Integrating Web Applications into Popular Survey Platforms

for Online Experiments

Abstract

Research using custom-made web applications is burgeoning as scholars increasingly conduct their experiments online. We show how researchers can integrate their web applications into popular survey software such as Qualtrics in five simple steps and provide the full JavaScript code and screenshots. This procedure allows participants to seamlessly switch from Qualtrics to their web applications without leaving the survey platform. This integration has two benefits: (1) it eliminates the risk that participants inadvertently drop out of the survey while switching from the survey software to the web application and vice versa; and (2) it saves researchers the fees charged by survey companies to host an external link on the survey platform. Hence, we make it easier to conduct research on targeted (e.g. national) samples using web applications.

Keywords: web applications, online experiments, economic games, survey software, Qualtrics

Word Count: 2,780

Important Links

Demo: LINK

Qualtrics Template: LINK

Introduction

Interactive experiments are increasingly conducted online with custom software (Arechar et al., 2018; Giamattei et al., 2020). Researchers using these designs may benefit from integrating their custom applications into survey software (e.g. Qualtrics, QuestionPro, SoSci Survey, etc.) to acquire targeted samples while mitigating attrition at a reduced cost. For example, researchers may wish to link survey software to applications designed for conducting incentivized experiments about spending and saving (Del Ponte & DeScioli, 2019), investing in public goods (Arechar et al., 2018; Suri & Watts, 2011), voting about taxes and redistribution (Carter et al., 2021), and many more topics in behavioral science, economics, and politics (e.g., DeScioli & Kimbrough, 2019; Greiner et al., 2014; Hahn et al., 2018; Lau & Redlawsk, 2006; Mahéo, 2017; Messing & Westwood, 2014). Integrating these applications with survey software may be helpful for researchers who need to access a nationally representative sample or wish to add custom applications to a larger survey with numerous survey questions and complex features such as randomization, survey logic, or IP screeners. Without integrating web applications into the survey, researchers must typically link participants away from the survey software to the web application's address and then linking them back once the application is complete.

However, there are two challenges: (1) Participants may inadvertently drop out as they exit and reenter the survey to access the web application;[1] (2) Popular survey providers of nationally representative samples typically charge hefty fees just for embedding an external link

---

[1] Although there is no published data on the relationship between using multiple links and participant drop-out, previous work has found higher drop-out rates in more burdensome surveys (Hoerger, 2010).

in their survey.[2] These challenges limit access to nationally representative samples for researchers using economic games or experiments requiring ad hoc web applications.

In this note, we overcome these challenges by providing ready-made JavaScript code and step-by-step guidance to integrate web applications into the Qualtrics survey software. Qualtrics has become increasingly popular for conducting social science experiments thanks to recent software add-ons (Molnar, 2019) and powerful customization tools which allow researchers to conduct elaborate studies featuring complex randomizations, including conjoint experiments (Bansak et al., 2021; Hainmueller et al., 2014). Accordingly, survey firms such as Bovitz and Lucid include Qualtrics among their preferred survey platforms. While the present paper describes how to integrate custom applications into Qualtrics, this tutorial will also be useful for integration into other survey software including QuestionPro, SoSci Survey, and Dynata, among others. The only requirements are that the survey software allows researchers to (1) Write HTML code into survey questions, (2) add JavaScript code on the platform and (3) create and store variables. Not all survey platforms allow this (for instance, SurveyMonkey does not have these capabilities), and each platform may use unique labels or programming rules, so researchers will need to examine the capabilities of their preferred survey software before proceeding.

Some programming knowledge is required, as researchers need to insert the code into their own web application and in Qualtrics. As a reward, participants will enjoy a seamless, customized experience and researchers will avoid fees. A list of online resources to learn the basic skills for programming web applications for behavioral research is available in the Appendix.

---

[2] For instance, in October 2020, the authors received a quote of $1,500 from Qualtrics for this service.

To place a web application within a survey, researchers can use an *iframe* (short for inline frame) which allows content from external websites to appear within the current survey question. Figure 1 shows an example where a custom-made web application for a behavioral political economy experiment is embedded in Qualtrics.

