JavaScript Injections

David Kelly

kellyd8@duq.edu

Duquesne University

600 Forbes Avenue, Pittsburgh, PA 15282 USA

December 12, 2017

Abstract

JavaScript injections, the process by which an individual can insert and use their own JavaScript code in a page – whether that be due to entering code into the address bar or through an exploit of a website’s cross site scripting, is a well-known problem in web development. The idea behind JavaScript injections are normally from a straightforward mindset: make the page act in such a way that is different from its original intention. The further mindset behind JavaScript injections are to use this exploit to make a page work in a way that is beneficial to you and/or detrimental to the owner(s) of the website which has the vulnerability.

1. Introduction

One of the most common things to do inside the realm of web development is to create a webpage and potentially have some product on that webpage which you, as an individual, market to create some additional income. Interacting with that webpage requires a few things: a webpage being created with pertinent information displayed on it, a way for users to get to that page, a way for users to purchase your product, and a means by which you can handle the transaction between the customer and your webpage.

Creating a webpage with a product to purchase and driving traffic to said page are only a matter of creating an HTML document and picking a domain name so potential customers can get to your page and see your product to purchase it. The deceivingly complicated part is handling the transaction between your customers and webpage. In this paper, we explore this problem.

1. Background
   1. HTML

HTML is the standard markup language for creating web pages. Each element is represented by a set of tags and these elements are the basic building blocks for a web page. Tags are items that are surrounded by angle brackets. Each set of tags defines a certain structure to the web page on how to display the contents of the web page.

* 1. CSS

CSS stands for Cascading Style Sheets. It describes how elements are displayed on screen, paper, or in other social media. It is used to define styles for webpages, including design, layout, and variations in display for different devices and screen sizes.

* 1. JavaScript and jQuery

JavaScript is a lightweight, object-oriented language with first class functions. It is most commonly attributed with being the scripting language for webpages. JavaScript runs on the client side of the web – which is used to design how the page reacts to certain events. Generally, these events are tied to a user’s actions which are, in turn, tied to manipulation of elements in the Document Object Module (DOM). Libraries such as jQuery make interaction with the DOM easier.

For example, in order for JavaScript to manipulate an element of the DOM with the id name ‘checkout’ and change its value to ‘7’, you would need to do the following:

document.getElementById(‘checkout’).innerHTML(‘7’);

To accomplish the same task with jQuery, we would do the following:

$(‘#checkout’).html(‘7’);

The $ denotes the document. The string inside of the first set of parenthesis denotes the class (denoted with a ‘.’ followed by the class name rather than the ‘#’) or id that you are manipulating. The second set of parenthesis represents what you are changing inside of the DOM element’s properties (in this example, the inner HTML is being changed with the .html() method). As you can see, it is much simpler to manipulate the DOM with jQuery over conventional JavaScript and, as such, jQuery is used to manipulate the DOM in the source code.

* 1. Injecting code into a Browser

JavaScript code can be injected into a browser in multiple ways. You can probe for a reflected value and enter the following to escape it and then insert your own code – “; . You can also enter code into the browser by prefixing any javascript code with javascript:. You could also open the developer console on your browser and enter javascript code directly into there. It is pertinent to know that refreshing the browser window removes any injected code.

1. Materials and Methods
   1. Code Location and Setup

For this experiment, I’ve created a test webpage that will serve as the example in this paper. The source code can be found at:

www.github.com/DaveRKelly/JavaScript-Injection

We will be using the code inside of the JSInjectable folder. I created a http server with python to host my code locally to simulate a malicious attacker’s viewpoint. My setup uses Python 2.7.10. Navigate to the JSInjectable folder in the terminal and create a python server:

python –m SimpleHTTPServer

* 1. Probing for Vulnerabilities

Loading the page and messing around it quickly shows that the page responds to changes to the item quantity, coupon code, and checkout button. Looking at the script.js file inside the js folder shows that there are some acceptable coupon codes and the discounts for each code:

let acceptableCouponCodes = [‘blackfriday’, ‘cybermonday’, ‘thanksgiving’, ‘christmas’];

let discounts = [.2, .4, .6, .8];

Looking further, it seems that changes to the item quantity or entering a coupon code causes the page to recalculate the total price. Furthermore, when you click the checkout button, the page simply gets the value in the total row and shows as we have paid that much.

