## Contributing to Flot ##

We welcome all contributions, but following these guidelines results in less

work for us, and a faster and better response.

### Issues ###

Issues are not a way to ask general questions about Flot. If you see unexpected

behavior but are not 100% certain that it is a bug, please try posting to the

[forum](http://groups.google.com/group/flot-graphs) first, and confirm that

what you see is really a Flot problem before creating a new issue for it. When

reporting a bug, please include a working demonstration of the problem, if

possible, or at least a clear description of the options you're using and the

environment (browser and version, jQuery version, other libraries) that you're

running under.

If you have suggestions for new features, or changes to existing ones, we'd

love to hear them! Please submit each suggestion as a separate new issue.

If you would like to work on an existing issue, please make sure it is not

already assigned to someone else. If an issue is assigned to someone, that

person has already started working on it. So, pick unassigned issues to prevent

duplicated effort.

### Pull Requests ###

To make merging as easy as possible, please keep these rules in mind:

1. Submit new features or architectural changes to the \*&lt;version&gt;-work\*

branch for the next major release. Submit bug fixes to the master branch.

2. Divide larger changes into a series of small, logical commits with

descriptive messages.

3. Rebase, if necessary, before submitting your pull request, to reduce the

work we need to do to merge it.

4. Format your code according to the style guidelines below.

### Flot Style Guidelines ###

Flot follows the [jQuery Core Style Guidelines](http://docs.jquery.com/JQuery\_Core\_Style\_Guidelines),

with the following updates and exceptions:

#### Spacing ####

Use four-space indents, no tabs. Do not add horizontal space around parameter

lists, loop definitions, or array/object indices. For example:

```js

for ( var i = 0; i < data.length; i++ ) { // This block is wrong!

if ( data[ i ] > 1 ) {

data[ i ] = 2;

}

}

for (var i = 0; i < data.length; i++) { // This block is correct!

if (data[i] > 1) {

data[i] = 2;

}

}

```

#### Comments ####

Use [jsDoc](http://usejsdoc.org) comments for all file and function headers.

Use // for all inline and block comments, regardless of length.

All // comment blocks should have an empty line above \*and\* below them. For

example:

```js

var a = 5;

// We're going to loop here

// TODO: Make this loop faster, better, stronger!

for (var x = 0; x < 10; x++) {}

```

#### Wrapping ####

Block comments should be wrapped at 80 characters.

Code should attempt to wrap at 80 characters, but may run longer if wrapping

would hurt readability more than having to scroll horizontally. This is a

judgement call made on a situational basis.

Statements containing complex logic should not be wrapped arbitrarily if they

do not exceed 80 characters. For example:

```js

if (a == 1 && // This block is wrong!

b == 2 &&

c == 3) {}

if (a == 1 && b == 2 && c == 3) {} // This block is correct!

```