Figure 1. Voting to tax wages in a web application embedded in survey software (The Authors, 2021).
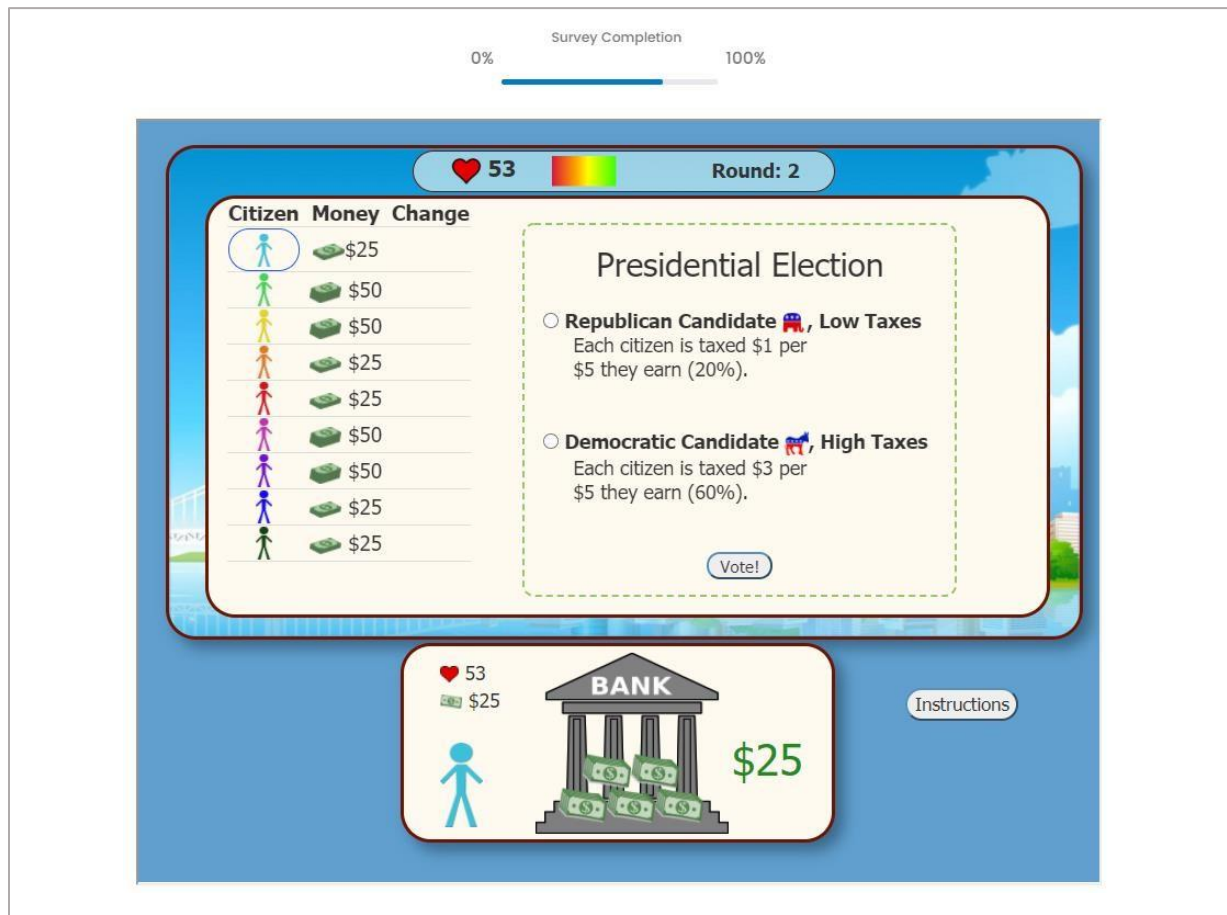
Embedding a web application in survey software in five steps

We describe five simple steps to embed a web application in survey software: (1) Hosting the web application; (2) Creating the iframe; (3) Sharing data between the web application and the

survey; (4) Making the survey advance after participants are done with the web application; (5) Adding event listeners to the survey.

