Welcome to Issue 141 of Phaser World

**THIS WEEK...**

- **EXTREME BIKERS**
- **PHASER 3 SHADERS TUTORIAL**
- **LEGEND OF THE LOOTER**
- **GETTING STARTED PART 1**
I've been working hard on the Phaser documentation this week, which you can read about in the Dev Log. So it was a nice break to get to play some Phaser games for the newsletter, and what great games they were! From the sheer insanity of Monster Clean Up, the addictiveness of Legend of the Looter and the bullet hell that Pluto Attacks descends in to. It's awesome to see all of your collective creativity unleashed :) 

I'm also on the hunt for help translating the Phaser guides. If you're able to spare a bit of time to translate one of our guides from English then I'd love to hear from you! More details in the Dev Log. As usual, be sure to read this newsletter on the web so you don't miss anything.

Got a game or tutorial you'd like featured? Simply reply to this email, or message me on Slack, Discord or Twitter. Until the next issue, keep on coding!

**Only 20 Phaser Sticker Packs left**

I'm down to my final 20 packs of Phaser stickers! I'm going to take a rest from posting out stickers after this, so get them fast if you want to secure them. All orders will get a free Phaser pin-badge and a new transparent sticker included as well.

Click here to order a sticker pack.
The Latest Games

Game of the Week
Extreme Bikers
Hop on your bike, ride and jump. Execute stunts perfectly, avoid obstacles and reach the end safely in this manic extreme racing challenge.

Staff Pick
Legend of the Looter
The Legend of the Looter and the Incredible Backpack that never runs out of space! A highly playable fast-paced dungeon treasure crawl.

Pluto Attacks
Blast down the infinite waves of incoming drones in this superbly fast-paced shooter. It starts out peaceful, but ends anything but.
Monster Clean Up!
The life of a janitor is never easy. But it's especially hard when there are 99 floors of monster and rat infested hell to try and clear away!

Jump with Justin
Take flight with Justin Beaver! Control his ascent as he soars into the sky, collecting as many coins as he goes.

What's New?
Phaser CE v2.12.0 Released
The latest version of Phaser CE is now available with updates to the Creature runtime, audio fixes and camera features.
Build a Space Shooter in Phaser 3
A comprehensive 5 part tutorial on creating a shoot-em-up in Phaser 3.

Phaser 3 Shaders Tutorial
Why shaders are cool and how to use them in Phaser 3.

Magick Game Tutorial
A new Phaser 3 version of this tutorial about creating a game based on the iOS game Magick.

Getting Started with Phaser Part 1
A new beginners tutorial series of animated videos on getting started with Phaser 3.

FullStack NYC Phaser Session
The first FullStack conference in NYC features a game creation session with
Phaser.

Phaser 3 Game Development Course
A complete Phaser 3 and JavaScript Game Development package. 9 courses, 119 lessons and over 15 hours of video content. Learn to code and create a huge portfolio of cross platform games.

Help support Phaser

Because Phaser is an open source project, we cannot charge for it in the same way as traditional retail software. What's more, we don't ever want to. After all, it's built on, and was born from, open web standards. The core framework will always be free, even if you use it commercially.

You may not realize it, but because of this, we rely 100% on community backing to fund development.

Your support helps secure a constant cycle of updates, fixes, new features and planning for the future. There are other benefits to backing Phaser, too:
I use Patreon to manage the backing and **you can support Phaser from $1 per month**. The amount you pledge is entirely up to you and can be changed as often as you like.

**Please help support Phaser on Patreon**

Thank you to these awesome **Phaser Patrons** who joined us recently:

Carl Lindahl  
Chan Naing  
Chris McQueen  
David Wilcox  
Jeff Hube
Welcome to Dev Log 141. I'm going to warn you up front, this log is a bit dull :) Because I've spent the week working like mad on the documentation, which despite being vital, doesn't make for very interesting reading in a dev log.

Documentation is important for two reasons. First, and primarily, it's the thing a lot of you turn to when looking for answers. I know for a fact that a number of you go to the source code first and will manually look at a function to see what it does. But to do that, you need a good understanding of the layout of the API, and learning that takes time. So the majority of you, quite rightly, go to the API docs.

I've been working tirelessly this year on improving the Phaser 3 docs. From running the Doc Jam, to the days spent refactoring the events, to enhancing them with every release. Not a single new feature has been added to the API that isn't completely documented and it's been this way for many months now. That doesn't mean there aren't still areas that need completing - and the overall general way you browse the docs still leaves a huge amount to be desired.

The cause of this is two-fold. First, the actual docs themselves (i.e. the JSDoc blocks within the source code) need improving and expanding. There are still some areas left that don't have documentation at all, although these get less and less every month. Secondly, the HTML you browse when you visit the docs site is the direct output of a JSDoc template using the Cerulean theme. And honestly, it's far from great.
This is no fault of the template. After all, just like Phaser, it can't be everything to everyone. The overall layout, though, is just wrong. Making your own JSDoc templates is an arcane art. They're a bunch of static files that make use of a custom templating system, with HTML injected into JS injected into html. It's quite something to behold, let alone work with. The bigger issue, though, is that it takes a significant amount of time to actually generate the documentation. It will build the docs for the entire API, every single time, which generates hundreds and hundreds of files. 65MB worth each time in fact, taking several minutes. Which is less than ideal if you're making a quick tweak and want to test it to see if it looks right. Again, this isn't the fault of JSDoc. It is designed to generate static doc files from your source, that is its whole purpose. But it's not what Phaser needs.

