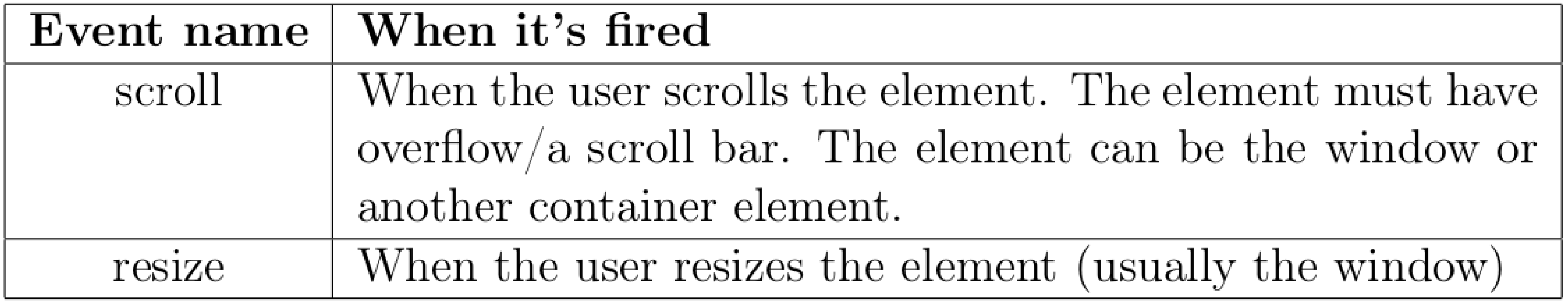
Events, Forms, and Animations

A simple event

Example 1 – Events make me feel bubbly

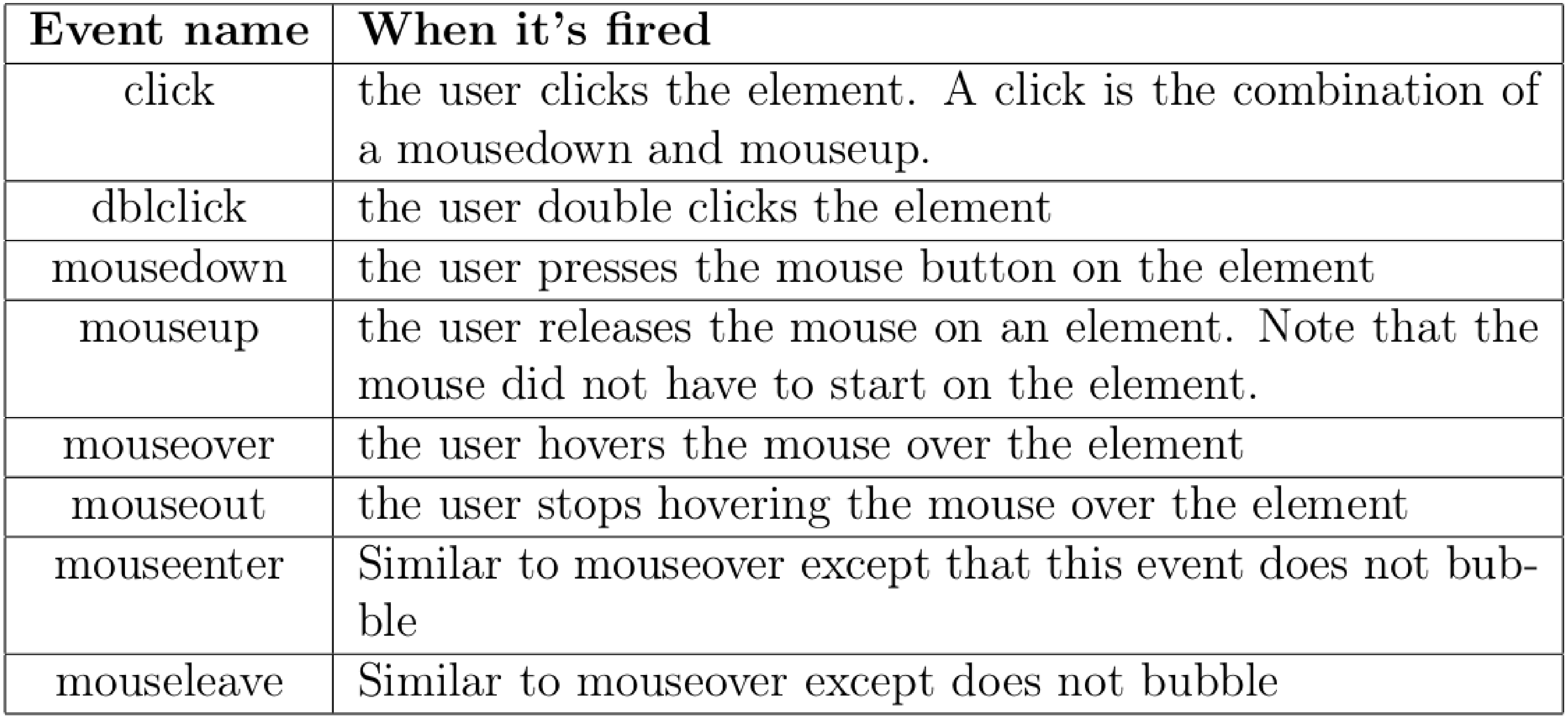
```
<html>
<!-- omitting head for brevity -->
<body>
  <form onclick='console.log("form")'>
    <input type='button' onclick='console.log("button")'>
  </form>
</body>
</html>

When the button is clicked, the console will display:
"button"
"form"
```

