

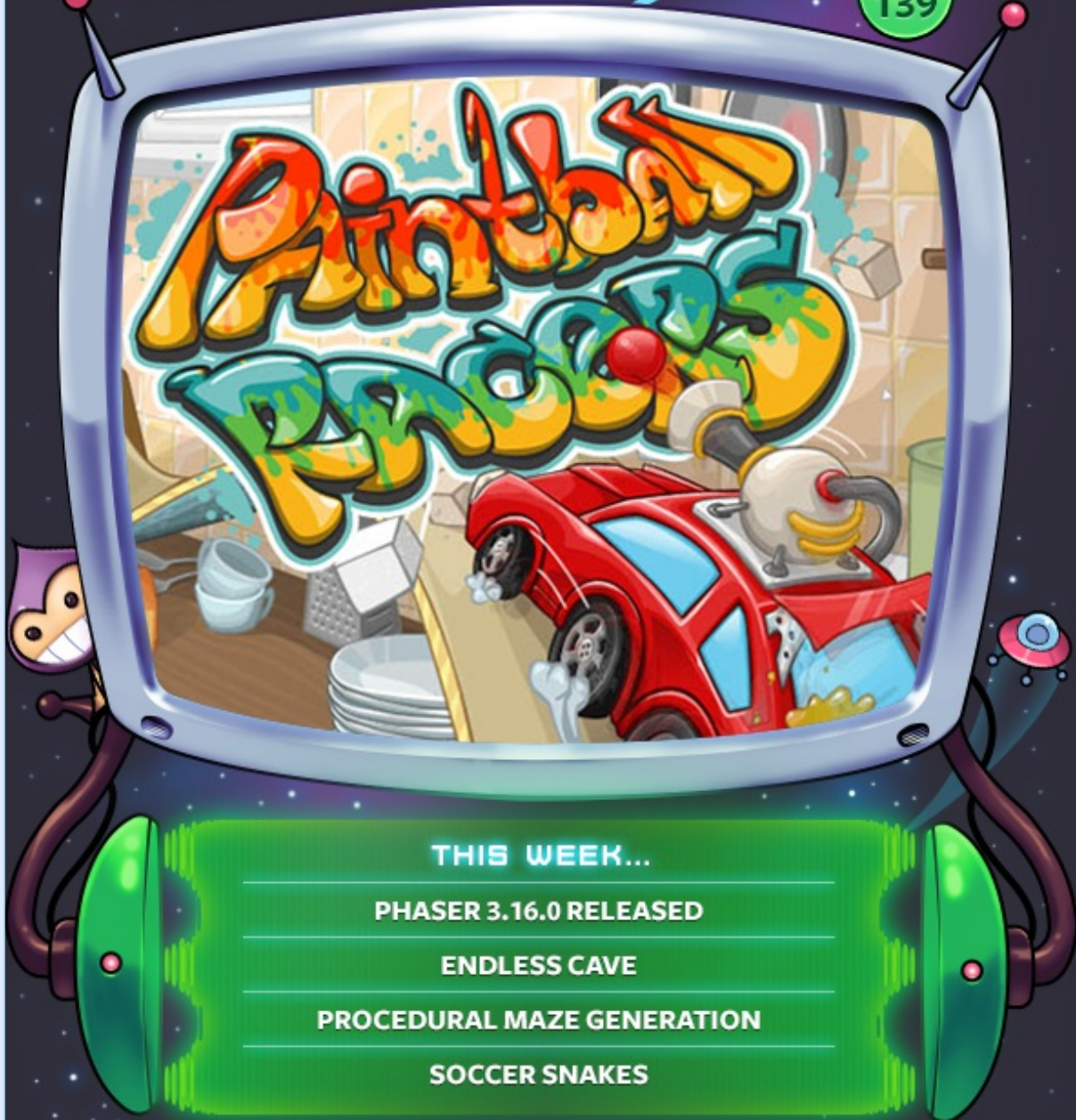
[Web Version](#)

[Unsubscribe](#)

PHASER WORLD

FEBRUARY 2019

ISSUE
139



Welcome to Issue 139 of Phaser World

After months of hard work, Phaser 3.16 is finally released! You can read all about it in this issue's Dev Log along with how to get it up and running. Once you've digested that, be sure to play the great games on offer and read the new bunch of tutorials. If the Phaser community is anything, it's prolific, in terms of both the amazing games you make and the tutorials you create :)

Be sure to [read this newsletter on the web](#) so you don't miss anything, as it's a biggun.