Step 1: Hosting the web application. First, the application must be hosted on a web domain that allows its webpages to be embedded on other sites. Hosting applications on the web domain of an employer or university may lead to integration problems if the organization has banned remote access to its webpages. This often results in a message that the iframe window "refused to connect" to the host domain. Hence, it is often best to host the application on a personal webpage or on a web domain like GitHub where remote access is allowed by default. Another convenient option is to use R-Shiny applications, which integrate the R statistical environment (R Core Team, 2014) with the web (Beeley, 2016). R-Shiny apps can be hosted for free on the shinyapps.io domain. (Links to tutorials on building R-Shiny apps and uploading software to the shinyapps.io domain are available in the Appendix.)

Step 2: Creating the *iframe*. Once the application is hosted, researchers can load the application into a survey question by adding an iframe to the question's JavaScript code editor (appendix, Figure S1). By default, Qualtrics includes three JavaScript sections which implement user code when survey is loading for the first time, when the survey is fully displayed, or when the survey is unloaded (appendix, Figure S2). Researchers can add their applications to the header (top) of a survey question by including code that creates an iframe in the addOnload section as follows:

```
Qualtrics.SurveyEngine.addOnload(function()
{
  /*Place your JavaScript here to run when the page loads*/
        jQuery("#Header").html('<iframe allowfullscreen="" src="myapplication.html"
        style="width:90%; height:625px;" id="myframe"></iframe>');
});
```

In the code, researchers should replace "myapplication.html" with the web address of their application. The width and height style attributes can be adjusted to fit the application to the survey window. This code sets the iframe id to "myframe" so it can be referenced later. Now the survey will be able to load the web application within the iframe once the corresponding survey question is loaded.

Step 3: Sharing data from the web application to the survey. The researcher's web application must be able to share information with the survey platform and store this information in variables. For security reasons, web documents cannot usually interact with the code of webpages contained in their iframes. This means that a researcher cannot manipulate their own web application and interact with its variables directly from the survey's JavaScript. However, researchers can work around this limitation by programming their applications to send information from their web application in the body of a *message event*.[3]

Message events are tools which web developers use to send signals from their web application to the user's browser at certain points of an application's execution. Importantly, researchers can program their surveys to *listen* for the application's message events and *handle* these messages by storing the contents as variables. Event listeners can be programmed to scan the browser for messages containing specific content (for instance, the presence of the text "ID"). Then, event handlers can be used to record this information and store it within a variable on the survey platform.

In one's web application, it may be beneficial to create variables that uniquely identify participants and to store this information on the survey platform. In this example, we generate a

---

[3] R-Shiny users can reference https://shiny.rstudio.com/articles/js-send-message.html for guidance on creating messages and events in R-Shiny, after which they can return to Step 5 of this article to create listeners in Qualtrics.

participant ID variable as a string of integers with the prefix "ID" (ex: "ID123456"). The prefix allows IDs to be easily distinguished from other messages that researchers might send to the survey platform. Researchers can create this variable by adding JavaScript to their web application as follows:

```
/* Web Application: Creating a unique ID variable */
        var participantID = "ID" + Math.random().toString().substring(2,7);
```

Next, researchers can program the application to post message events that pass the ID values on to the survey software as follows:

```
/* Web Application: Posting the value of the user-created variable "participantID" */
        window.parent.postMessage(participantID, "*");
```

In the code, the asterisk specifies that *any* parent website hosting the application can receive the message being sent. If the researcher wishes to specify a single web domain which can receive a message and exclude others, this can be done by replacing the asterisk with the web domain's address (for instance, "https://qualtrics.com"). After this step, the web application will now be able to load in the survey window, generate ID variables, and send them in message events to be saved by Qualtrics.

Next, researchers need to create the corresponding ID variable in the survey software. In Qualtrics, this can be done by creating new embedded data fields in the survey flow (appendix, Figure S3).

