1. What is a target?
   1. The target tells us what element generated the event.
   2. A target holds the object on which the event occurred, it can be different kinds of objects, but most often it is an element object.
   3. Simple example: when clicking on the button:

Graphical user interface

Description automatically generated

1. What does this line of code do: getElementsByTagName(“img”); ?
   1. We should expect to get a list of all <img> tags in our HTML code.
   2. So that line of code can return many elements, one element or even zero elements depending on how many images we have in our page.
   3. That list can be treated like an array, but it’s actually an object called a NodeList.
2. How many threads of control does a browser have?
   1. Page 381-383, we should know that most JavaScript is written to react to events, those kind of events could be user clicking on a page, data arriving from the network, timers expiring in the **browser**, changes happening in the DOM and may more other threads of control the **browser** can have.
   2. Understanding that the **browser** retrieves a page and all of that page’s contents and then renders the page is not what the **browser** only doing, but behind the scene the **browser’s** doing a lot more than just that such as:
      1. Knows when the page is fully loaded.
      2. Keeps track of all clicks.
      3. Knows when a user submits a form
      4. Watches the click and manages timers and timed events
      5. Watches all mouse movement
      6. Identifies the geo location of the browser
   3. **So we can say we have a single thread to run all JavaScript in a given page, and that single thread is being distributed to sub threads that would be responsible for all the variety of events that the browser renders.**
3. What is the name of the property of an event object to know when an event happened?
   1. The **target (event.target)** event property is an event object to know when an event happened.
4. Are events handled synchronously or asynchronously? Why?
   1. Event handled asynchronous.
   2. Because code written to handle events is different from code that executes top to bottom and then completes. Event handlers can run at any time and in any order.
5. What is an event handler’s main purpose?
   1. Event handlers purpose is to handle events. Handlers are typically small pieces of code that know what to do when an event occurs. In terms of code, a handler is just a function. When an event occurs, its handler function is called
6. List and define all the events discussed in Chapter 9. (Hint: Event Soup)
   1. click
      1. Get this event when you click (or tap) in the page.
   2. load
      1. The event you get when the browser has completed loading a web page.
   3. mousemove:
      1. when you move your mouse over an element you’ll generate this event.
   4. keypress:
      1. This event is generated every time you press a key.
   5. unload:
      1. This event is generated when you close the browser window, or navigate away from a web page.
   6. mouseover:
      1. When you put your mouse over an element, you’ll generate this event.
   7. mouseout:
      1. When you move your mouse off an element, you’ll generate this event.
   8. resize:
      1. Whenever you resize your browser window, this event is generated.
   9. dragstart:
      1. If you drag an element in the page, you’ll generate this event.
   10. touchstart:
       1. On touch devices, you’ll generate a touchstart event when you touch and hold an element.
   11. play:
       1. You will get this event when you click the play button in your <video> tag.
   12. pause:
       1. This one when you click the pause button.
   13. drop:
       1. You’ll get this event when you drop an element you’ve been dragging.
   14. douchend:
       1. You’ll get this event when you stop touching
7. Older versions of Internet Explorer have a different event model from other browsers. Discuss what they are and how they work.
   1. IE8 and older browsers do support most of the “on” properties you can use to assign event handlers.
   2. IE8 and older browsers use a method named attachEvent instead of the addEventListener method.
   3. When an event is triggered and your event handler is called, instead of passing an event object to the handler, IE8 and older versions store the event object in the window object
   4. And here is a simple example to demonstrate how they work:

Diagram

Description automatically generated with medium confidence

1. Consider the code: for (var i = 0; i < images.length; i++) { images[i].onclick = showAnswer; } How would you alter this to set the normal (not blurred) image on mouse over and reblur the image on mouse out? (It’s 2 lines of code)

**for (var i = 0; i < images.length; i++) {**

**images[i].onmouseover = showAnswer;**

**images[i].onmouseout = reblur;**

**//We should consider removing the timer (setTimeout(reblur, 2000, image) from the // showAnswer function since it won’t be needed after refactoring the code as required in // this question.**

**}**

1. Write a line of JavaScript code that sets the interval of function ticker( ) to 5 seconds.

The required line assuming that function ticker() is defined ( **setInterval( ticker, 5000)** )

Here is the full code:

A screenshot of a computer

Description automatically generated with medium confidence

Here is the output:

Graphical user interface, text, application, email

Description automatically generated

Here is the GitHub repo:

<https://github.com/AhmedAbdelRazak/RCC/blob/master/Chapter9/WrittenAssign_Q10/index.html>