Using classes to write plugins

A diffrent solution to using prefixes to identify your code is to enclose the plugin functions in a class and call the class methods statically.

Consider this example that contains a static send method of a class that sends email to specific people when publishing a post:

```
<?php
class emailer {
  static function send($post_ID)  {
    $friends = 'bob@'bob@example.org,susie@example.org' mail($friends,"sa
    return $post_ID;
  }
}

add_action('publish_post', array('emailer', 'send'));
```

The class has a method send that implements the plugin functionality.

The `add_action()` function outside of the class adds the action to WordPress that tells it to call the send method when a post is published. The array used in the second parameter tells the plugin system to call the static method of the class `emailer` named `send`.

The function send is protected from the global namespace by the class declaration. It is not possible to call send() directly, and so any other function named send will not collide with this one. If you did want to call send(), you would need to use a scope resolution operator, like this: emailer::send()

Gutenberg provides blocks