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!



Phaser Sticker Packs + badges + freebies

The Phaser sticker packs continue to prove popular :) and each sale helps contribute towards the development of Phaser, so it's win-win! All orders placed this month will get a free **Phaser pin-badges** and a new **transparent sticker** included as well.

[Click here to order a sticker pack.](#)



The Latest Games



Game of the Week

[Paintball Racers](#)

Beat all the racers while being bombarded with paint balls! In this fast and fun physics hill racing game.



Staff Pick

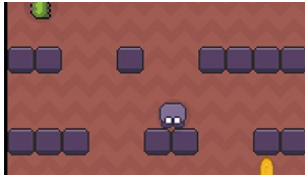
[Esciit](#)

Crash into the obscure in this surreal adventure platformer, full of low-fi pixel charm, story telling roots and the creepiest bunnies you'll ever see.



[Soccer Snakes](#)

Unleash the beasts! Snakes are battling it out in their own World Cup. Can you score 3 goals before your opponent?



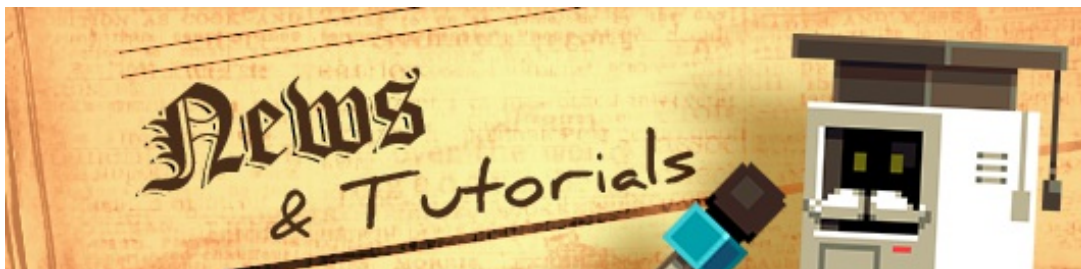
Endless Cave

It's a vertical endless runner with dungeon crawler elements! Avoid those slimes, grab the gold and see just how deep you can get.



Spiked Heart

A visual novel about Spike the cat. You've only a few hours to try and figure out the love of your life, or else.

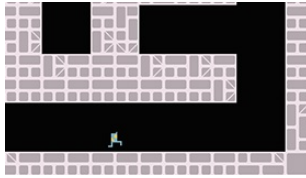


What's New?



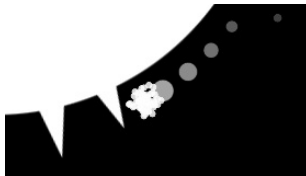
Mobile Game Devs Tutorials 1

The first in a series of videos about creating mobile games with Phaser 3 from scratch.



Procedural Maze Generation in Phaser 3

A tutorial on how to create perfect mazes for your game, including Phaser 3 plugin and demo code.



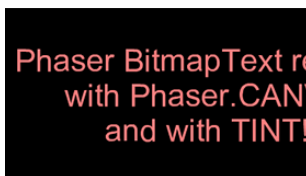
Circular Endless Runner Tutorial Part 4

In this part the game is enhanced with double jumps, particle trails, explosions and camera effects.



