DOM Events

Relevant Links

- Flanagan's book, Chapter 17
- Flanagan's book, Part IV (Event section)
- devdocs on DOM Events[1]
- MDN event reference[2]
- jQuery events section[3]

Notes

Events

Web Applications are interactive: The user needs to take an action, and the application needs to act and perform a task in response to that action. This is done through *Events*.

Event-based programming is the most standard of asynchronous behaviors: Our page is ready, and our application does nothing for a while, waiting for a user to interact with the page. When the user clicks a button or types something in, the application does something in response, then goes back to "sleep" until the next event.

There are a number of different types of events:

- Mouse Events
- Keyboard Events
- Load Events
- Touch Device Events
- and so on.

Here is a list of the most important event types/names:

click Triggered when the user clicks something
dblclick Triggered when the user double-clicks something
focus Triggered when an element gets "focus", either by being selected or by being activated via the keyboard.
change Triggered when an element loses focus, and had changes done to it while it had focus (e.g. someone typed something in an input box, then navigated out of the box).
hashchange The fragment portion of the URL has changed (This is the #... bit at the end.)

[1]http://devdocs.io/dom_events/
[2]https://developer.mozilla.org/en-US/docs/Web/Events
[3]http://api.jquery.com/category/events/

input The value of an element changes (e.g. when typing into a text field).
keydown A key is pressed down.
keyup A key is released.
load When a resource has finished loading.
mousedown Mouse button is pressed on an element.
mouseup Mouse button is released.
mouseenter / mouseleave The mouse has entered / left an element's area.
more mouse events There are a couple more mouse events. Look at documentation.
touch device events Similarly, a number of touch events are available.
copy / paste Events related to the clipboard.
reset / submit When a form resets / is submitted.

Furthermore, you can create your own custom events, and have different parts of your application use those events for communication. We will discuss this aspect of events later.

Registering Handlers

You register for events via a handler. Essentially, a function is attached to a specific element for a specific event name, and if that event occurs within that element, the function will get to run, being passed appropriate information about the event and the element on which it occured.

Bubbling of Events and default behavior

Event Bubbling, also called *propagation*, is the process by which events on an element are propagated up the DOM to all ancestors of that element. Registered handlers along the way are called, and they have the option to stop the further propagation of the event.

This is very useful, especially when we want to handle a variable list of items. If we want something to occur whenever a list item is clicked, we can instead attach a handle on the enclosing "ul" or "ol" element. This way one handler can handle all the list items, and we do not have to worry about registering / deregistering the handlers as the items change.

Events also tend to have a default behavior. For instance clicking on a link is meant to send the browser to a new location. Handlers also have the option of preventing that default behavior from occuring.

Events in jQuery

jQuery offers a variety of methods for registering for events. We will cover some aspects of it, but you can get a more complete view in jQuery's Event API[4].

[4] http://api.jquery.com/category/events/

The main jQuery function is `on`[5]:

```
// The second argument (selector) is optional
$("ul").on("click", "li.big", function(ev) {
  // The ev object contains info on the event
  // "this" is set to the DOM element on which the handler is attached
  console.log(ev);
});
// Bubbling, yey!
$(document).on("click", "li.big", function() { console.log("hi there!"); });
// Oops, no more bubbling
$("li").on("click", function(ev) {
  console.log("Stop it!");
  ev.stopPropagation();
});
```

[5] http://api.jquery.com/on/