Executing Async Tasks Serially With Array#Reduce

by Can Ho MVB · Aug. 21, 16 · Web Dev Zone

Suppose that we’re assigned the task of writing a migration tool for a database with the following requirements:

- The tool will read a list of SQL scripts and execute them serially one after another.
- Each script will run once the previous script has completed.
- If any script execution fails, no more scripts will execute.

This can be done by using a library such as **async**#reduce (or **async**#series)

```
async.reduce(files, Promise.resolve(), function(prevPromise, file, callback){
    prevPromise.then(function(){
        return readFileAsync(file, {encoding: 'utf-8'});
    }).then(function(query){
        return execDbAsync(query);
    }).then(function(data){
        callback(data);
    });
}, function(err, result){

});
```

Without using an additional library, this executing **async tasks serially** problem can be solved by using the Array#reduce method:

```
files.reduce(function(prevPromise, curr, curIdx, arr){
    return prevPromise.then(function(){
        return readFileAsync(curr, {encoding: 'utf-8'});
    })
    .then(function(query){
        return execDbAsync(query);
    });
}, Promise.resolve());
```

The above code will return a Promise which:

- gets resolved when all chaining promises get resolved.

- gets rejected when any of chaining promises get rejected.

Compared to using **async**#reduce, using the built-in Array#reduce method has some benefits:

- No additional library needed
- No callback
- Less code as we don't need to explicitly call "callback (data)" to notify a task completion

Execution order looks like bellow:

read (script 1)—> exec (script 1)—> read (script2)—> exec (script 2)—>...–> read (script n)—> exec (script n)

Topics: ARRAY, PROBLEM, LIBRARY, MIGRATION, TASKS, TOOL

Hacking Javascript Games to Improve Your Testing

by Alan Richardson MVB · Sep 15, 16 · Web Dev Zone

For the last six months or so, I have, on and off, been learning to code JavaScript. That process has mainly involved:

- Writing some simple games.
- Hacking other games to see how they were written.

I’m going back to the good old days when we could ‘break’ the ZX Spectrum game and view the source, or disassemble the games on the Atari ST and hook/hack them through debuggers and monitors.

To do all of that now, JavaScript and the Browser Dev tools are perfect companions.

Learn How to Hack Games

I watched Philip Hödtke's talk from JS Unconf 2016 yesterday " Hacking Games and Why You Should Do It." and added a few new techniques into my ‘cheating/hacking’ repertoire.

I encourage you to watch Philip’s talk.

After watching Philip at work I realized:

- I haven’t been using the find functionality in the source part of the browser dev tools.
- I had an over-reliance on adding breakpoints and don’t need to.
- I had not been using `setInterval` from the console.

After watching I went off and tried the cookie clicker game that Philip demonstrates.

But I Test Things, Why Should I Care?

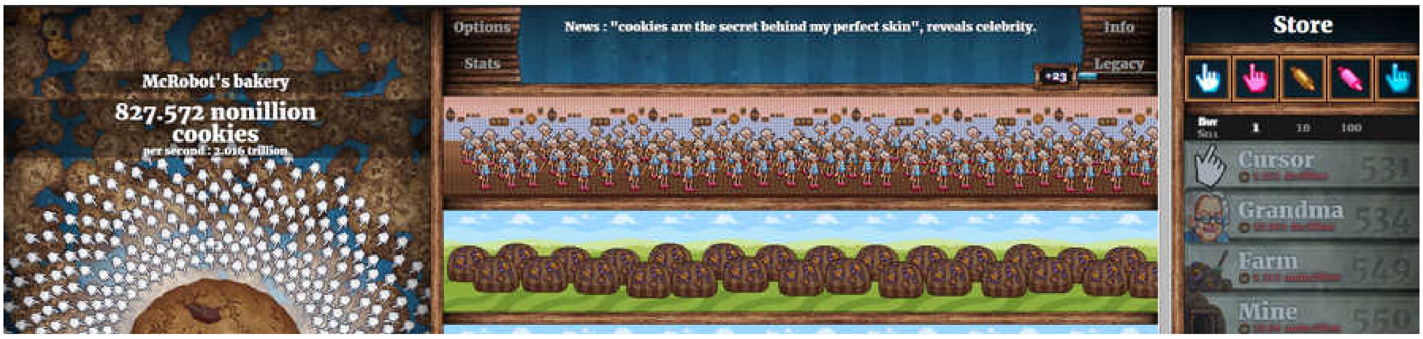
Well:

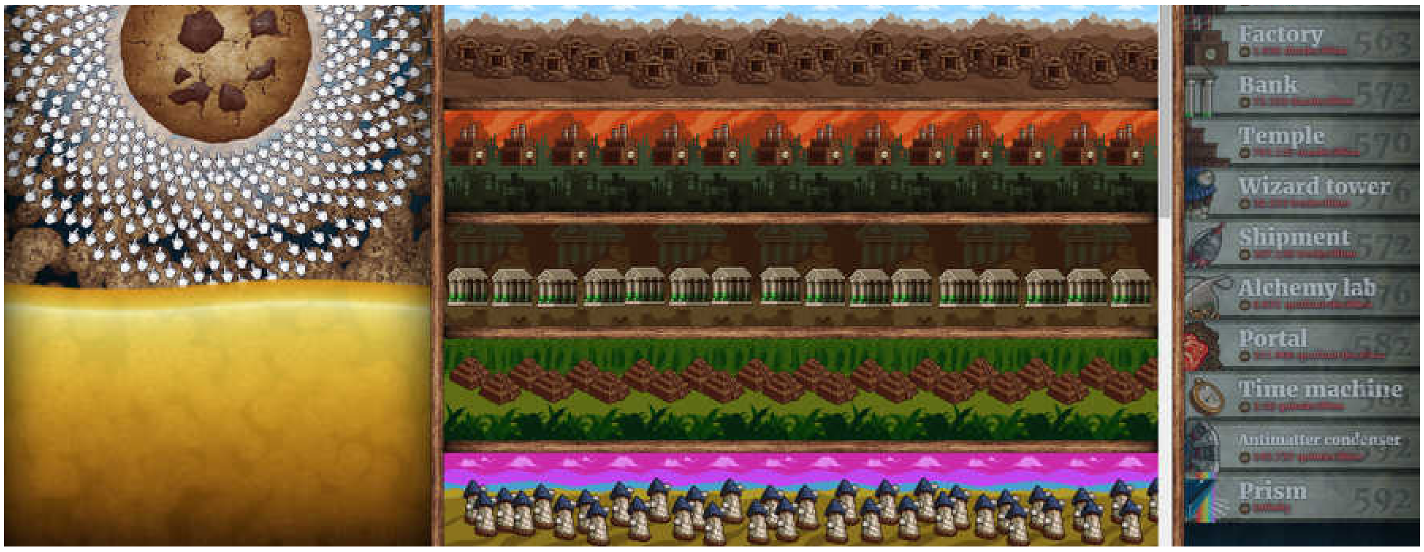
- Imagine that instead of having to use an automated tool to put the application into a certain state, or having to manually click around a do a lot of work.
- Imagine that you could just write a few lines of code into the browser console and automate there.
- Imagine that you could write a single line of code that would execute every second and ‘do stuff’ like click on a link, or close a dialog, or <insert something in here that you regularly have to do when you use an app>.

I describe an example later in this post of a one-line automated ‘bot’ that could play a game better than a human (me).

Back to Cookie Clicker

And I did quite well.


News : "cookies are the secret behind my perfect skin", reveals celebrity.



I used a slightly different approach than Philip used and found that `for` loops worked better for me than `setInterval` when manipulating as I explored, in this game.

Go Beyond the for Loop

I went off to find games where `setInterval` was a better fit for me.

