Timer Patterns

Aaron Powell

http://readify.net | http://www.aaron-powell.com

pluralsight
hardcore developer training

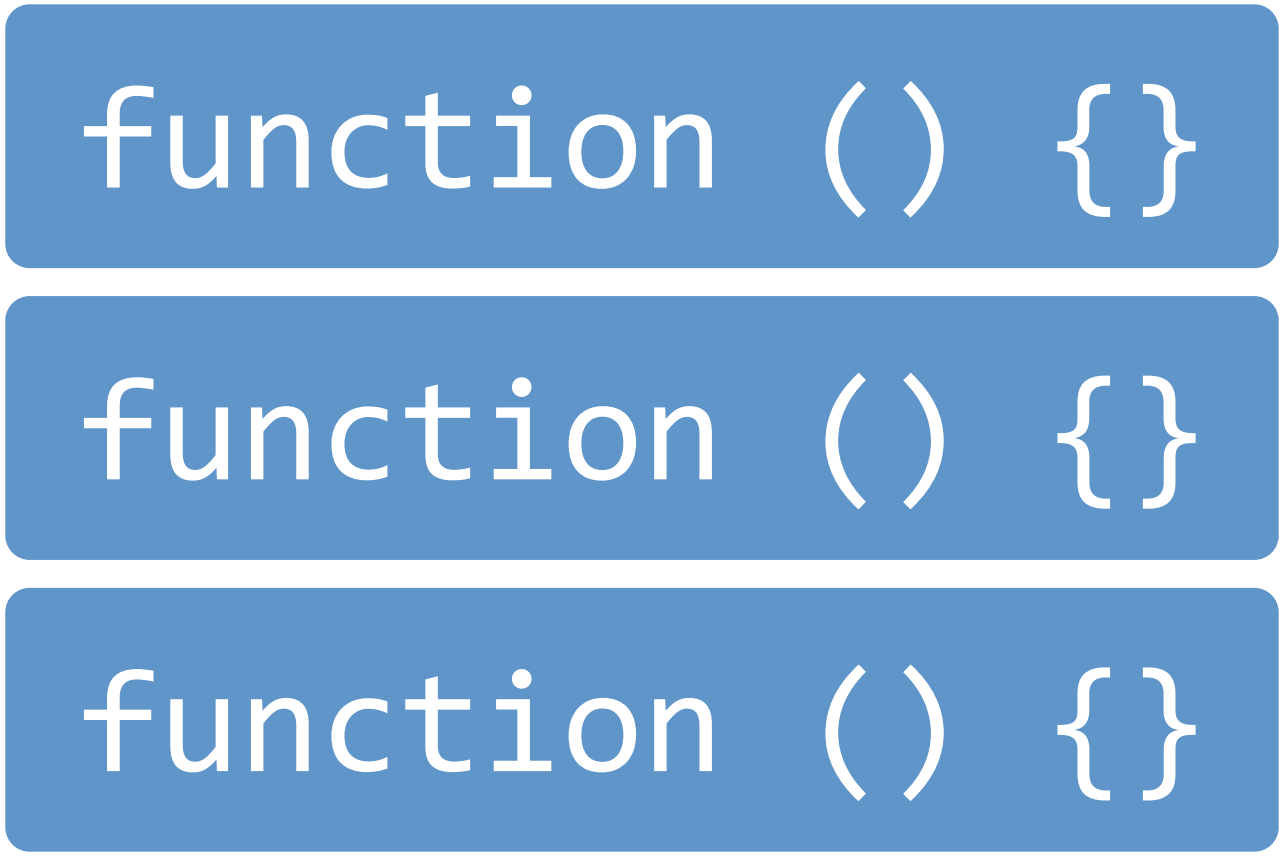
JavaScript timers

- **Two kinds of timers in JavaScript**
  - setTimeout
  - setInterval
- **Both take two arguments, a function to invoke and a delay period**
- **setTimeout will execute the function once**
- **setInterval will execute continuously with the specified delay between each execution**
- **Delays of <4ms will be bumped to 4ms**
- **Timers wont start until the outer most function is finished**

