Monitor Events

By Meggin Kearney
Meggin is a Tech Writer

By Flavio Copes
Flavio is a Full Stack Developer

The Chrome DevTools Command Line API offers various ways to observe and inspect event listeners. JavaScript plays a central role in interactive pages, and the browser provides you some useful tools to debug events and event handlers.

TL;DR

- Listen to events of a certain type using `monitorEvents()`.
- Use `unmonitorEvents()` to stop listening.
- Get listeners of a DOM element using `getEventListeners()`.
- Use the Event Listeners Inspector panel to get information on event listeners.

Monitor events

The monitorEvents() method instructs the DevTools to log information on the specified targets.

The first parameter is the object to monitor. All events return if the second parameter is not provided. To specify the events to listen to, pass either a string or an array of strings as the second parameter.

Listen to click events on the body of the page:

```
monitorEvents(document.body, "click");
```

If the monitored event is a supported *event type* that the DevTools maps to a set of standard event names, then the method listens to the events for that type.

The Command Line API has a full mapping of *event types* to the events they cover.

To stop monitoring events, call the `unmonitorEvents()` method and give it the object to stop monitoring.

Stop listening to events on the `body` object:

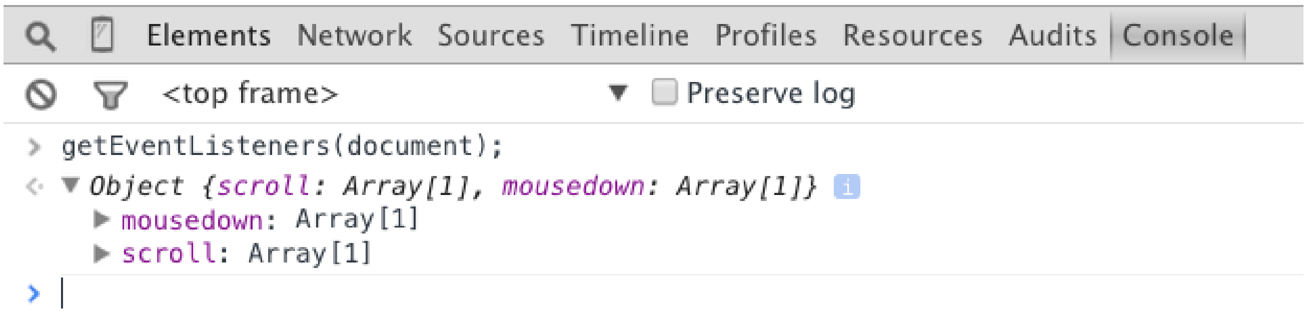
```
unmonitorEvents(document.body);
```

View event listeners registered on objects

The getEventListeners() API returns the event listeners registered on the specified object.

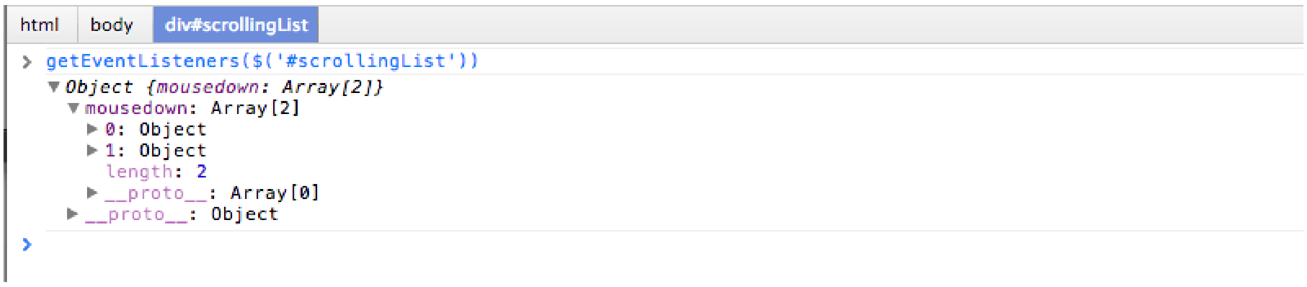
The return value is an object that contains an array for each registered event type (`click` or **keydown**, for example). The members of each array are objects that describe the listener registered for each type. For example, the following code lists all the event listeners registered on the document object:

```
getEventListeners(document);
```



