Web Development – Mr. Turner

Project – Dynamic Events

**Project Overview**

The purpose of this project is to create an interface whereby a user can simply flip cards onto a table and then put them back into the deck.

**The Page**

*Step 1*

At the top of your page should be a single image element showing the back of a card. Whenever the user clicks on this image, a card will appear on the screen below it. As the user continues to click on the image, more and more cards will continue to appear.

In order to build this page, you must create a deck of card objects. Each click removes a card from the top of the deck.

In order to show the card on screen, you need to add an image element to the page and show the appropriate image as it relates to the card.

When the deck is empty, no more cards should appear.

*Step 2*

The interface for putting the cards back into the deck should be a simple click. If the user clicks on the card, the image will disappear from the screen and the object will go back into the deck.

In order to make this happen, you will need to keep track of which card is represented by each image. You will also need to add a click event to each image as it is created.

As per Step 1, you are creating the image element, setting its source, and appending it to the page.

You also need to assign it a click event. You may find different ways to do this online, but the best way is by calling the **addEventListener** function for the new image object.

**addEventListener** takes two parameters. The first parameter is the event itself, passed as a string. We’re going to pass a click event.

***Note: when you define an inline event, it is preceded by the word “on” (onclick, onmouseover, etc…). When passing an event type as a parameter to addEventListener, we drop the “on”.***

The second parameter is the function that is called by the event. Passing a function can be done in one of two ways.

You can define the function as normal and then pass it by name.

function doSomething()

{

alert (“I’ve done something”);

}

myElement.addEventListener(“click”, doSomething);

The reference to the function does not have a parameter list because it is ***not*** a function call. It is a reference to the function as an ***object***.

You can also pass the function anonymously.

myElement.addEventListener(“click”, function(){alert(“I’m anonymous”)});

In this example, the function is created as the parameter.

Note that, in neither of these examples, does the function take parameters. You cannot pass parameters to functions in this way. If you want to pass parameters, you’ll need to combine these two methods and have the anonymous function call the regular function.

Let’s assume, we need the function to change the innerHTML of myElement. In order for this to work, we would need to create an anonymous function that calls the regular function and passes through the element.

function doSomethingToElement(element)

{

element.innerHTML = “I’ve done something.”;

}

myElement.addEventListener(“click”, function(){doSomethingToElement(this);});

The above example will change the innerHTML of myElement when it is clicked.

\*\*\*

For this project, you will add an event listener to every image that you create. That click event will trigger the return of the card object to the deck and the removal of the image from the screen.

**Programming Requirements**

|  |
| --- |
| * You must code your own HTML and CSS. Use of a drag and drop interface or the usage of code downloaded from the internet is not permitted. |
| * Use of deprecated code is not permitted. Use an HTML 5 reference as your guide. |
| * You must comment your name into the top of every page. |
| * Your code must be structured in a consistent and legible manner |
| * Your text must be organized through the use of sectioning tags (div, span, p, h1 – h6). Tables are acceptable for data but not for page design. |
| * Your interface must be smooth and easy to figure out for a client. |
| * Your pages must be formatted using CSS. |