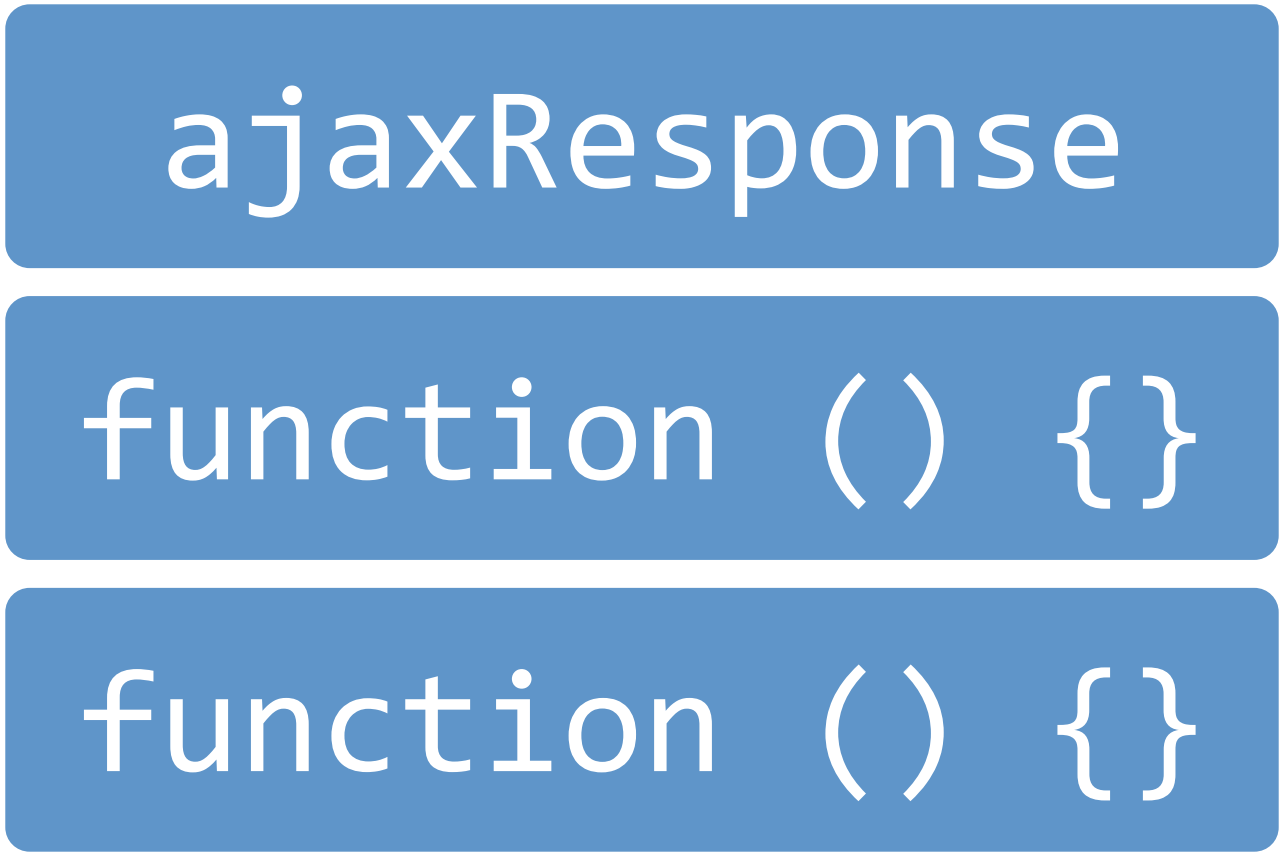
```
var fn = function () {
   setTimeout(function () { … }, 0);  1
   setTimeout(function () { … }, 0);  2
   setTimeout(function () { … }, 0);  3
};

fn();
```