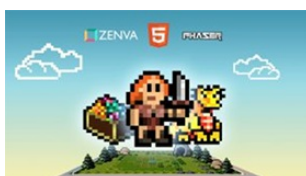
How to re-use Animations Tutorial

A tutorial detailing a flexible way to add and remove skins from Sprites.



Bitmap Text Canvas Tint Support Tutorial

Learn how to add tint support to BitampText even when using the Canvas Renderer in Phaser 3.



Phaser 3 Game Development Course

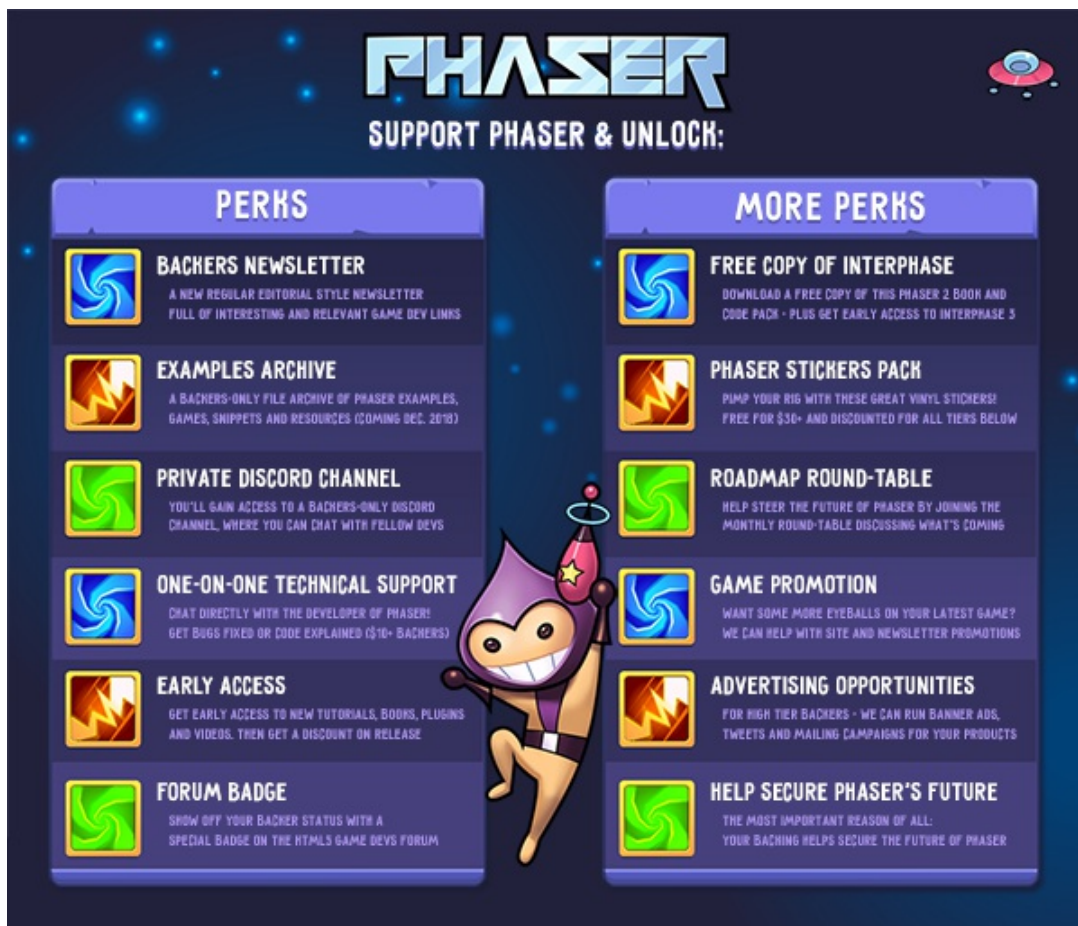
A complete Phaser 3 and JavaScript Game Development package. 9 courses, 119 lessons and over 15 hours of video content. Learn to code and create a huge portfolio of cross platform games.

