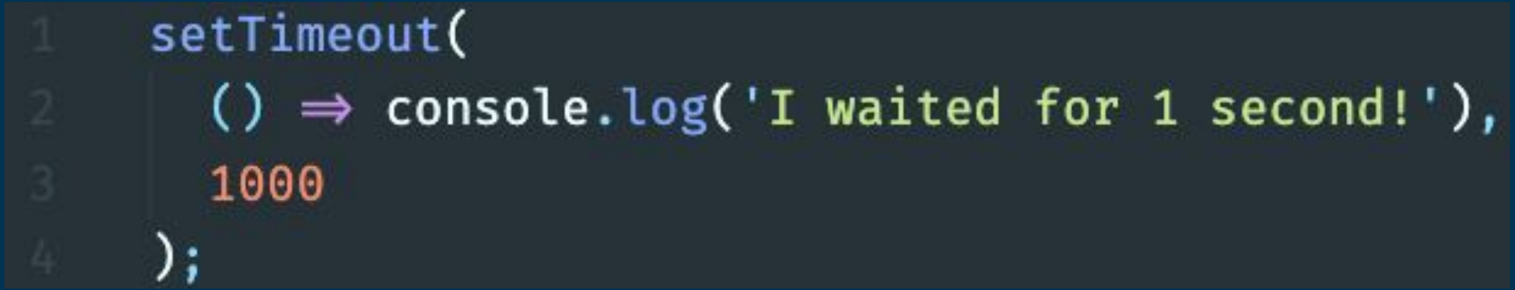
JS Concurrency

Callbacks

How do we schedule event handlers?



```
setTimeout(
  () => console.log('I waited for 1 second!'),
  1000
);
```

- We can schedule an event handler by calling an API function, like setTimeout above, that is supplied by the JavaScript engine.
- By passing it a function of our own, we can tell the engine to execute that function when the event managed by the API is triggered.
- Here, we tell Node (or whatever our engine is) to schedule a console log for 1000 milliseconds after the setTimeout call completes.

Callbacks

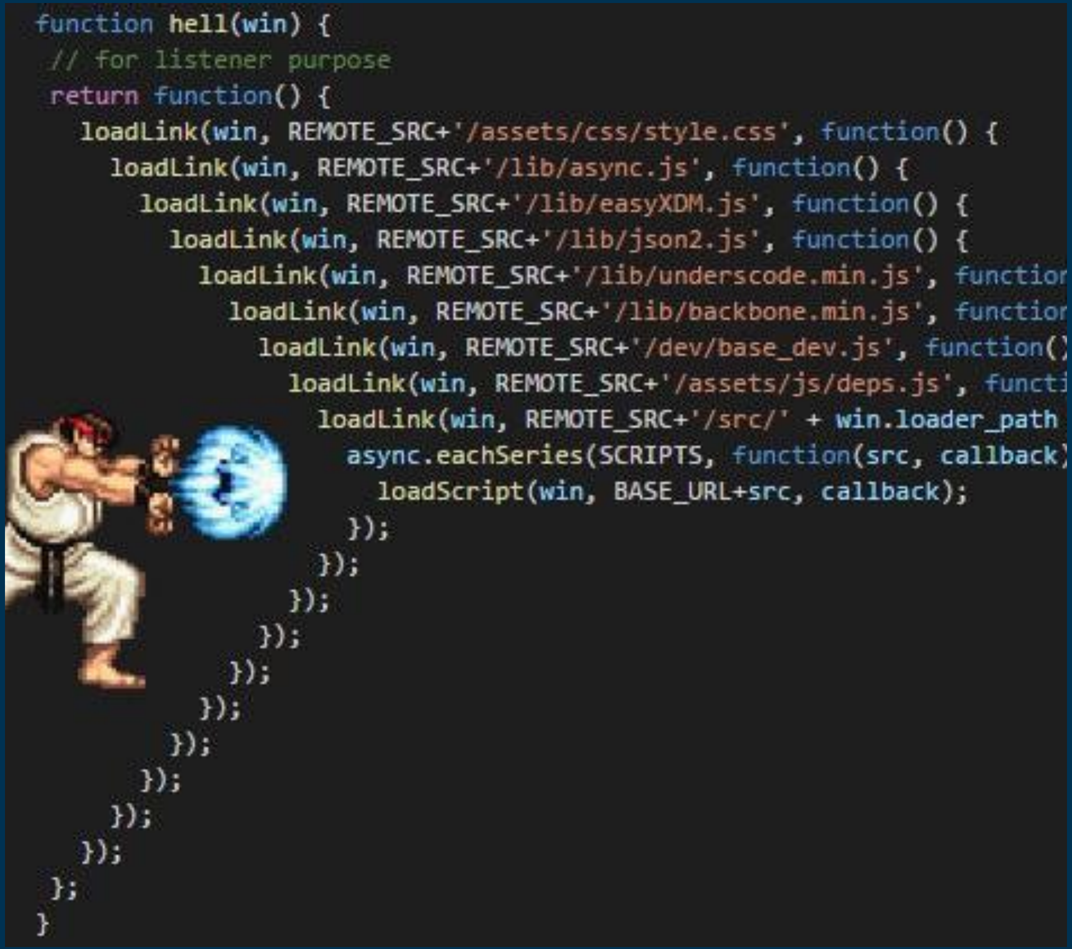
A Function by Any Other Name

What are Callbacks?

- A callback is just a function that we pass as an argument to another function.
- It’s called a “callback” because we expect the *receiving* function to call *our* function when the receiving function has completed its task.
- This is a style of control flow called continuation-passing-style, and it’s an alternative to returning values from functions.
- It’s also perfect for scheduling asynchronous event handlers!

Callback Hell

- Problem: Callback syntax is hard to chain
- Solution: Promises!



```
function hell(win) {
 // for listener purpose
 return function() {
   loadLink(win, REMOTE_SRC+'/assets/css/style.css', function() {
     loadLink(win, REMOTE_SRC+'/lib/async.js', function() {
       loadLink(win, REMOTE_SRC+'/lib/easyXDM.js', function() {
         loadLink(win, REMOTE_SRC+'/lib/json2.js', function() {
           loadLink(win, REMOTE_SRC+'/lib/underscode.min.js', functio
             loadLink(win, REMOTE_SRC+'/lib/backbone.min.js', functio
               loadLink(win, REMOTE_SRC+'/dev/base_dev.js', function(
                 loadLink(win, REMOTE_SRC+'/assets/js/deps.js', functi
                   loadLink(win, REMOTE_SRC+'/src/' + win.loader_path
                     async.eachSeries(SCRIPTS, function(src, callback
                       loadScript(win, BASE_URL+src, callback);
                     });
                   });
                 });
               });
             });
           });
         });
       });
     });
   });
 };
}
```