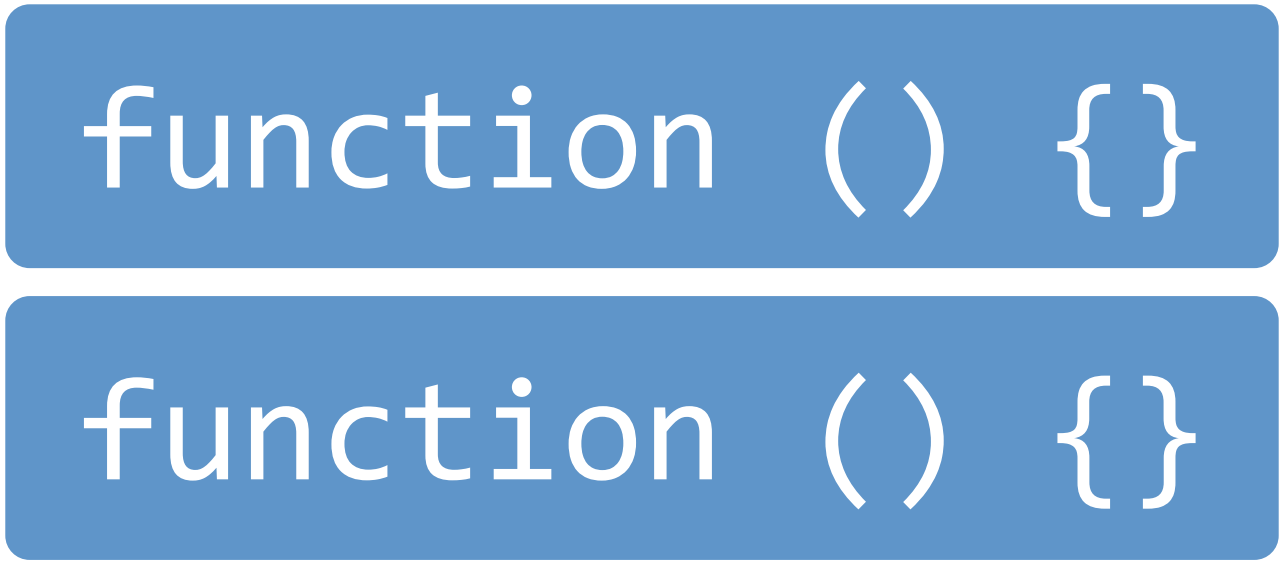
Visualising timers



Visualising timers



Visualising timers



Asynchronous Execution Pattern

Overview

- **Browsers are typically single threaded**
  - Either updating the UI or executing JavaScript
- **Long-running JavaScript blocks the UI**
  - Browser is unresponsive
- **Splitting long-running code over setTimeout blocks releases the thread**
  - While processing a loop limit the scope to a small time window
  - Ensure that there is enough of a gap between timeouts restarting

Demo

Asynchronous Execution Pattern

Recap

- **Using a setTimeout of <4ms will become 4ms**
- **Long running functions will block the UI**
  - Split them over setTimeout calls
- **Using too short a timeout wont release to the UI either**
- **Pattern most useful on low powered devices**

Recursive setTimeout Pattern

Overview

- **Periodically running a piece of functionality, related to a duration**
  - Most commonly used to query a data source
- **Sounds like setInterval**
- **setInterval has a problem though…**
  - JavaScript is asynchronous
  - Once an interval function finishes it is added back to the timer queue
  - But what if that function was waiting for an AJAX response?
    - Function execution can get out of order

Using setInterval

```
setInterval(function () {
   $.get('/foo', function (result) {
       //do something with the results
   });
}, 1000);
```

Using setTimeout

```
setTimeout(function getFoo() {
   $.get('/foo', function (result) {
       //do something with the results
       setTimeout(getFoo, 1000);
   });
}, 1000);
```

Demo

Implementing timeout patterns

Recap

- **setInterval ordering is unpredictable across browsers**
  - Recursively invoking setTimeout can ensure order of execution