RFC process

What is the RFC process?

Many changes, including bug fixes and documentation improvements can be implemented and reviewed via the normal GitHub pull request workflow.

Some changes, however, are "substantial", and we ask that these be put through a bit of a design process and produce a consensus among the Gatsby core team.

The "RFC" (request for comments) process is intended to provide a consistent and controlled path for new features to enter the project.

- Active RFC List
- List of implemented RFCs

When to follow this process

You should consider using this process if you intend to make "substantial" changes to Gatsby or its documentation. Some examples that would benefit from an RFC are:

- A new feature that creates new API surface area, and would require a feature flag if introduced.
- The removal of features that already shipped as part of the release channel.
- The introduction of new idiomatic usage or conventions, even if they do not include code changes to Gatsby itself.

The RFC process is a great opportunity to get more eyeballs on your proposal before it becomes a part of a released version of Gatsby. Quite often, even proposals that seem "obvious" can be significantly improved once a wider group of interested people have a chance to weigh in.

The RFC process can also be helpful to encourage discussions about a proposed feature as it is being designed, and incorporate important constraints into the design while it's easier to change, before the design has been fully implemented.

Some changes do not require an RFC:

- Rephrasing, reorganizing or refactoring addition or removal of warnings
- Additions that strictly improve objective, numerical quality criteria (speedup, better browser support)

- Additions only likely to be *noticed by* other implements-of-Gatsby, invisible to users-of-Gatsby.

What the process is

In short, to get a major feature added to Gatsby, one usually first puts up a Discussion on GitHub. At that point the RFC is 'active' and may be implemented with the goal of eventual inclusion into Gatsby.

- Copy the RFC template into your clipboard:

```
# Summary

Brief explanation of the feature.

# Basic example

If the proposal involves a new or changed API, include a basic code example.
Omit this section if it's not applicable.

# Motivation

Why are we doing this? What use cases does it support? What is the expected
outcome?

Please focus on explaining the motivation so that if this RFC is not accepted,
the motivation could be used to develop alternative solutions. In other words,
enumerate the constraints you are trying to solve without coupling them too
closely to the solution you have in mind.

# Detailed design

This is the bulk of the RFC. Explain the design in enough detail for somebody
familiar with Gatsby to understand, and for somebody familiar with the
implementation to implement. This should get into specifics and corner-cases,
and include examples of how the feature is used. Any new terminology should be
defined here.

# Drawbacks

Why should we _not_ do this? Please consider:

- implementation cost, both in term of code size and complexity
- whether the proposed feature can be implemented in user space
- the impact on teaching people Gatsby
- integration of this feature with other existing and planned features
- cost of migrating existing Gatsby applications (is it a breaking change?)
```

```
There are tradeoffs to choosing any path. Attempt to identify them here.

# Alternatives

What other designs have been considered? What is the impact of not doing this?

# Adoption strategy

If we implement this proposal, how will existing Gatsby developers adopt it? Is
this a breaking change? Can we write a codemod? Should we coordinate with
other projects or libraries?

# How we teach this

What names and terminology work best for these concepts and why? How is this
idea best presented? As a continuation of existing Gatsby patterns?

Would the acceptance of this proposal mean the Gatsby documentation must be
re-organized or altered? Does it change how Gatsby is taught to new developers
at any level?

How should this feature be taught to existing Gatsby developers?

# Unresolved questions

Optional, but suggested for first drafts. What parts of the design are still
TBD?
```

- Open a new GitHub discussion in the RFC category
- Fill in the RFC. Put care into the details: **RFCs that do not present convincing motivation, demonstrate understanding of the impact of the design, or are disingenuous about the drawbacks or alternatives tend to be poorly-received**.
- As the RFC will receive design feedback from the larger community, you should be prepared to revise it in response. So build consensus and integrate feedback. RFCs that have broad support are much more likely to make progress than those that don't receive any comments.
- Eventually, the team will decide whether the RFC is a candidate for inclusion in Gatsby.
- RFCs that are candidates for inclusion in Gatsby will enter a "final comment period" lasting 3 calendar days. The beginning of this period will be signaled with a comment.
- An RFC can be modified based upon feedback from the team and commu-

nity.

- An RFC may be rejected by the team after public discussion has settled and comments have been made summarizing the rationale for rejection. A member of the team should then add the label "status: rejected".
- An RFC may be accepted at the close of its final comment period. A team member will add the label "status: accepted".

The RFC lifecycle

Once an RFC becomes active, the authors may implement it and submit the feature as a pull request to the Gatsby repo. Becoming 'active' is not a rubber stamp, and in particular still does not mean the feature will ultimately be merged; it does mean that the core team has agreed to it in principle and are amenable to merging it.

Furthermore, the fact that a given RFC has been accepted and is 'active' implies nothing about what priority is assigned to its implementation, nor whether anybody is currently working on it.

Modifications to active RFCs can be done by editing the discussion/commenting on it. We strive to write each RFC in a manner that it will reflect the final design of the feature; but the nature of the process means that we cannot expect every accepted RFC to actually reflect what the end result will be at the time of the next major release; therefore we try to keep each RFC document somewhat in sync with the language feature as planned, tracking such changes via followup changes to the document.

Implementing an RFC

The author of an RFC is not obligated to implement it. The RFC author (like any other developer) is welcome to post an implementation for review after the RFC has been accepted.

If you are interested in working on the implementation for an 'active' RFC, but cannot determine if someone else is already working on it, feel free to ask (e.g. by leaving a comment on the associated issue).

Reviewing RFCs

Each month the team will attempt to review some set of open RFC discussions.

Every accepted feature should have a core team champion, who will represent the feature and its progress.

Gatsby's RFC process owes its inspiration to the React RFC process, Yarn RFC process, Rust RFC process, and Ember RFC process