UIEvents



Event name	**When it's fired**
scroll	When the user scrolls the element. The element must have overflow/a scroll bar. The element can be the window or another container element.
resize	When the user resizes the element (usually the window)

MouseEvents



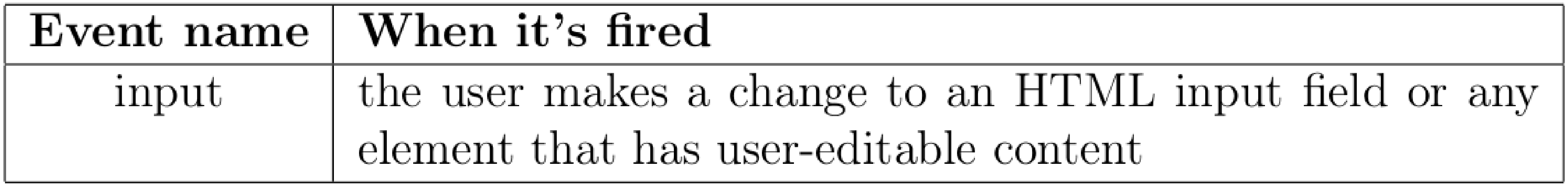
Event name	**When it's fired**
click	the user clicks the element. A click is the combination of a mousedown and mouseup.
dblclick	the user double clicks the element
mousedown	the user presses the mouse button on the element
mouseup	the user releases the mouse on an element. Note that the mouse did not have to start on the element.
mouseover	the user hovers the mouse over the element
mouseout	the user stops hovering the mouse over the element
mouseenter	Similar to mouseover except that this event does not bubble
mouseleave	Similar to mouseover except does not bubble

KeyboardEvents

Event name	**When it's fired**
keydown	the user presses a key on the keyboard
keyup	the user releases a key on the keyboard