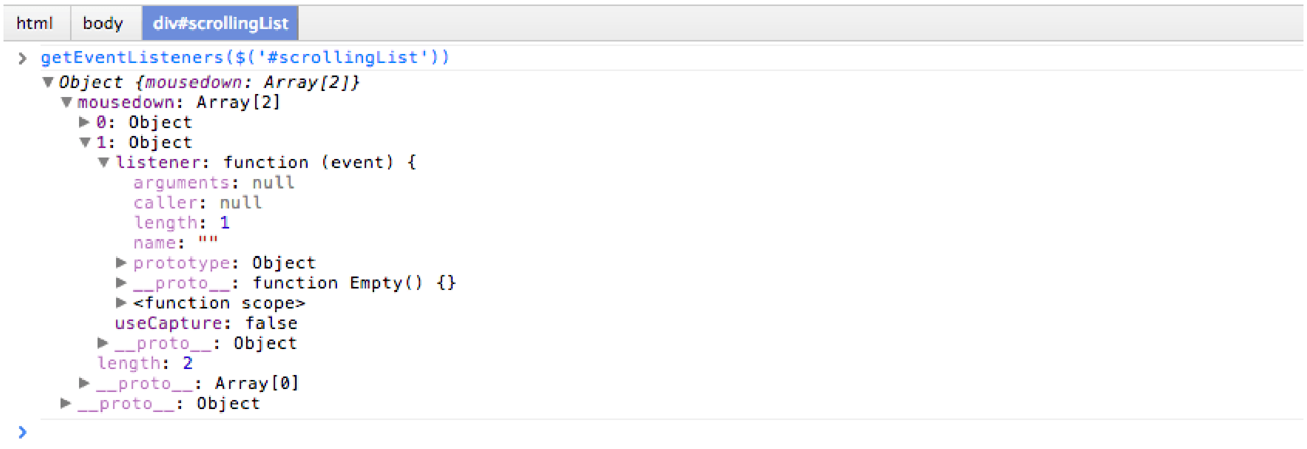
```
> getEventListeners(document);
<- ▼ Object {scroll: Array[1], mousedown: Array[1]}
    ▶ mousedown: Array[1]
    ▶ scroll: Array[1]
>
```

If more than one listener is registered on the specified object, then the array contains a member for each listener. In the following example, there are two event listeners registered on the #scrollingList element for the **mousedown** event:



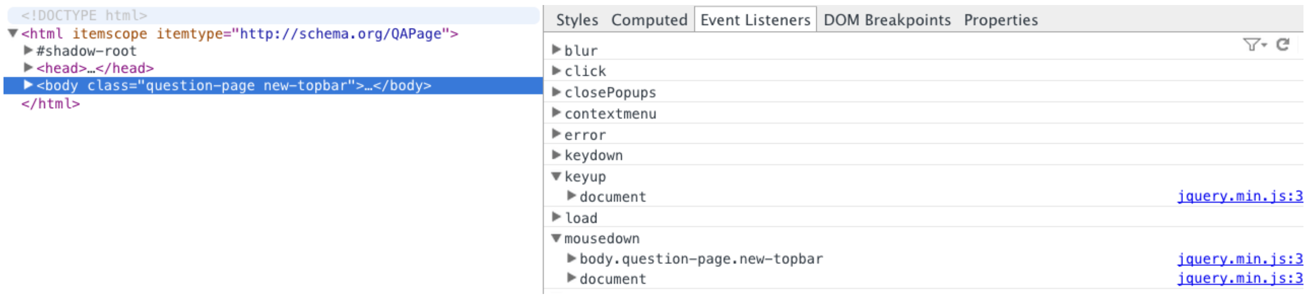
```
> getEventListeners($('#scrollingList'))
  ▼ Object {mousedown: Array[2]}
    ▼ mousedown: Array[2]
      ▶ 0: Object
      ▶ 1: Object
        length: 2
      ▶ __proto__: Array[0]
    ▶ __proto__: Object
>
```

Further expand each of these objects to explore their properties:

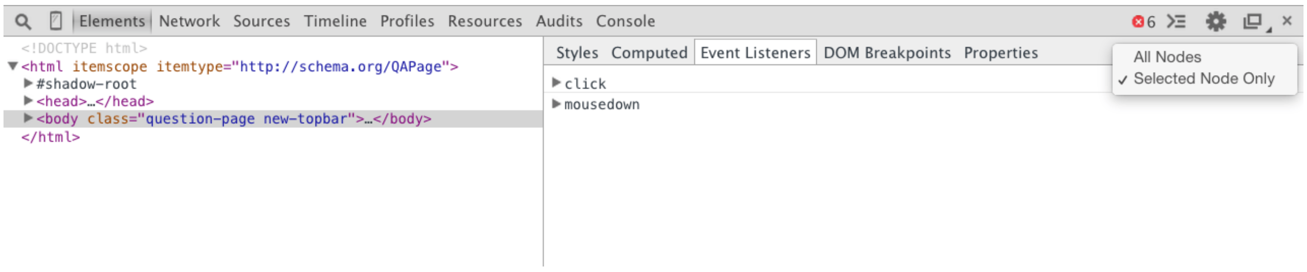

```
html   body   div#scrollingList
> getEventListeners($('#scrollingList'))
  ▼Object {mousedown: Array[2]}
    ▼mousedown: Array[2]
      ▶0: Object
      ▼1: Object
        ▼listener: function (event) {
            arguments: null
            caller: null
            length: 1
            name: ""
          ▶prototype: Object
          ▶__proto__: function Empty() {}
          ▶<function scope>
          useCapture: false
        ▶__proto__: Object
        length: 2
      ▶__proto__: Array[0]
    ▶__proto__: Object
>
```