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:

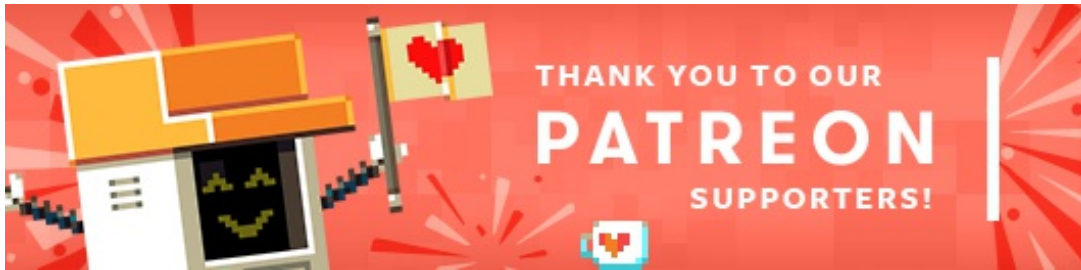


Click to see the full list of backer perks

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:

Alpha Coders
Dimitris Sarmis
Endre Johnsen
Florent LEVASSEUR
Jack Le Riche
James Pierce
jestarray
Sam Sawyer
Spencer Jones



Dev Log #139

Welcome to Dev Log 139. On February 5th, just past midnight, I finally did it. I slammed that release button ...



... and Phaser 3.16 was born.

This is the culmination of 4 months of hard work that I started in October 2018, and other than a short break for Christmas, worked on solidly until release today.

It brings together a huge number of significant changes, which I'll talk about in this Dev Log and in the coming weeks. Most importantly of all, though, it finishes the Phaser 3 core feature set. Yes, of course there is more I'd love to do with Phaser, but 3.16 represents the end of a very long journey. One I often wondered if we'd ever see. There will be time for reflection later. For now, it's time to celebrate and dive headfirst into the update.

There's simply too much to cover each item in detail in a single Dev Log, so let's start with the headliners, and the things you should be aware of when upgrading.

3.16.1 Download

You can download Phaser 3.16.1 from the [GitHub releases page](#), or pull it in via the [npm package](#). There are also the pre-built files available on the [jsDeliver CDN](#).

What happened to 3.16.0 I hear you cry? Well, sadly, it was an npm issue. While publishing the 3.16 Beta versions a few weeks ago, I accidentally published a version without using the beta tag, which made it become 3.16.0 immediately. You can 'unpublish' npm packages, as long as they are < 72 hours old, which I did immediately. However, even after unpublishing successfully, npm, in their infinite wisdom, lock the version number you used accidentally, forever. It can never be used again. My apologies to those of you who were on the Phaser

Slack channel last night when I discovered this. Things got a little fruity :)

So, in order to keep it all clean, I had to jump immediately to 3.16.1. There is literally no difference between that and 3.16.0, it's just an infuriating version bump, nothing else.

Documentation and TypeScript

As part of the release, I generated the brand new 3.16 documentation and published it to GitHub. As before, you can [read it online](#), or check out the [repo](#) to have a local copy instead.

The 3.16 documentation is the best the Phaser 3 docs have ever been. The painstaking work I undertook redoing all of the Phaser Events are now visible for all to see! You can clearly find them in the Events drop-down, and every method that fires an event now links directly to the event too. This release also incorporates all the effort the community put into the Phaser Doc Jam. We collectively finished hundreds of areas of docs. As it stands today, there are just 748 items left to document in Phaser and then we'll have 100% coverage. It may sound like a lot, just remember before we started the process there were over 16,000 items to document, so it's a massive achievement for such a short space of time. I will focus on getting it down to zero in the coming months, and of course, every single new feature or change made in the API has been documented fully from the start.

I also spent a long, *long* time, getting the [TypeScript definitions](#) sorted out. It took some serious refactoring of the JSDocs, nevertheless, the defs are now complete. Not a single compiler warning is thrown now! I definitely want to improve the defs further, especially with regard to the new events, still, I'm happy with their current status. Again, a lot of worthwhile work. A number of you have been asking if the defs can be either moved into the main Phaser repo, or to another like DefinitelyTyped. Once we've got events strictly typed in there, this can now happen.