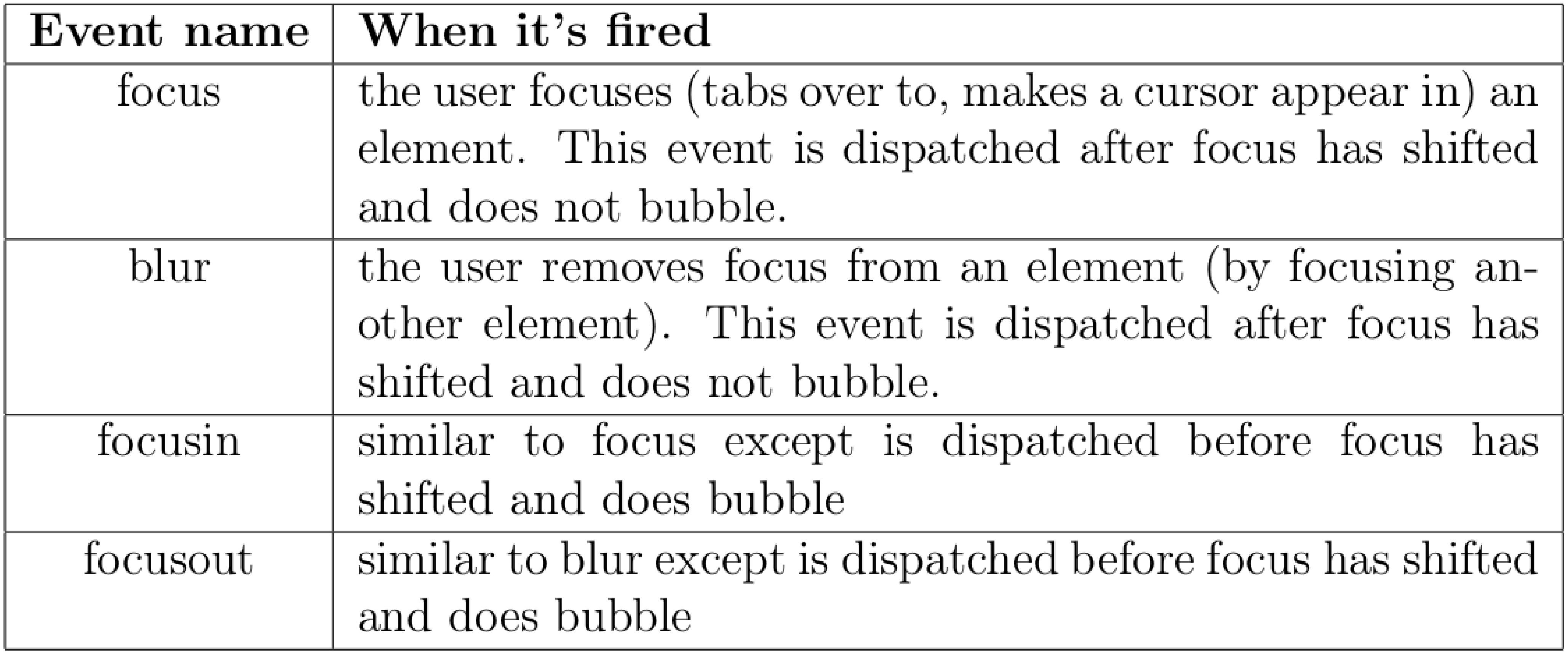
- `key` - a String representing the pressed key (only really useful for control/direction keys)
- `code` - the keycode for the pressed key. This will be the same for all keyboard mappings. The enter key has code 13.
- `ctrlKey`, `shiftKey`, `altKey`, `metaKey` - booleans that are set if a particular modifier key was pressed during the event

InputEvent



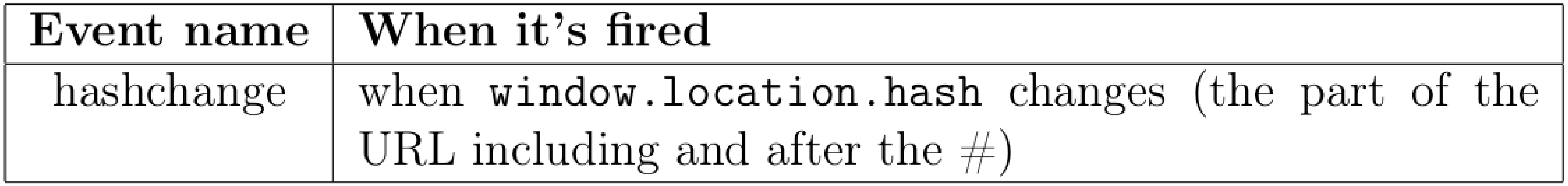
Event name	**When it's fired**
input	the user makes a change to an HTML input field or any element that has user-editable content