View event listeners registered on DOM elements

By default, the *Event Listeners* panel in the Elements Inspector shows all the events attached to a page:

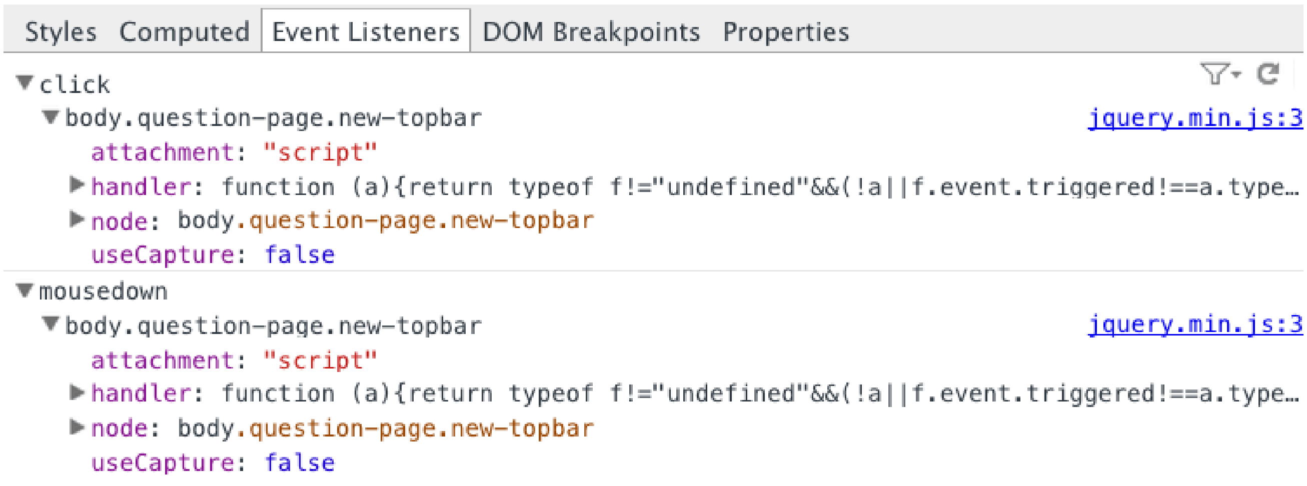

```
<!DOCTYPE html>
▼<html itemscope itemtype="http://schema.org/QAPage">
  ▶#shadow-root
  ▶<head>…</head>
  ▶<body class="question-page new-topbar">…</body>
  </html>
```
```
Styles  Computed  Event Listeners  DOM Breakpoints  Properties
▶blur
▶click
▶closePopups
▶contextmenu
▶error
▶keydown
▼keyup
  ▶document                                          jquery.min.js:3
▶load
▼mousedown
  ▶body.question-page.new-topbar                     jquery.min.js:3
  ▶document                                          jquery.min.js:3
```

The filter limits the events just to the selected node:



```
<!DOCTYPE html>
▼<html itemscope itemtype="http://schema.org/QAPage">
  ▶#shadow-root
  ▶<head>…</head>
  ▶<body class="question-page new-topbar">…</body>
  </html>
```
```
Styles  Computed  Event Listeners  DOM Breakpoints  Properties
▶click
▶mousedown
```
All Nodes [✓ Selected Node Only]

By expanding the object, the panel shows the event listener details. In this example, the page has two event listeners attached via jQuery:



```
▼click
  ▼body.question-page.new-topbar                                              jquery.min.js:3
     attachment: "script"
    ▶handler: function (a){return typeof f!="undefined"&&(!a||f.event.triggered!==a.type…
    ▶node: body.question-page.new-topbar
     useCapture: false
▼mousedown
  ▼body.question-page.new-topbar                                              jquery.min.js:3
     attachment: "script"
    ▶handler: function (a){return typeof f!="undefined"&&(!a||f.event.triggered!==a.type…
    ▶node: body.question-page.new-topbar
     useCapture: false
```

Except as otherwise noted, the content of this page is licensed under the Creative Commons Attribution 3.0 License, and code samples are licensed under the Apache 2.0 License. For details, see our Site Policies. Java is a registered trademark of Oracle and/or its affiliates.

Last updated July 2, 2018.