## Contents

1. About Bambu AJAX .................................................. 3
2. Installation ......................................................... 5
3. Basic usage .......................................................... 7
4. Shortcut ............................................................. 9
5. Dynamically updating page content .............................. 11
6. Additional decorators .............................................. 13
7. Todo ................................................................. 15
8. Questions or suggestions? ........................................ 17
AJAX utility functions for Django that can be accessed via a single URL (ala WordPress admin-ajax.php)
About Bambu AJAX

Bambu AJAX lets you write utility functions for your Django apps that can be called via AJAX, without having to specify a separate URL pattern and view for each one.
Install the package via Pip:

```bash
pip install bambu-ajax
```

Add it to your `INSTALLED_APPS` list:

```python
INSTALLED_APPS = (
    ...
    'bambu.ajax'
)
```

Add `bambu.ajax.urls` to your URLconf:

```python
urlpatterns = patterns('',
    ...
    url(r'^ajax/', include('bambu.ajax.urls')),
)
```
Create a file called `ajax.php` within your Django app, and import the necessary module from the `bambu-ajax` package, like so:

```python
from bambu.ajax import site

@site.register
def my_ajax_function(request):
    return ['a', 'list', 'of', 'things']
```

Using the `@site.register` decorator registers your AJAX function with the `bambu.ajax` view.

To leverage this function from within a Django template, use the `ajaxurl` template tag, like so:

```html
{% load ajax %}
<script>
    $.getJSON('{% ajaxurl 'my_project.my_app.my_ajax_function' %}&callback=?',
        function(data) {
            console.log(data);
        }
    );
</script>
```

Here, `my_project` should refer to the name of your Django project, `my_app` should be the name of the app you put your `ajax.py` file in, and `my_ajax_function` is the name of the function you defined within `ajax.py`.

The notation is similar to that used when referring to Django models, in that you always skip the common portion `ajax` from the naming convention.
Shortcut

Make your life easier by including the utility library in your template:

```html
<script src="{% url ajax_utility %}" ></script>
<script>
  bambu.ajax.get('poddle.podcasting.my_ajax_function',
      function(data) {
        console.log(data);
      }
  );
</script>
```

This achieves the same result, but in a much cleaner way.
As well as returning JSON-serialisable data per the examples above, you can also use AJAX functions just like views, in that they can return an HttpResponse object.

Using that method, the example above would print out the HTML (or other data) returned in the HTTP response.
Additional decorators

You can of course add other decorators, just as you would with normal views.
Todo

- Add a dedicated `login_required` decorator that returns a more helpful response for anonymous users
- Look into integrating this with Plunja, my dynamic JavaScript templating library.
Questions or suggestions?

Find me on Twitter (@iamsteadman) or visit my blog.