Integrating Web Applications into Popular Survey Platforms for Online Experiments

Benjamin Carter[*] & Alessandro Del Ponte[†]

Abstract

Amid the Covid-19 pandemic, research using custom-made web applications is burgeoning as researchers increasingly conduct their experiments online. Moreover, integrating web applications and survey software is needed when collecting data using nationally representative samples from survey companies. 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 samples (e.g. nationally representative) using web applications.

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

Important Links

Demo: LINK

Qualtrics Template: LINK

[*] Center for Behavioral Political Economy and Department of Political Science, Stony Brook University
[†] Global Asia Institute, National University of Singapore

Introduction

As interactive experiments are conducted online amid the Covid-19 pandemic, researchers may increasingly need to create web applications for their studies and link their custom application to their survey software (e.g. Qualtrics). For example, researchers may wish to link survey software to custom applications designed for conducting experiments using economic games (e.g. Del Ponte & DeScioli, 2019; DeScioli & Kimbrough, 2019; Greiner et al., 2014), political communication experiments (e.g. Hahn et al., 2018; Mahéo, 2017; Messing & Westwood, 2014) or other custom applications. Moreover, integrating a web application with survey software may be an unavoidable step if these researchers wish to access a nationally representative sample. This is typically done by linking participants away from the survey software to the web application's address and then linking them back once the application is complete.

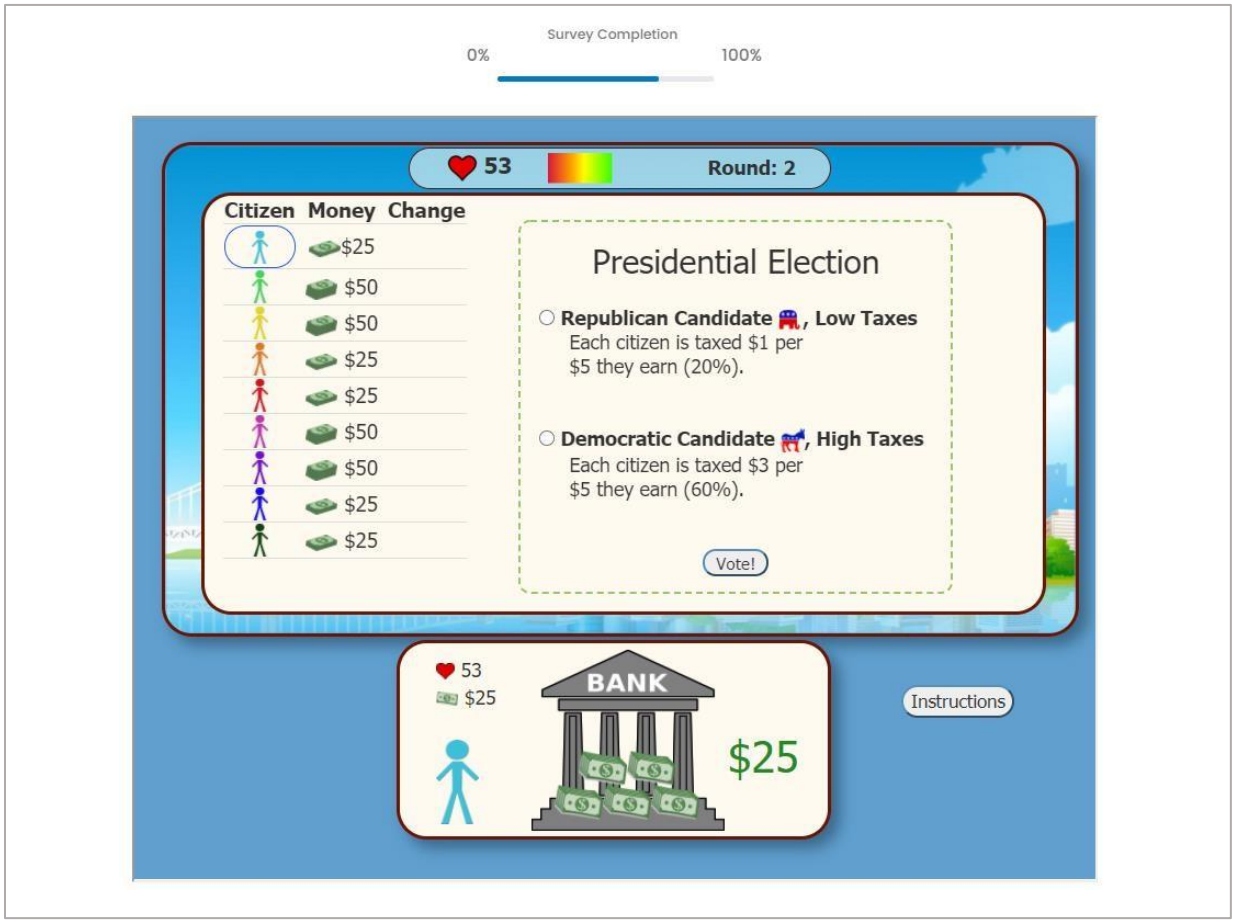
However, there are two challenges: (1) Participants may inadvertently drop out as they exit and reenter the survey to access the web application; (2) Popular survey providers of nationally representative samples typically charge hefty fees just for embedding an external link in their survey[1]. These 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 popular survey software such as Qualtrics. Some programming knowledge is required, as researchers need to insert the code into their own web application and in Qualtrics. However, as a reward, participants will enjoy a seamless experience and researchers will avoid fees.

Survey platforms such as Qualtrics allow users to write custom JavaScript code into each survey question, making it possible to host external applications within a survey window. 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. Our working example uses the Qualtrics survey platform. However, the methods and much of the code will remain helpful for integrating web applications into similar online platforms such as Dynata or others. Figure 1 shows an example where a custom-made web application for a behavioral political economy experiment is embedded in Qualtrics.

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

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.

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*.

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 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 hope that this allows more researchers to conduct experimental research using web applications on nationally representative samples. Easier and more affordable access to nationally representative 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

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.

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.

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

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.

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.