It's quite incredible just how much effort all the *other* tasks around Phaser take. Documentation, TypeScript defs, examples, change logs, support. The list goes on. It'd be nice to think we could focus on just developing the API; enhancing code and features, however, it's actually only part of the jigsaw. An important part, yet by no means the only one.



Breaking Changes

Yes, there are some. Depending on how you code your games, you may not even notice them. I've heard from people who upgraded to 3.16 and didn't have to change a single thing. Yet I'm sure many will need to adapt their code.

There have been some important namespace and event name changes. You can see the full list in the [Release Notes](#) but these are the most used ones:

From underscores to hyphens

Phaser dispatches 'keyed' events, from the Data Manager, Input system and elsewhere. A keyed event is when the object name is part of the event string. For example, ``keydown_SPACE`` would listen for the space key specifically. In 3.16 all keyed events have changed so instead of using an underscore, you now use a hyphen. So, the previous event has become ``keydown-SPACE``. This change has happened for keydown, keyup and changedata, from the Data Manager.

Event Name Changes

For reasons of consistency, some event names were changed:

- The ``ended`` event

in
`WebAudioSound`
has
been
renamed
to
`complete`.

- The
`ended`
event
in
`HTML5AudioSound`
has
been
renamed
to
`complete`.

- Game
Objects
used
to
emit
`collide`
and
`overlap`
events
if
they
had
an
Arcade
Physics
Body
with
`onCollide`,
or
`onOverlap`
set
and
collided
with
a
Tile.
The
events
have

been
renamed
to
``tilecollide``
and
``tileoverlap``
respectively.
Listen
for
them
from
the
Arcade
Physics
World
itself,
i.e.:
``this.physics.world.on('tilecollide')``,
as
Game
Objects
no
longer
emit
them.

If you use the new Event constants, instead of strings, you don't need to worry about changes like this again :)

Important Changes to the Input System

Until Phaser 3.16, the Input system worked using an event queue. All native DOM events, such as from the Keyboard, Touch or Mouse, were stored in a queue within the Input Manager. The queue was processed during the next game step. All the events were dealt with and then it was cleared. As they were processed, the internal events such as ``pointerdown`` or ``keyup`` were dispatched to your game code.

This worked fine in that you were able to guarantee *exactly* when the events would arrive because it was always at the same time in the game step. However, it had the side effect of you not being able to do things like open external browser windows, or go into Full Screen mode, during your event handlers - because they weren't "real" events, so didn't pass the browser security checks. To this end, methods like ``addUpCallback`` were added to try and provide this support (although it was never possible for keyboard events).

In 3.16 this has changed. The DOM Events now trigger the respective internal events immediately, in the same invocation. So if you click on the canvas, the ``pointerdown`` event you receive in your game is still part of the 'native' event handler, so you're now free to do things like go into full screen mode, or open external windows, without any browser warnings or work-arounds.

It does, however, mean that the point at which these handlers are called is no longer always consistent, and is no longer bound to the speed of the Request Animation Frame update. We've tested as much as possible, and so far, things carry on working as before. We've noticed a slight increase in responsiveness, due to the removal of the fractional delay in processing the events, which feels good. However, if for whatever reason this change has broken your game then you're able to easily switch back to the previous version: in your Game Config, create an ``input`` object and give it the property ``queue: true``. This will tell Phaser to use the legacy event queue system.

We *will* remove this legacy system in the near future. So, please try and adapt your games to use the new system. If you've found an edge-case where something breaks, report it so we can look into it.

There are lots and lots of changes to the way input is handled in 3.16. Please do look at the Change Log for the full list.

That's it for the most important 'breaking' changes. If something else no longer works, it wasn't intentional. So, please report it on GitHub by opening an issue!