Step 4: Making the survey advance after participants are done with the web application.
Researchers may also wish to send their survey a signal that participants have completed the task

in their web application, such that the survey can advance to the next stage. This can be done by adding JavaScript near the end of an application's code as follows:

```
/* Web Application: Posting the string "complete" to signal the end of the application*/
        window.parent.postMessage("complete", "*");
```

Step 5: Adding event listeners to the survey. The final step is to program the survey to listen for iframe message events and to take actions if messages are observed. Event listeners can be programmed to distinguish between ID and completion messages in the onload section as follows:

```
Qualtrics.SurveyEngine.addOnload(function()
{
  /*Place your JavaScript here to run when the page loads*/

        /*Create iframe*/
        jQuery("#Header").html('<iframe allowfullscreen="" src="myapplication.html"
        style="width:90%; height:625px;" id="myframe"></iframe>');

        /*Create variables for event listeners*/
        var eventMethod = window.addEventListener ? "addEventListener" : "attachEvent";
        var eventListen = window[eventMethod];
        var messageEvent = eventMethod == "attachEvent" ? "onmessage" : "message";

        /*Create event listeners and handlers*/
        eventListen(messageEvent, function (e) {
                /*If application posts "complete" message, hide the iframe and advance the
                survey */
                if (e.data == "complete"){
                        document.getElementById("myframe").style = "display: none";
                        qobj.clickNextButton();
                        }
                /*If application posts message with "ID" store as 'participantID' */
                else if(e.data.includes("ID")){
                        Qualtrics.SurveyEngine.setEmbeddedData('participantID', e.data);
                        }
                }, false);
});
```

This code creates event listeners in the survey which wait for the posted message, e.data, to equal "complete" before hiding the *iframe* by setting its display attribute to "none". The code also stores the ID data in Qualtrics for each respondent as embedded data.

Bonus step for incentivized experiments. For incentivized experiments, it may be useful to store participants' bonus data in the survey platform. This can be done by passing the data saved in their web application's "bonus" variable to the survey in a message event.

```
/* Web Application: Posting the value of the user-created variable "bonus" */
        window.parent.postMessage(bonus, "*");
```

Then, similar to the steps for completion and participant ID messages, one can append to the section containing event listeners and handlers contained in the onload function an event listener for bonus messages as follows:

```
/*If application posts message with numeric value, store as 'bonus' */
else if(e.data != "complete" && e.data < 1000){
        Qualtrics.SurveyEngine.setEmbeddedData('bonus', e.data);
        }
```

Final recommendations. We advise researchers to store identification variables in both the web application and survey such that the two resulting datasets can be merged. We also recommend placing all screening questions in a separate block before the web application question to ensure that respondents are screened out of the survey before they start working on the task in the web application. Finally, we recommend preventing participants from advancing in the survey until the application is complete by hiding the continue button while the *iframe* is visible.

Discussion

In this note, we have outlined how researchers can integrate their web applications into popular survey software. We have used Qualtrics to illustrate the process due to the popularity of this survey software, which is available at no cost to many academic researchers. Integrating web applications into survey platforms other than Qualtrics will require researchers to follow the rules of each platform, but the basic steps outlined here still apply. Users should host the web app and embed it in a survey question using an *iframe*. If researchers wish to share data from the web application to the survey and make the survey advance in response to participants' progress

in the web application, researchers should use JavaScript notifications and event listeners to pass messages from the application to the survey software.

We hope that this tutorial will allow more researchers to conduct behavioral research using web applications on targeted or nationally representative samples. Easier and more affordable access to targeted samples will help experimental researchers address concerns about external validity and the generalizability of results obtained using convenience samples (for a discussion, see e.g. Druckman & Kam, 2011; Krupnikov & Levine, 2014; McDermott, 2011; Mullinix et al., 2015).

References

Arechar, A. A., Gächter, S., & Molleman, L. (2018). Conducting interactive experiments online. *Experimental Economics*, *21*(1), 99–131. https://doi.org/10.1007/s10683-017-9527-2

Bansak, K., Hainmueller, J., Hopkins, D. J., Yamamoto, T., Druckman, J. N., & Green, D. P. (2021). Conjoint survey experiments. *Advances in Experimental Political Science*, *19*.

Beeley, C. (2016). *Web application development with R using Shiny*. Packt Publishing Ltd.