* + 1. Exploiting the Coupon Codes that are Accepted

One way in which we can exploit the page is by simply adding our own coupon code which is, conveniently, 100% off – meaning we don’t pay anything. We can open the developer console in whichever browser we are in and enter:

acceptableCouponCodes.push(‘ourFakeCode’);

discounts.push(1);

Now, if we go to the coupon code box and enter ourFakeCode, suddenly the price of whatever we want is $0.00 and we can take as much as we like. (See Figures 4.1 and 4.4)

* + 1. Exploiting the Individual Item Prices as Fields of the HTML document

Another way in which a setup like this can be exploited is by changing the individual item prices. Again, by looking at our script.js, we see that when you click on the plus or minus button to change the quantity, the calculatePrice function is called and returns the product of its two values – one of which is the field of the item in the HTML document. That field seems to be the base price for the item. Changing that means that we effectively make the item cost $0.00 and then we can order as much as we want. We can change that by entering the following into our developer console:

$(‘.item1price’).attr(“field”, 0);

$(‘.item2price’).attr(“field”, 0);

$(‘.item3price’).attr(“field”, 0);

$(‘.item4price’).attr(“field”, 0);

Now all we have to do is update the quantity and, suddenly, our total price has dropped to $0.00 again. (See Figures 4.2 and 4.4)

* + 1. Exploiting the Total Price Displayed in the Page

Yet another way that we can change the price that we pay is by changing the total price entirely. This can be done by typing:

$(‘.thick’).html(formatPrice(0));

Now, we can notice that the price has been changed to $0.00 and we can simply checkout from here. (See Figures 4.3 and 4.4)

1. Results

Looking at what we have accomplished, we have successfully documented three ways in which you can use JavaScript to change the amount that you pay as a customer to a website. In all three instances that we have explored, the injection was successful.

