.value || elemStarted.value === 'not-set') {


    // set the started HVA
    let now = new Date()
    elemStarted.value = now.toString()
    if (LOG) console.log("elemStarted.value = ", elemStarted.value)


    // set the complete-by HVA
    const completeByDate = new Date(now.getTime() + (setupObj.hours * 60 + setupObj.minutes) * 60 * 1000)
    elemCompleteBy.value = completeByDate.toString()
    if (LOG) console.log("elemCompleteBy.value = ", elemCompleteBy.value)
  }


  // click Next
  document.getElementById("sg_NextButton").click()
}


/***
 * Initialize HVAs on the page AFTER the timed section and click Next
 *
 * startedString (string) Timestamp the timed section was started from the HVA
 *                        on the page BEFORE the timed section
 * completedTitle (string) HVA title to save the completed timestamp
 * timeSpentTitle (string) HVA title to save time spent in hh:mm:ss format
 */
const initEndTimedSection = (startedString, completedTitle, timeSpentTitle) => {
  if (LOG) console.log("initEndTimedSection()")


  const elemCompleted = getHvaByTitle(completedTitle)
  const elemTimeSpent = getHvaByTitle(timeSpentTitle)


  // first time through survey, set the 'completed' and 'time-spent' fields
  if (!elemCompleted.value || elemCompleted.value === 'not-set') {
    if (LOG) console.log("completed page, setting 'completed' and 'time-spent'")


    // set the 'completed' HVA
    let now = new Date()
    elemCompleted.value = now.toString()
    if (LOG) console.log("elemCompleted.value = ", elemCompleted.value)
```

```
    // get 'started' date/time
    assert(startedString, 'Timed section ending but the STARTED Hidden Value is empty')
    const dtStarted = new Date(startedString)


    // Find the timeDiff between 'started' and 'comtpled' in milliseconds
    let timeDiff = now - dtStarted


    elemTimeSpent.value = getHHMMSS(timeDiff)
    if (LOG) console.log("elemTimeSpent.value = ", elemTimeSpent.value)
  }


  // Click Next or Submit to move to next page
  (document.querySelector("#sg_NextButton") || document.querySelector("#sg_SubmitButton")).click()
}


// ************************************************************
// ************************************************************


/***
 * main()
 */


if (LOG) console.log("main()")


if (isStartPage()) {
  if (COMPLETE_BY_STRING && isOverTimeLinit(new Date(COMPLETE_BY_STRING), COMPLETED_STRING)) {
    forceEndTimedSection(FINISH_PID)
    return
  }
  initStartTimedSection(SETUP_TITLE, STARTED_TITLE, COMPLETE_BY_TITLE)
  return
}


if (isEndPage()) {
  initEndTimedSection(STARTED_STRING, COMPLETED_TITLE, TIME_SPENT_TITLE)
  return
}


if (!isTimedPage()) {
  if (LOG) console.log("** this is not a timed page **")
  return
}


assert(COMPLETE_BY_STRING, "Timed section started but the COMPLETE-BY string empty")
const completeByDate = new Date(COMPLETE_BY_STRING)


let timer = null


// get the div in which to show the countdown timer
//    'sg-footer-hook-2' is part of the stadard SG page layout, see Style tab > Custom HTML
const countdownElem = document.querySelector(".sg-footer-hook-2")
```

```
  // Interval will fire every second to display countdown and check if expired
  timer = setInterval(function() {
    if (LOG) console.log("--- timer interval tick ---")
    countdownOrExpire(timer, countdownElem, completeByDate, COMPLETED_STRING, FINISH_PID)
  }, 1000)
  // call countdown() to get the timer to display immediately
  countdownOrExpire(timer, countdownElem, completeByDate, COMPLETED_STRING, FINISH_PID)


  // Cancel timer when page is submitted via Back / Next / Submit
  document.forms[0].onsubmit = function() {
    if (LOG) console.log("** onsubmit() event **")
    if (timer) {
      if (LOG) console.log("clearing timer")
      clearInterval(timer)
    }
  }
})


</script>
```

Related Articles