FocusEvent



Event name	**When it's fired**
focus	the user focuses (tabs over to, makes a cursor appear in) an element. This event is dispatched after focus has shifted and does not bubble.
blur	the user removes focus from an element (by focusing another element). This event is dispatched after focus has shifted and does not bubble.
focusin	similar to focus except is dispatched before focus has shifted and does bubble
focusout	similar to blur except is dispatched before focus has shifted and does bubble

HashChangeEvent



Event name	**When it's fired**
hashchange	when `window.location.hash` changes (the part of the URL including and after the #)

Registering Event Listeners

```
<tagname onevent="JavaScript code">
<!-- where event is the event for which to listen -->
```

```
/* DOM API */
element.addEventListener("event", handlerFunction);

/* jQuery */
$jQueryObject.on("event", handlerFunction);

/* where event is the event type, like click, or mousemove,
and handlerFunction is a function that accepts one argument,
the eventObject (can be anonymous) */
//this in the handlerFunction is set to the listening DOM node
```

Registering and Triggering Listeners

- `$jQueryObject.click([handler])` - when called with no arguments, "clicks" on the element or adds `handler` as a listener if present
- `$jQueryObject.focus([handler])` - focuses the element or adds the function `handler` as a focus listener
- see *Additional resources* for a link to more just like these

Example 2 – This example is not uneventful

```
//Creates a button that, when clicked, displays a popup
var aButton = document.createElement("input");
aButton.type = "button";
aButton.value = "Click me!";
aButton.addEventListener("click", function() {
  alert("Thank you for making a simple button very happy.");
});
document.body.appendChild(aButton);

var $aButton = $("<input type='button' value='Click me!'>");
$aButton.click(function () {
  alert("Thank you for making a simple button very happy.");
});
$aButton.appendTo(document.body);
```

Removing Event Listeners

```
/* DOM API */
element.removeEventListener("event", handlerFunction);

/* jQuery */
$jQueryObject.off("event", handlerFunction);

/* where event is the event type, like click, or mousemove,
and handlerFunction is a function that accepts one argument,
the eventObject (can be anonymous) */
```

Triggering Events and Listeners

```
/* DOM method */
//Left as an exercise to the reader

/* jQuery method */
//To call an element's event listeners
$jQueryObject.triggerHandler("event");

//To dispatch an event to the element
$jQueryObject.trigger("event");
//or
var $jQueryEvent = jQuery.Event("eventname", [properties]);
$jQueryObject.trigger($jQueryEvent);
```

Properties of the Event Object

```
//To stop event bubbling
eventObject.stopPropagation();

//To stop event bubbling and prevent other listeners on the
//same element from being triggered
eventObject.stopImmediatePropagation();
```

```
//To cancel an event/prevent its default action
eventObject.preventDefault();
```

