Removing an element that is added to the screen after the 'Ready' or 'Load' event

In this article we're going to look at how we can hide an element that is loaded into CRM after the page 'Ready' or 'Load' event has fired; e.g. Date and Time field images.

In this article we're looking at two different methods we can use:

1. 'Mutation Observer'
2. 'Timeout Loop'

For modern browsers the 'Mutation Observer' is the best solution but if this fails, we can fall back on the 'Timeout Loop' method. In the 'Opportunity Detail Screen (ASP: OpportunityDetailBox)' 'Custom Context' script example below we're using the 'Timeout Loop' for IE and the 'Mutation Observer' for everything else.

```
<script>
crm.ready(function() {
  // Check that we are in 'Edit Mode'
  if ($("#oppo_opened").length == 1) {
    // Check the browser
    if (BrowserType == Browser.InternetExplorer) {
      // Jquery selector for the element to hide;
      //   1) find the underlying Span
      //   2) then look for the img element within the span
      var selector = "#_Dataoppo_opened > img[class='ui-datepicker-trigger']";
      // 'Timeout Loop'
      WaitForElementToDisplay(
        selector,
        function() {
          // Call back function for when the element has been found.
          $(selector).hide();
        },
        10,
        1000
      );
    }
    else {
      // 'Mutation Observer'
      WaitForElementToDisplay(
        "_Dataoppo_opened",
        "IMG",
        [{"name":"class", "value":"ui-datepicker-trigger"},
         {"name":"src", "value":"/crm/Themes/img/ergonomic/Buttons/calCalendar.gif"}],
        function() {
          // Call back function for when the element has been found.
          $("#_Dataoppo_opened > img[class='ui-datepicker-trigger']").hide();
        }
        );
    }
  }
});

// Javascript Enum equivalent for the Browser types that we're detecting
const Browser = Object.freeze({
  Other: 0,
  Chrome: 1,
  Chromium: 2,
  Edge: 3,
  Firefox: 4,
  InternetExplorer: 5,
  Opera: 6,
  Safari: 7
});
// Function to detect browser type
function BrowserType() {
  return (function (agent) {
    switch (true) {
      case agent.indexOf("edge") > -1:
        return Browser.Edge;
      case agent.indexOf("edg/") > -1:
        return Browser.Chromium;
      case agent.indexOf("opr") > -1 && !!window.opr:
        return Browser.Opera;
      case agent.indexOf("chrome") > -1 && !!window.chrome:
        return Browser.Chrome;
      case agent.indexOf("trident") > -1:
        return Browser.IE;
      case agent.indexOf("firefox") > -1:
        return Browser.Firefox;
      case agent.indexOf("safari") > -1:
        return Browser.Safari;
      default:
        return Browser.Other;
    }
  })(window.navigator.userAgent.toLowerCase());
}

// Timeout loop function
function WaitForElementToDisplay(selector, callback, checkFrequencyInMs, timeoutInMs) {
  // We grab our start time
  var startTimeInMs = Date.now();
  // Then we loop until we reach the timeout
  (function loopSearch() {
    if ($(selector).length > 0) {
      // If we find the element we can call the 'callback' function and exit the loop
      callback();
      return;
    }
    else {
      // Otherwise we keep looping
      setTimeout(function () {
        if (timeoutInMs && Date.now() - startTimeInMs > timeoutInMs)
          return;
        loopSearch();
      }, checkFrequencyInMs);
    }
  })();
}

// Mutation Observer function
function WaitForElementToDisplay(rootNode, nodeName, nodeAttributes, callback) {
  var MutationObserver =
    window.MutationObserver || window.WebKitMutationObserver || window.MozMutationObserver;
  // Create our observer with the function that is called when the DOM changes
  var observer = new MutationObserver(function (mutations) {
    // Iterate through all the changes
    mutations.forEach(function (mutation) {
      var blnFound = true;
      // Check to the Node Tag Name (IMG in this cas)
      if (mutation.target.nodeName != nodeName) {
        blnFound = false;
      }
      if (blnFound) {
        // Check using the attributes passed in to verify if we've found
        for (var i = 0; i < nodeAttributes.length; i++) {
          if (mutation.target.getAttribute(nodeAttributes[i].name) != nodeAttributes[i].value)
          {
            blnFound = false
            break;
          }
        }
      }
      // If we've found the element we're looking for we can stop observing and
      // call the 'callback' function
      if (blnFound) {
        observer.disconnect();
        callback();
      }
    });
  });
  // Start the observer N.B. we can specifiy the 'root' node that we want to observer
  observer.observe(document.getElementById(rootNode), {
    attributes: true,
    attributeOldValue: true,
    childList: true,
    subtree: true,
    characterData: true
  });
}
</script>
```