Hi,

I would like to be able to ask a class or module what its source location is without knowing the name. For example, I want to do this:

```ruby
module A
  class B
  end
end
p A::B.const_source_location
```

In other works `A::B.const_source_location` would be equivalent to `A.const_source_location(:B)`.

The reason I want to do this is because sometimes it is very difficult to get the name of a constant, and sometimes I don't have access to the constant that "encloses" the class or module.

One example:

```ruby
ObjectSpace.each_object(Class) do |k|
  p k.const_source_location
end
```

In this case I have class objects, but I can't tell what constant `k` was defined `inside` of. Also I can't trust the "name" method on `k` because sometimes it's not the default method (of course I could work around that, but it's not fun).

I've attached a patch that implements the feature, and there is a PR [here](http://example.com).

Side note: I think I would like "source_location" better than `const_source_location`, but I wanted to just file a feature request so we could talk about the feature in general.

Related issues:
- Related to Ruby master - Feature #13383: [PATCH] Module#source_location

History

#1 - 03/26/2021 12:12 AM - dylants (Dylan Thacker-Smith)
I agree that `source_location` would be better for this new feature, since it is referring to the receiver of the call. `const_source_location` makes sense for the existing method, since it is referring to the source location of something else, a nested constant. So I think you should make a new `source_location` method rather than trying to combine them.

#2 - 03/26/2021 02:01 AM - shevegen (Robert A. Heiler)
Personally I agree with Aaron's use case or the general potential usefulness of being able to query `const_source_location` where Aaron's suggestion is a simplification over `A::B` (because the ruby user no longer HAS to know that specific name, since the ruby user can just call a method instead, in this case simply `.const_source_location`); and I love introspection anyway, so +1.

To the comment about the `name` (s) : `.source_location` is in general better than `.const_source_location`, in my opinion, for two reasons:

(1) it's shorter. :) We all like being concise ... two words are better than three words most of the time. (One word may be even better but we often have clashes with other names/methods, so two words are a bit better in these cases.)
(2) even aside from (1), I believe .source_location is better as a name from a design point of view. While Dylan reasoned that .const_source_location() makes more sense, IMO we could actually go the other approach and say "ruby, I don't quite care if it is a constant, or a method, or a class, or anything, I just want to query the source_location to it". So I think it depends on the point of view you have for the name. In my opinion, omitting "const_" would be perfectly fine; also I struggle with things such as "when to use class_eval or instance_eval" so I am all for simpler names in general. ;)

Now we do have Method#source_location but I think and also const_source_location as well - but I think it would be simpler to "unify" towards one general name, no matter what that name may be, if we look at it from the point of view of "how ruby users may want to use the feature". So from that point of view, from a ruby user perspective, I think it would be better to omit "const_" completely. But this is quite secondary; IMO the use case described makes sense.

#3 - 03/26/2021 06:57 AM - sawa (Tsuyoshi Sawada)
I also think that a method name other than const_source_location should be used for this feature because this feature has nothing to do with constants. Modules are not necessarily constants, and constants are not necessarily modules.

#4 - 03/26/2021 04:09 PM - tenderlovemaking (Aaron Patterson)
- File 0001-Add-constant-location-information-to-classes.patch added

I've renamed it to source_location. So it's A::B.source_location.

#5 - 03/26/2021 05:44 PM - Eregon (Benoit Daloze)
Module#source_location sounds good to me.

I wonder if we should try to collect all places where the class/module is opened/reopened. const_source_location returns where the module was first defined, but there we might want to know e.g. in which files that class was modified.

I could imagine that could be useful in an IDE or debugger. OTOH it's probably not so reasonable for class_exec/module_exec which could be done in a loop potentially.

#6 - 03/26/2021 05:45 PM - Eregon (Benoit Daloze)
Actually a good way to list all files where methods were added to the class is to go through instance_methods and ask their source_location, so that may be good enough for that purpose.

#7 - 04/14/2021 04:50 AM - ko1 (Koichi Sasada)
- Related to Feature #13383: [PATCH] Module#source_location added

#8 - 04/16/2021 06:21 AM - mame (Yusuke Endoh)
@tenderlove I'm not sure about the use case. What purpose do you want this for? Memory profiling?

const_source_location and source_location are very similar but different for an anonymous class. The former returns the site where the class object is assigned to a constant, while the latter returns the site where it is created. But in almost all cases, they are same and redundant. I don't think it is a good idea to introduce similar and redundant fields in terms of maintenance, unless it is really needed. So, I'd like to ask a question: is the difference really needed for your use case?

(I'm never negative for this proposal, but I wanted to confirm.)

#9 - 04/16/2021 05:59 PM - tenderlovemaking (Aaron Patterson)
mame (Yusuke Endoh) wrote in #note-8:

@tenderlove I'm not sure about the use case. What purpose do you want this for? Memory profiling?

const_source_location and source_location are very similar but different for an anonymous class. The former returns the site where the class object is assigned to a constant, while the latter returns the site where it is created. But in almost all cases, they are same and redundant. I don't think it is a good idea to introduce similar and redundant fields in terms of maintenance, unless it is really needed. So, I'd like to ask a question: is the difference really needed for your use case?

(I'm never negative for this proposal, but I wanted to confirm.)

For my use case the difference doesn't matter. If we had namespace as in #17753, then I think I could write klass.namespace.const_source_location(klass.name) (or make the implementation do this).
But to confirm the difference between "allocation location" vs "assignment location" doesn't matter to me.

#10 - 04/17/2021 07:38 AM - mame (Yusuke Endoh)
Thank you! Then, I think the patch should use const_source_location info instead of adding file and line fields.

tenderlovemaking (Aaron Patterson) wrote in #note-9:

For my use case the difference doesn't matter. If we had namespace as in #17753, then I think I could write
klass.namespace.const_source_location(klass.name) (or make the implementation do this).

Sorry I am missing something but if an anonymous class does not matter at all, I think you are already able to write
Object.const_source_location(klass.name) even if #17753 is not accepted.

Files

<table>
<thead>
<tr>
<th>Patch Path</th>
<th>Size</th>
<th>Date</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001-Add-constant-location-information-to-classes.patch</td>
<td>5.34 KB</td>
<td>03/25/2021</td>
<td>tenderlovemaking (Aaron Patterson)</td>
</tr>
<tr>
<td>0001-Add-constant-location-information-to-classes.patch</td>
<td>5.97 KB</td>
<td>03/26/2021</td>
<td>tenderlovemaking (Aaron Patterson)</td>
</tr>
</tbody>
</table>