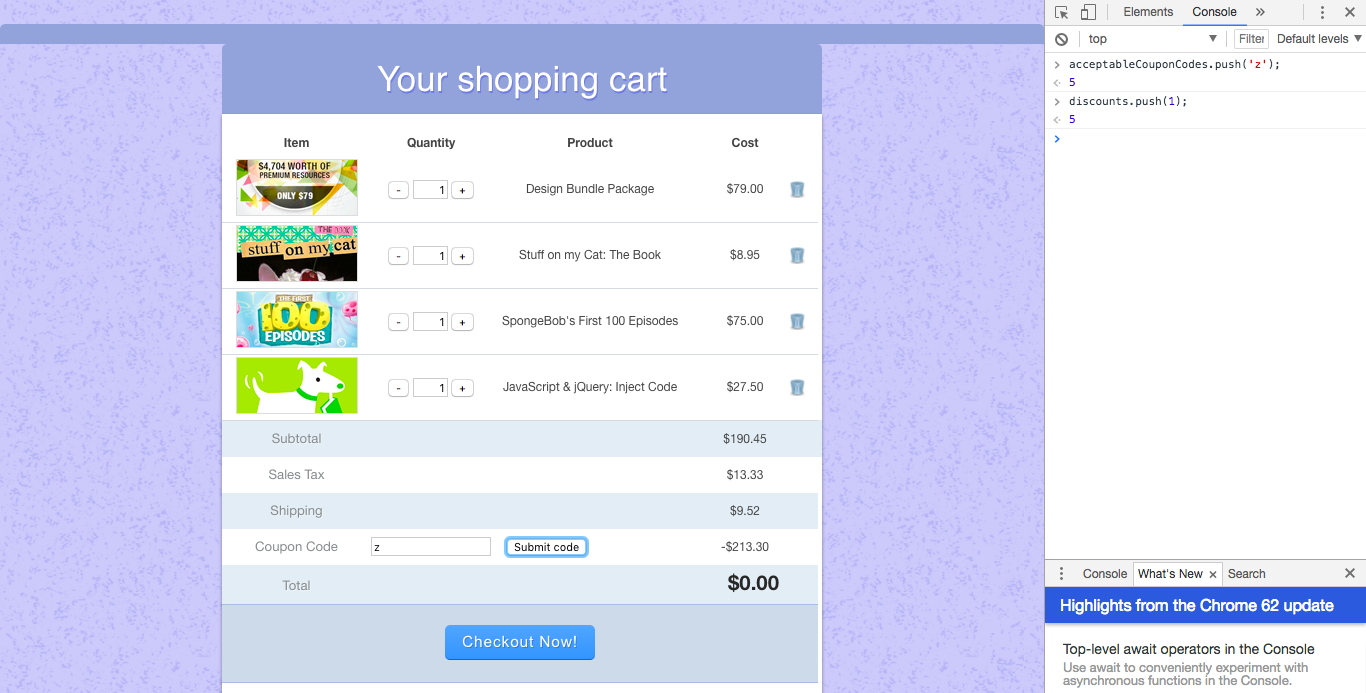


Figure 4.1: Exploiting accepted coupon codes

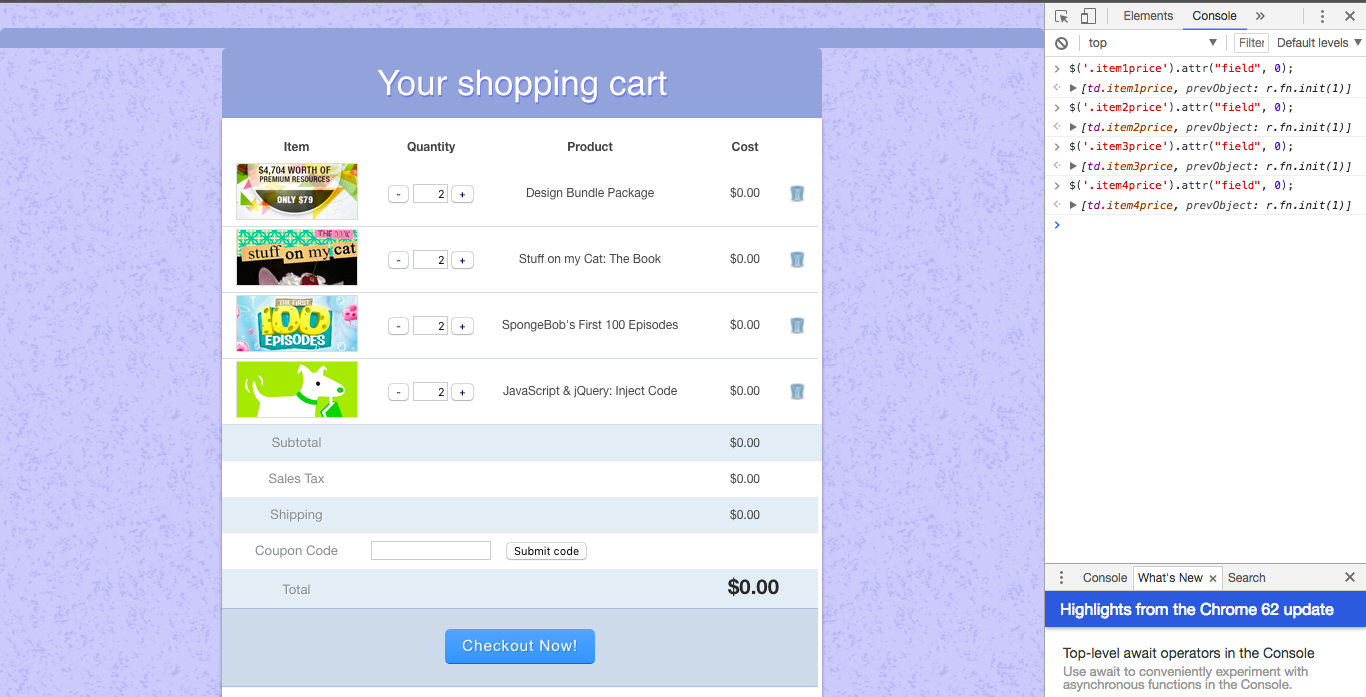


Figure 4.2: Exploiting individual item prices as fields of the HTML document

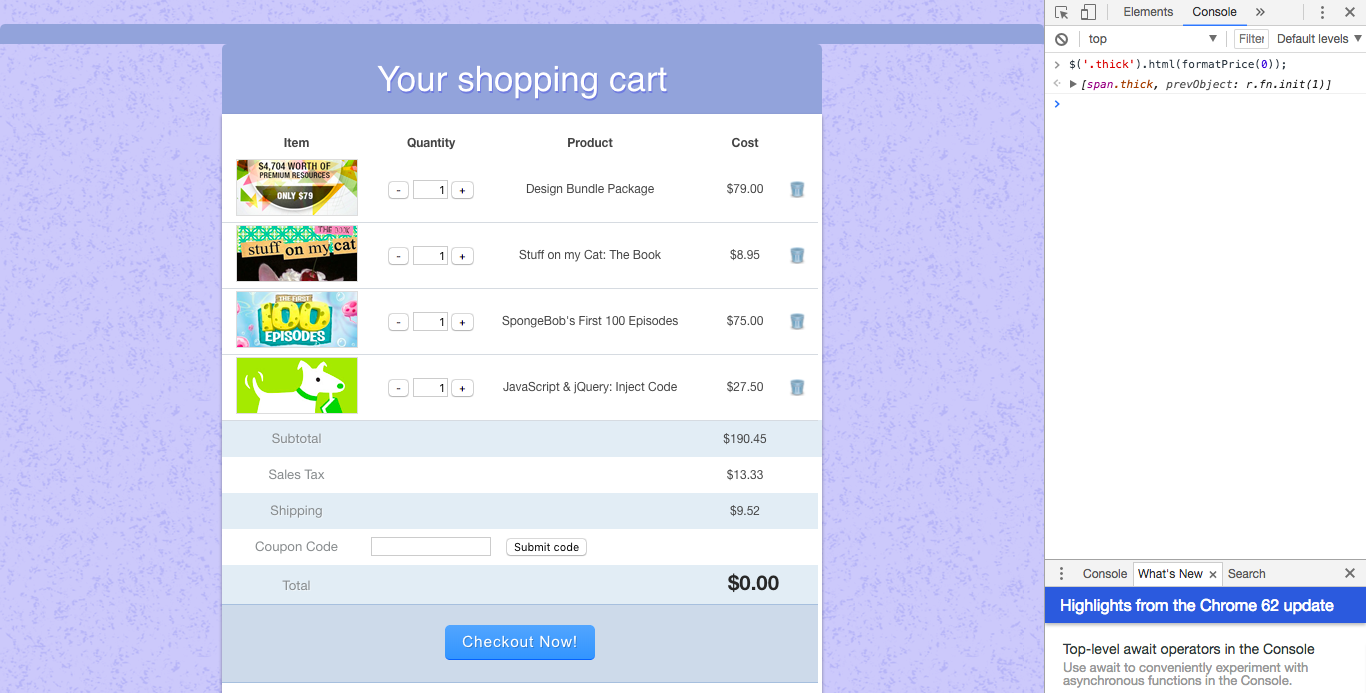


Figure 4.3: Exploiting the total price displayed in the page

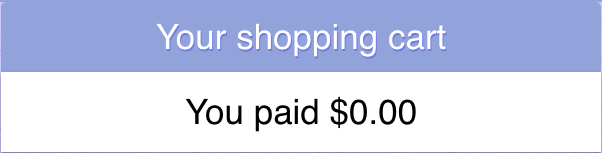


Figure 4.4: The result of our exploitation and injection

1. Discussion and Conclusions

The primary takeaway from this paper is that JavaScript injection can be a very serious threat. In cases like those shown in this paper, the owner of the website could lose several thousands of dollars if a malicious attacker notices this vulnerability and exploits it.

There are ways in which you can prevent against attacks like this is however. In a purely JavaScript approach, you can use objects to create private methods and only allow those methods to be accessed through a public object that your form interacts with. An example of such an object would be a database with public accessor methods and private methods that perform all of the calculations and hide the prices of the items to prevent them from being changed due to a loss of scope. Another layer of security that you can add on top of this is to freeze that newly created object with Object.freeze(obj). Doing so prevents any changes to the object and, assuming your document has no other way of accessing the database, no one can make unauthorized changes the database.

An alternative to using a database as an object in JavaScript is to use a RDBMS such as SQL. This brings up problems outside of the scope of this paper such as SQL injections that would be something to worry about and prevent.

1. References

W3Schools. (2017) Introduction to HTML

W3Schools. (2017) CSS Introduction

MDN Web Docs. (2017) About JavaScript

W3Schools. (2017) jQuery Introduction