A “fun” example

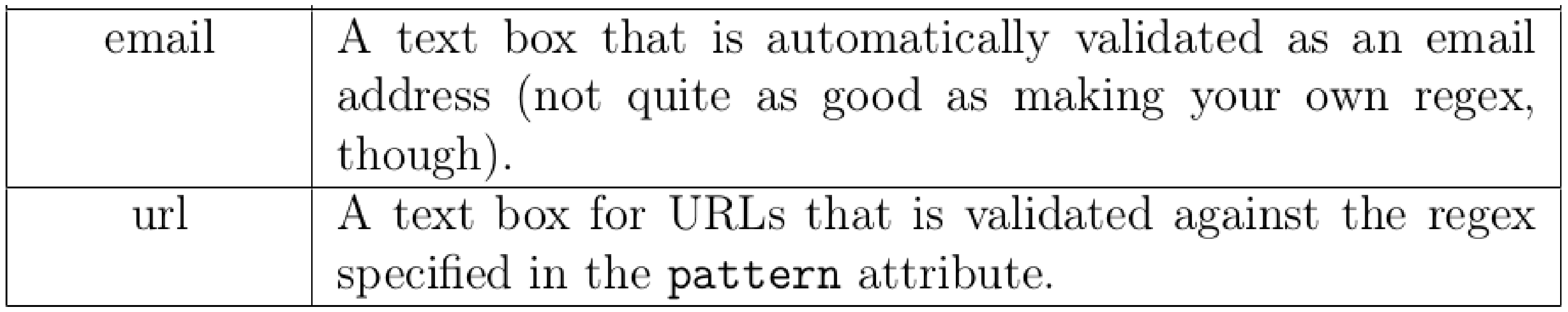
Example 3 – How to use JavaScript to troll your friends

```
//This is unlikely to work in a page that has its own listeners
//You'd have to remove those first
window.addEventListener("mousedown", function(event) {
  event.preventDefault();
  event.stopImmediatePropagation();
  //results in nothing on the page being left-clickable
});
```

Basic `input types`

Input type	**Description**
text	A text box. Not very exciting, I know, but its metaphorical bread and butter of HTML forms.
password	Like a text box except the entered characters are hidden. For example, in your browser they are likely displayed as dots.
checkbox	A checkbox
radio	A radio (multiple choice, single answer) button. Radio buttons with the same name are considered to be a group in which only one is selectable at a time.
submit	A button that has the text "Submit" and the default action of submitting its parent form.
reset	A button that has the text "Reset" and the default action of resetting its parent form's fields.
button	A button. Does not contain text and has no default action. These can be styled and given click handlers, though.
hidden	A hidden form field. You can use this as a way to store and non-user-editable data to a webpage on form submit.
file	A file select control. Allows the user to select files from the system for upload.

Validated `input types`



email	A text box that is automatically validated as an email address (not quite as good as making your own regex, though).
url	A text box for URLs that is validated against the regex specified in the `pattern` attribute.

The `value` of an `input`

```
//To get/set the value/user input of an input-type element
/* DOM API */
var value = element.value;         //getter
element.value = "someValue";       //setter

/* jQuery */
var value = $jQueryObject.val();  //getter
$jQueryObject.val("someValue");   //setter
```

Checking an `input` out

```
//To get the value of the checked element (radio or checkbox)
/* DOM API */
var checkables = document.getElementsByName("inputName");
for(var i = 0; i < checkables.length; i++) {
  if(checkables.item(i).checked) {
    console.log(checkables.item(i).value);
  }
}

/* jQuery */
var $checked = $("selector").filter(':checked');
var response = $checked.val();
```