Carter, B., Del Ponte, A., & DeScioli, P. (2021). Give Some to Get Some: Do Voters Understand Who Benefits from Higher Taxes? *Working Paper*.

Del Ponte, A., & DeScioli, P. (2019). Spending too little in hard times. *Cognition*, *183*, 139–151.

DeScioli, P., & Kimbrough, E. O. (2019). Alliance formation in a side-taking experiment. *Journal of Experimental Political Science*, *6*(1), 53–70.

Druckman, J. N., & Kam, C. D. (2011). Students as experimental participants. *Cambridge Handbook of Experimental Political Science*, *1*, 41–57.

Giamattei, M., Yahosseini, K. S., Gächter, S., & Molleman, L. (2020). LIONESS Lab: A free web-based platform for conducting interactive experiments online. *Journal of the Economic Science Association*, *6*(1), 95–111. https://doi.org/10.1007/s40881-020-00087-0

Greiner, B., Caravella, M., & Roth, A. E. (2014). Is avatar-to-avatar communication as effective as face-to-face communication? An Ultimatum Game experiment in First and Second Life. *Journal of Economic Behavior & Organization*, *108*, 374–382.

Hahn, K. S., Lee, H.-Y., Ha, S., Jang, S., & Lee, J. (2018). The Influence of "Social Viewing" on Televised Debate Viewers' Political Judgment. *Political Communication*, *35*(2), 287–305.

Hainmueller, J., Hopkins, D. J., & Yamamoto, T. (2014). Causal inference in conjoint analysis: Understanding multidimensional choices via stated preference experiments. *Political Analysis*, *22*(1), 1–30.

Hoerger, M. (2010). Participant dropout as a function of survey length in Internet-mediated university studies: Implications for study design and voluntary participation in psychological research. *Cyberpsychology, Behavior, and Social Networking*, *13*(6), 697–700.

Krupnikov, Y., & Levine, A. S. (2014). Cross-sample comparisons and external validity. *Journal of Experimental Political Science*, *1*(1), 59.

Lau, R. R., & Redlawsk, D. P. (2006). *How voters decide: Information processing in election campaigns*. Cambridge University Press.

Mahéo, V.-A. (2017). Information campaigns and (under) privileged citizens: An experiment on the differential effects of a voting advice application. *Political Communication*, *34*(4), 511–529.

McDermott, R. (2011). Internal and external validity. *Cambridge Handbook of Experimental Political Science*, 27–40.

Messing, S., & Westwood, S. J. (2014). Selective exposure in the age of social media: Endorsements trump partisan source affiliation when selecting news online. *Communication Research*, *41*(8), 1042–1063.

Molnar, A. (2019). SMARTRIQS: a simple method allowing real-time respondent interaction in qualtrics surveys. *Journal of Behavioral and Experimental Finance*, *22*, 161–169.

Mullinix, K. J., Leeper, T. J., Druckman, J. N., & Freese, J. (2015). The generalizability of survey experiments. *Journal of Experimental Political Science*, *2*(2), 109–138.

R Core Team. (2014). *R: A language and environment for statistical computing*. R Foundation

for Statistical Computing.

Suri, S., & Watts, D. J. (2011). Cooperation and contagion in web-based, networked public

goods experiments. *PloS One*, *6*(3), e16836.

Appendix for

Integrating Web Applications into Popular Survey Platforms
for Online Experiments