Out of the Game

3.16 introduces a number of ways of telling what's going on outside of your game canvas when it comes to input events. Previously, if you were to click down on a Sprite, move the pointer outside of the game canvas, then release the button, your game would never know about it. Worse, it'd think the button was still held down.

To address this, 3.16 will now monitor input events on the ``window`` object if it's able to. This means you now get ``pointerup`` events on your Game Objects, regardless if the pointer was inside the canvas or not. It's entirely possible that other DOM elements on the page steal the event, but sadly that's well outside the responsibility of Phaser.

Conversely, there are new pointers event such as [POINTER_DOWN_OUTSIDE](#) which allows you to tell if someone has clicked outside of your game canvas. Pointer objects have the new ``downElement`` and ``upElement`` properties, so you can tell which DOM element as actually clicked or released on. Most the time this will be your game canvas, but not always.

Finally, you can also use the events [GAME_OUT](#) and [GAME_OVER](#) to tell when a pointer leaves and returns to your game canvas, without being linked to a button press. Perhaps, if the pointer leaves, you could automatically pause your game? Each game is different, but at least you can now determine what is going on.

Keyboard handling has received a similar update. Previously, if you were to hold down a key, and then trigger an action that stole keyboard focus (for example, resizing the browser window), the key state would get stuck 'down', but 3.16 will reset it. You may not have actually released the key, however, focus was lost, so you're clearly not playing the game any longer either.

New Features

Oh boy, where to start?! There are hundreds of new features in 3.16. Too many to cover at once, so instead I've decided to pick just a handful, in no particular order, which I'll explain in more depth. I'll continue this process over the next few Dev Logs as well. First up, external Scene files.

External Scene Files

If there is one thing I've learned over the years, it's that when it comes to game dev, we *all* have our own unique ways of doing things. One of those is how you approach the architecture of your project. For some of you, you can fit your whole game into a single JS file. For others, you'll start out with a bundler like Parcel or

Webpack and have that in place before you even start coding. Another way, that was recently brought to my attention, was that they wanted the ability to load their game code in progressively.

The initial bundle would contain only the base code along with a few assets. Then they wanted to be able to talk to their website API, figure out which path the user had taken and then load a new Phaser Scene via HTTP as a result. All told, there were likely to be hundreds of potential Scenes that could be loaded, so bundling them all into a single package up front wasn't appealing, and also meant that the core game would need updating every single time a new 'Scene' was published.

Which got me thinking: there was literally no reason why the existing Phaser Loader couldn't handle this. It already had filetype support for JS files and plugins, so it would only take a small evolution of this to provide support for loading Scenes from external files. And thus, the new "SceneFile" Loader FileType was created.

Let's assume that the following is the mini-game Scene we wish to load. All it does is show a picture and rotate a sprite, but that's fine for this purpose:

```
class MiniGame520 extends Phaser.Scene {  
  
  constructor ()  
  {  
    super('game520');  
  }  
  
  preload ()  
  {  
    this.load.image('face', 'assets/pics/bw-face.png');  
    this.load.image('arrow', 'assets/sprites/longarrow.png');  
  }  
  
  create ()  
  {  
    this.face = this.add.image(400, 300, 'face');  
    this.arrow = this.add.sprite(400, 300, 'arrow').setOrigin(0, 0.5);  
  }  
  
  update ()  
  {  
    this.arrow.rotation += 0.01;  
  }  
}
```

It's a standard ES6 Scene, just like you'd create when building locally. To load it, you can use the new `load.sceneFile` method:

```
this.load.sceneFile('MiniGame520', 'js/minigames/520.js');
```

It's important that the key given to the Scene File loader is the *class name* of your Scene. For example, in the class above it is called `MiniGame520`, so that is the name we use in the call to the loader.