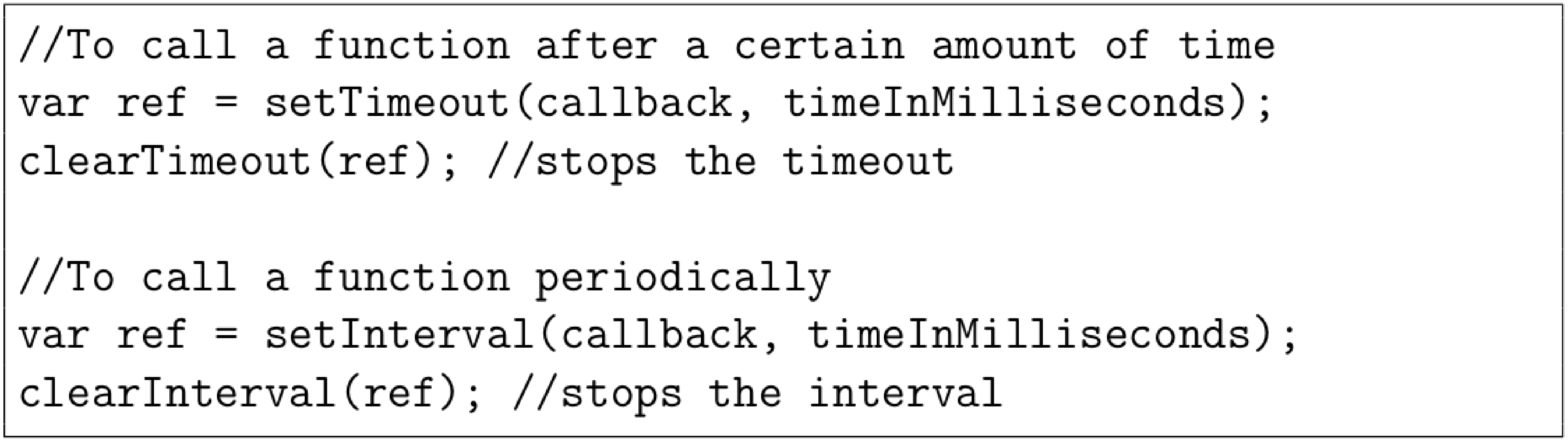
Example 4 – A formidable form

```
<html>
<!-- head and jQuery omitted for brevity -->
<body>
  <form onsubmit='submitForm()' action='javascript:void(0)'>
    <input type='text' placeholder='Username' id='username'>
    <br>
    <input type='password' placeholder='Password'
    id='password'>
    <br>
    <input type='checkbox' checked='true' id='keepLogged'>
    Keep me logged in.
    <br>
    <input type='submit' value='Submit'>
  </form>
  <script type='application/javascript'>
    function submitForm() {
      var username = $("#username").val();
      var password = $("#password").val();
      var stayLoggedIn = $('#keepLogged').attr("checked");
      doLogin(username, password, stayLoggedIn);

      alert("None shall pass.");
    }
  </script>
</body>
</html>
```

CSS3 Animations: `transition`

```
transition: css-prop duration [timing-function] [delay], ...;
/* timing-function and delay are optional parameters */

/* or use all to animate all CSS properties */
transition: all duration [timing-funciton] [delay];
```

Keeping Time



```
//To call a function after a certain amount of time
var ref = setTimeout(callback, timeInMilliseconds);
clearTimeout(ref); //stops the timeout

//To call a function periodically
var ref = setInterval(callback, timeInMilliseconds);
clearInterval(ref); //stops the interval
```

jQuery UI Effect

```
//With jQuery UI imported,
$jQueryObject.effect("bounce"); //Wow, that was easy
```

Example 5 – Not an accurate physical representation

```
<html>
<head>
  <style type='text/css'>
    .bouncingBall {
      transition: top 250ms linear;
      width: 100px; height: 100px;
      border-radius: 100px; /* makes a circle */
      position: absolute;
      top: 70%;
      background-color: red;
    }
  </style>
</head>
<body>
  <div class="bouncingBall"></div>
```

```
  <!-- jQuery import omitted for brevity -->
  <script type='application/javascript'>
    function bouncyAnimation(maxHeight, goingUp) {
      var $bouncy = $(".bouncingBall");
      maxHeight = maxHeight || $bouncy.css("top");
      if(goingUp) {
        $bouncy.css("top", maxHeight);
        setTimeout(function() {
          bouncyAnimation(maxHeight, false); }, 250);
      } else {
        var screenBottom = window.innerHeight;
        var elemHeight = $bouncy.height();
        $bouncy.css("top", screenBottom - elemHeight);
        setTimeout(function() {
          bouncyAnimation(maxHeight, true); }, 250);
      }
    }
    bouncyAnimation(); //make the magic happen
  </script>
</body>
</html>
```