Supplementary Figures

Figure S1. Researchers can access the JavaScript editor in Qualtrics by (1) navigating to the survey tab and (2) selecting the gray gear to the left of the question to which they want to add their web Application.
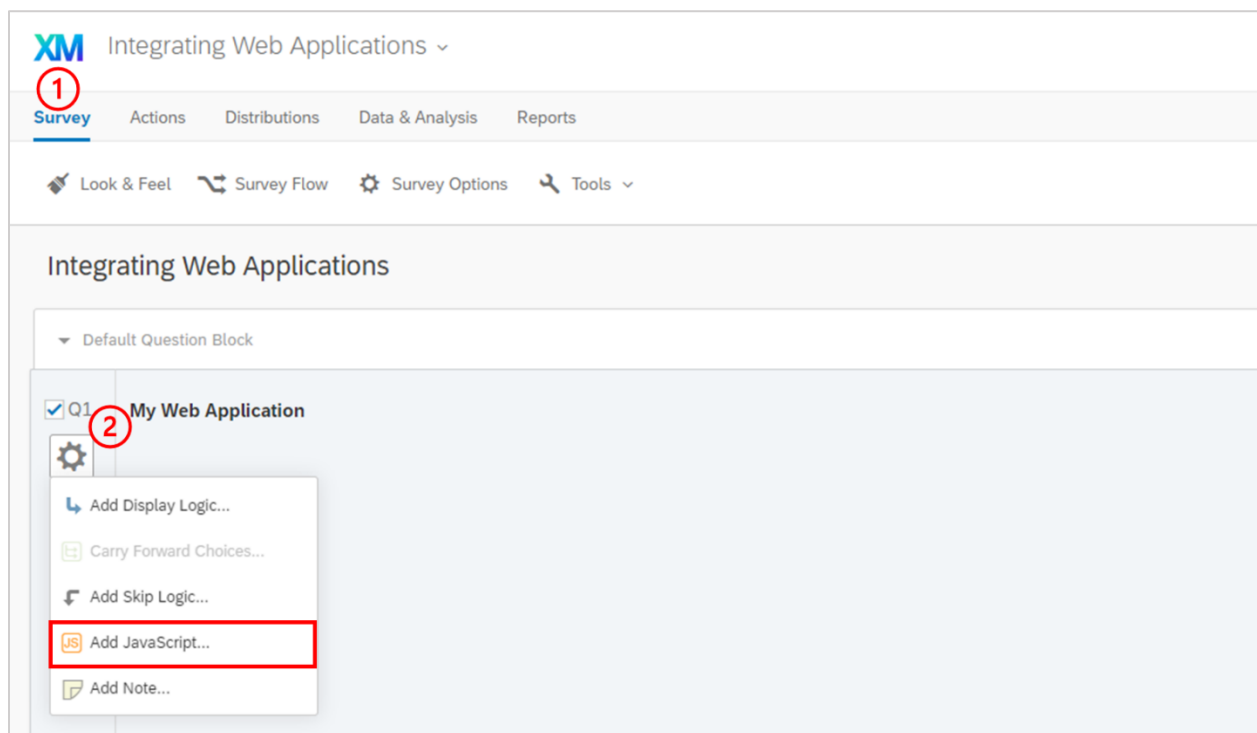
Figure S2. The default JavaScript code for a Qualtrics question.

Edit Question JavaScript

```
Qualtrics.SurveyEngine.addOnload(function()
{
    /*Place your JavaScript here to run when the page loads*/

});
Qualtrics.SurveyEngine.addOnReady(function()
{
    /*Place your JavaScript here to run when the page is fully displayed*/

});
Qualtrics.SurveyEngine.addOnUnload(function()
{
    /*Place your JavaScript here to run when the page is unloaded*/
```

Full Screen   ✖ Clear   Cancel   ✔ Save

Figure S3. Adding variables to the embedded data in Qualtrics. Simply open the survey flow, select embedded data, and enter the name of the variables you would like to create. In this example, we name the ID variable "participantID" and the bonus variable (only for incentivized experiments) "bonus".

Further Online Resources

Below, we provide a brief list of online tutorials where beginners can learn basic web design for creating apps in JavaScript or hone their skills.

- Introductory Programming for Social Science Experiments (especially modules 3 and 4): https://socsciprogramming.github.io/module3.html
- R-Shiny Apps:
  - o Tutorial on designing web applications in R-Shiny: https://shiny.rstudio.com/tutorial/
  - o Tutorial on uploading software to the shinyapps.io domain: https://shiny.rstudio.com/articles/shinyapps.html.
  - o Tutorial on creating messages and events in R-Shiny: https://shiny.rstudio.com/articles/js-send-message.html.
- Basic web design: https://www.w3schools.com/