Once the Loader has pulled the JS file down, it's added to the DOM via a script tag and ingested by the Scene Manager. Assuming no errors, you can then access the Scene using its key:

```
this.scene.start('game520');
```

As with all Scenes, the key used in this case is the one you used in the Scene constructor. Remember it like this: The Scene key is the unique name of the scene within the Phaser Scene Manager, and the key used for loading is the class name of your Scene.

Using this feature you now have a way for a Phaser Game to programatically load in new sections from the file system. Something that, one day, may prove to be extremely useful to you.

What Next?

3.16 was such a huge release that there are absolutely going to be bugs, despite the many days of testing I did. It's a strange feeling. Of course, I'm really happy I finally published it. However, at the same time I'm apprehensive it's going to cause some of you problems. There is no avoiding this I suppose. If you do find an issue, please [report it](#) (and do so nicely). I'll be monitoring the issues list as I start work on 3.17, so there's a good chance your problem can be resolved fast.

My plan for the rest of February is to not start any new features. I will bash the GitHub Issues list into shape to get that total down. There are lots of issues related to tweens and timelines, so a short burst of energy on those specifically will allow me to close off a decent number. I'm also going to complete the Spine Plugin and more of the docs.

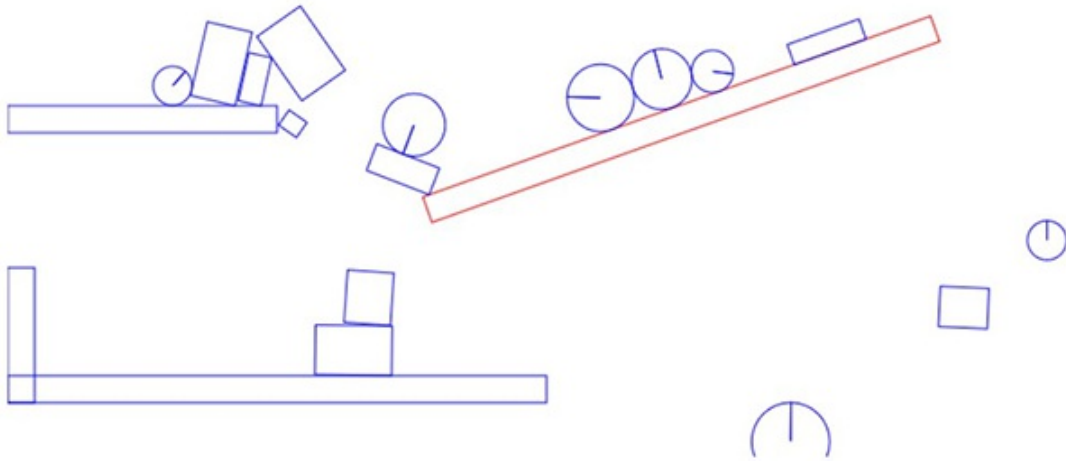
There are some large changes I want to make re: Containers, however I need to let this version bed-down a bit first. This means 3.17 should, theoretically, be more of a maintenance release that I can get out quickly, before starting the

overhaul that Containers will require through-out the core. Outside of the API I'm also going to start work on the new website and authoring more tutorials, which backers on the Phaser Patreon can expect to see arriving this month.

As I said at the start, it's been a heck of a journey to get to this point. To all of you who have joined me on this trip, either by supporting Phaser on Patreon, or submitting PRs or issues, I humbly thank you. Let's use February to calm the changes down, focus on bugs and then look to the future and where we can take it.

Until then, kanpai!





A [2D Physics system](#) in 1.6KB :)

[Sheety](#) - This sweet little web app will turn any Google Sheet into an API instantly, for free!

[Something incredible](#) is happening on Reddit (now, how often does that really happen?!)

Phaser Releases

Phaser 3.16.1 released February 5th 2019.

Phaser CE 2.11.1 released October 2nd 2018.

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.



©2019 Photon Storm Ltd | Unit 4 Old Fleece Chambers, Lydney, GL15 5RA, UK

[Preferences](#)

[Forward](#)

Powered by **Mad Mimi®**
A GoDaddy® company