Alert long-time readers may remember that back in Dev Log 100 I wrote about the work I was doing on the docs. At the time I was mostly talking about the app I built called Percy, which helped me generate all the docblocks the code required. However, I also alluded to the fact that I wanted to release the Phaser docs as a SQLite file and build a front-end around that. I spent a couple of days work on this at the time, but it never got any further because I was too wrapped up in actually getting Phaser itself working and the docs were written. Fast forward to this week, though, and that is where my work has been focused.

**Types Ahoy**

The first stage was sorting out all of the types. JSDoc offers a feature called a `@typedef` which allows you to create a new custom type that you then reference in the JSDocs in your code. I had been using these already, but they weren't properly namespaced, the location of the defs was inconsistent and the use of them was patchy. So the week began by addressing this.

For all areas that use them I created `typedefs` folders and then created a single file per type in there. For example, the Animation namespace has 7 unique typedefs at the moment, so I moved the defs out of the source code and into that folder:
Each of those JS files only contains a JSDoc typedef block. There is no code in them, so they don’t add anything to the library size. Here is the typedef file for the Animation object:

```javascript
//
* @typedef {object} Phaser.Ani
* @since 3.8.0
* *
* @property {string} [key] - The key that the animation will be associated with. i.e. sprite.animations.play(key)
* @property {Phaser.Ani.Types.AnimFrame[]} [frames] - An object containing data used to generate the frames for the animation
* @property {string} [defaultTextureKey=null] - The key of the texture all frames of the animation will use. Can be overridden on a per frame basis.
* @property {integer} [frameRate] - The frame rate of playback in frames per second (default: 24 if duration is null)
* @property {integer} [duration] - How long the animation should play for in milliseconds. If not given its derived from frameRate.
* @property {boolean} [skipMissedFrames=true] - Skip frames if the time lags, or always advanced anyway?
* @property {integer} [delay=0] - Delay before starting playback. Value given in milliseconds.
* @property {integer} [repeat=0] - Number of times to repeat the animation (-1 for infinity)
* @property {integer} [repeatDelay=0] - Delay before the animation repeats. Value given in milliseconds.
* @property {boolean} [yoyo=false] - Should the animation yoyo? (reverse back down to the start) before repeating?
* @property {boolean} [showOnStart=false] - Should sprite.visible = true when the animation starts to play?
* @property {boolean} [hideOnComplete=false] - Should sprite.visible = false when the animation finishes?
*/
```

As you can see it’s a description of an Animation object with all of the properties listed and described. Having it like this means I can reference this type in the rest of the docs, such as in the `AnimationManager.create` method:
In the code above you can see that the parameter needs to be an Animation configuration object. Previously, before typedefs, lots of methods this like would have just said 'object' as the type. With typedefs you now know the exact shape of the object it expects. This should prove especially useful for TypeScript users too.

Of course, creating all of these types was a lot of work. And it was that mind-numbing labor-intensive type of work, too. Absolutely zero fun, zero creativity, no challenge. Just hour after hour, day after day, of cutting and pasting. The things I do for you guys :) Ultimately, it was worth it. It instantly made the JSDocs better. Even using the current lacklustre template it improves it hugely, because types are organized into a single place per namespace, and you can click them when viewing a method that uses one as a parameter and get taken right to the definition for it.

**create(config)**

Creates a new Animation and adds it to the Animation Manager. Animations are global. Once created, you can use them in any Scene in your game.

If an invalid key is given this method will return `false`.

If you pass the key of an animation that already exists in the Animation Manager a brand new animation is only created if the key is valid and not already in use.

If you wish to re-use an existing key, call `AnimationManager.remove` first, then try!

**Parameters:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>config</td>
<td>PhaserAnimationsTypesAnimation</td>
<td>The configuration for the new Animation.</td>
</tr>
</tbody>
</table>
So, time well spent, despite the monotony of it. I find this to be a recurring trend when it comes to Phaser, or I guess any large scale open source project. There's the fun creative side of it, like adding cool new features such as improving something in the renderer. There's the challenging 'triumphant' side of it when you fix a particularly gnarly bug and get to close a bunch of GitHub issues. It's not creatively satisfying, but it's rewarding all the same. And then there is all the more menial tasks like the documentation. It's not glamorous, it rarely taxes the old grey matter, but in its own way it's just as important. You could probably argue even more so and have a valid point.