- First The World's Biggest Pac-man where `setInterval` was a great way to provide infinite lives, restoring a new life each time one was lost.

I had a few false starts with other games that looked promising but which either:

- Had such obfuscated code I could make no headway.
- Contained game crashing bugs.

I tried some typing games but encountered the "how do I trigger keyboard events" stumbling block.

Automating ZType from the Console

And then I went back to that old favourite zty.pe.

I've played zty.pe before. I really like it.

> **I suspect I might end up playing z-type quite a lot https://t.co/Un6YG2SrKx just to improve my typing you understand.**
> **— Alan Richardson (@eviltester) July 21, 2016**

- I periodically still play "Typing of the Dead" on the Dreamcast. Yup. I have a Dreamcast with a keyboard.
- I periodically still play "R-type" - primarily from "R-Types" on the Playstation One (Yup, I still have a PS1).

[illegible]

Given both of the above. I really like ZType even though I'm not very good at it.

If You Can't Trigger Keyboard Events, How Could You Play ZType?

Yes, I know what you're thinking:

> **"But if I encountered the 'how do I trigger keyboard events' stumbling block, how do I automate zty.pe, which is all about keyboard events."**

Well, zty.pe is pretty well-designed. It's almost as though it was built to be tested and automated.

- rather than add the 'shoot' functionality in the keyboard event hook, so that the only way to trigger it, is by issuing a keyboard event.
- the keyboard event calls a method, so the method can be called outside the keyboard event

If this was an app you were testing and automating, it means that you can automate most of the app by calling the 'shoot' method. And leave the risk of 'do keyboard events fire on this platform' to a much reduced test scope with a different test approach.

Human vs. Bot

I only managed to reach level 15, but my automated 'bot' reached level 93. You can see the results of my automating here on youtube.

I'm not going to explain exactly what I did since that would ruin the fun. And the video doesn't show you how either.

Tips on Hacking JavaScript Games

But I'll give some additional tips, some you might see in action in Philip's video:

- Pretty print the source.
- Use 'find' to search for classes and variable you find in the source.
- If you type something into the console and it comes back with 'function' then you need to find where the class is instantiated in the code.
- Use 'find' to search for 'new' instantiations of the classes.
- You can do a lot of exploration and manipulation with 'for' loops.
- For bots, you'll need to use `setInterval`.
- I assign the result of `setInterval` to a bot e.g. `ztypebot = setInterval(...)` so that I can shut the bot down later with a `clearInterval(ztypebot)`.
- Find some 'quick hacks' that you can use early in your investigation to give you more room to find a better hack e.g. when automating ztype, I started with a bot that had infinite emps and triggered an emp every 2 seconds, and when that was working it gave me time to experiment with the objects and source to figure out how to make a bot that could shoot properly.
- Keep notes as you go about what you tried, and how you found the objects.
- Sometimes I start by working through the code and look for hints as Philip demonstrates.
- Sometimes I start by looking to see what code is triggered by the Event Listeners in the browser dev tools.
- Sometimes I breakpoint code that I think is interesting.
- They key (and Philip demonstrates this well) is to find the big 'namespace' type objects as early as possible.

And do make sure that you don't submit any high scores after 'cheating' or 'hacking' in this way, it is most annoying to other people. A proxy tool can help avoid you accidentally sending a high score by blocking any high score submissions e.g. Fiddler's auto-responder works well for this.

Testing Summary

When you hack/cheat at JavaScript games in this way, you are developing skills that will transfer to your testing:

- Understand JavaScript
- Learn how to use the browser dev tools
- Interacting with a running application
- Exploring the internal object state of an application

- Putting a running application into a specific state for testing automatically
- Writing very small amounts of code to automate an application state

If you find any good games to play with, let me know.

Topics: GAME, AUTOMATED, TEST, TOOLS, APPROACH, JAVASCRIPT, QUALITY ASSURANCE, HACK