I tend to think of working on Phaser as like playing an RPG. Phaser is my main character. It has attributes such as 'documentation', 'tutorials', 'community support' and 'features' that I have to keep working on improving and raising. Making sure not to leave any one of them lagging behind too far. It has a health count, which I guess is the active number of people using it. When bugs are posted it takes damage and you have to counter-act that, least it bleed out. And finally, with new versions, it levels up. It's a constant journey of self-improvement and discovery.
Who knows what will happen when we reach level 20. Maybe it'll be time to start earning epic boons :)

**JSDoc to SQLite**

As I mentioned at the start of this Dev Log my work on the docs this week was two-folder. Firstly, doing all the types as mentioned above. And secondly working on both my JSDoc to SQLite script and the new docs interface. JSDoc has the ability to generate a JSON dump of the documentation, rather than HTML. It's a massive unrelenting file. The Phaser JSON export is over 8MB in size.

While there are tools that exist to query JSON as if you're working with a database, mostly expanding on the JSON Pointer format, this isn't ideal because the very structure of the JSON isn't conducive to the sorts of data we wish to
group together. It made far more sense, to me at least, to get the data into a database where it can be searched, queried and cross-referenced with ease. I'm a big fan of SQLite. The single file approach, combined with extensive support from both the JavaScript and PHP side (unlike, say, MySQL) makes it a perfect candidate for this. So I took the JSON to SQLite script I coded almost a year ago and finished it off. I expanded the structure of the tables, reworked the script to make it more modular and ran lots of tests. I'm pleased to say this work paid off and I've now got a work flow that allows me to quickly export the entire Phaser API documentation to JSON and then to SQLite in a matter of seconds. It also means I can ship the docs as single SQLite files along with each release.

My hope is that with the docs in this format, more tools can make use of them and grow from around them. For example, I think Phaser Editor is a good candidate for making use of this. But there's no reason why they could also form the basis of extensions for VSCode or a Sublime plugin.

The main reason of having the docs in this format is so that I can build a much better interface to them on the Phaser site. Which is what I spent the final two days of this week doing. My initial approach was to re-create the JSDoc template, just with better formatting. It was easy to get to this point thanks to a combination of using Bootstrap 4 and being able to quickly query the SQLite file and extract what is needed. I still need to spend more time refining the pages but I'm confident that I can get the docs live on the Phaser site next week. This is important for two reasons. The first, of course, is that they look and work a whole lot better than the JSDoc template. The layout is cleaner and less cluttered. You can quickly and easily navigate between sections and toggle things you don't need on and off automatically. I can also tweak the layout and keep improving it
over time, without having to publish another 60MB+ to GitHub.

The second reason is actually monetary. I run small un-obtrusive ads on the Phaser site using a combination of Carbon and CodeFund. The money they generate, while not being a large amount, is still enough to cover the cost of hosting the Phaser site and also put a bit towards the general fund pot. However, up until now, the Phaser 3 docs have been hosted on GitHub pages. This is taking traffic away from the Phaser site and not counting towards ad revenue either. Both are things I need to reverse.

People find it very confusing to go to the main site and see the docs (and examples, for that matter) are all Phaser 2 based. Not even Phaser CE, but the several years old 2.6.2 release. And for anything more modern they're taken away to another site, with another style. It's just all wrong and too disparate. Moving the docs back to the site is stage 1. Stage 2 is doing the same for the examples and updating the sandbox. The final stage is a complete site redesign, for which I have lots of plans - but this is a marathon, not a sprint, so let's take it one step at a time.

Translation Help Needed

One thing I'd love to do is get the Phaser guides translated into other languages. I appreciate that translating the whole documentation is a truly massive task, so I feel that the tutorials like the Getting Started and Making your First Game are much better suited as they are shorter and more focused.
Is this something you'd be able to help with? Do you have a bit of spare time to help translate any of the tutorials to languages other than English? If so, please email me (rdavey@gmail.com).

Just let me know which language you are willing to translate from English to. Then I'll organize who does which tutorial. This is so I can make sure we don't have any overlap with more than one person doing the same tutorial in the same language.

I'm open to translations to any language, but would love if we could include Chinese or Japanese. You will be credited as the translator which can include a link to whatever site you like (your own site, a game you've released, etc). These guides have a lot of monthly readers so it'll be brilliant to offer them in languages other than English.

**What's Next**

I'm going to carry on working on the docs, so I can finally get them published to the web site. Any extra time will be spent on Phaser 3.16.3. A couple of issues have arrived, including yet another with the damned full screen api :) which needs addressing, so I'll package up some more small fixes into another point release, hopefully towards the end of the week. As usual, check out the next dev log for further details!
Bassoon Tracker is a fully in-browser soundtracker editor and player. The beautiful interface is 100% canvas driven too! Check out some of the classic demo songs.

swc 1.0 is out. swc stands for 'speedy web compiler' and is an alternative to babel that runs up to 20x faster.

A glass of water and lemons still life, created entirely in CSS!

---

Phaser Releases

**Phaser 3.16.2** released February 11th 2019.
**Phaser CE 2.12.0** released 6th February 2019.

Please help support Phaser development

Have some news you'd like published? Email support@phaser.io or tweet us.

Missed an issue? Check out the Back Issues page.