
django-dajax Documentation

Release 0.9

Jorge Bastida

Nov 16, 2017

Contents

1	Documentation	3
1.1	Installation	3
1.2	API	4
1.3	Migrating to 0.9	8
1.4	Changelog	8
2	How does it work?	11
3	Example	13
4	Supported JS Frameworks	15

Dajax is a powerful tool to easily and super-quickly develop asynchronous presentation logic in web applications, using Python and almost no JavaScript source code.

It supports four of the most popular JavaScript frameworks: Prototype, jQuery, Dojo and mootools.

Using `django-dajaxice` as communication core, Dajax implements an abstraction layer between presentation logic managed with JavaScript and your Python business logic.

With Dajax you can modify your DOM structure directly from Python.

1.1 Installation

In order to use `dajax` you should install `django-dajaxice` before. Please follow these instructions [here](#).

1.1.1 Installing Dajax

Install `django-dajax` using `easy_install` or `pip`:

```
$ pip install django_dajax
$ easy_install django_dajax
```

Add `dajax` in your project `settings.py` inside `INSTALLED_APPS`:

```
INSTALLED_APPS = (
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.sites',
    'dajaxice',
    'dajax',
    ...
)
```

Create a new `ajax.py` file inside your app with your own `dajax` functions:

```
from dajax.core import Dajax
def multiply(request, a, b):
    dajax = Dajax()
    result = int(a) * int(b)
    dajax.assign('#result', 'value', str(result))
    return dajax.json()
```

Include dajax in your <head>:

Dajax supports up to four JS libraries. You should add to your project base template the one you need.

- [jQuery 1.7.2](#) - `dajax/jquery.dajax.core.js`
- [Prototype 1.7](#) - `dajax/prototype.dajax.core.js`
- [MooTools 1.4.5](#) - `dajax/mootools.dajax.core.js`
- [Dojo 1.7](#) - `dajax/dojo.dajax.core.js`

For example for jQuery:

```
{% static "/static/dajax/jquery.dajax.core.js" %}
```

1.1.2 Use Dajax

Now you can call your ajax methods using `Dajaxice.app.function('Dajax.process')`:

```
<button onclick="Dajaxice.example.myexample(Dajax.process); ">Click here!</button>
```

The function `_Dajax.process_` will process what the server returns and call the appropriate actions. If you need your own callback, you can change the callback with a function like:

```
function my_callback(data) {  
    Dajax.process(data);  
    /* Your js code */  
}
```

And use it as:

```
<button onclick="Dajaxice.app.function(my_callback) ">Click here!</button>
```

1.2 API

1.2.1 alert(message)

Alert a message.

- **message:** Any message you want to alert

Usage Example:

```
from dajax.core import Dajax  
  
def alert_example(request):  
    dajax = Dajax()  
    dajax.alert('Hello from python!')  
    return dajax.json()
```

1.2.2 assign(selector, attribute, value)

Assign to all elements that matches with the `selector` as *attribute* the value.

- **selector**: CSS selector.
- **attribute**: Any valid attribute.
- **value**: The value you want to assign.

Usage Example:

```
from dajax.core import Dajax

def assign_example(request):
    dajax = Dajax()
    dajax.assign('#button', 'value', 'Click here!')
    dajax.assign('div .alert', 'innerHTML', 'This email is invalid')
    return dajax.json()
```

1.2.3 add_css_class(selector, value)

Assign to all elements that matches with the `selector` the CSS class value. `value` could be a string or a list of them.

- **selector**: CSS selector.
- **value**: Any CSS class name or a list of them.

Usage Example:

```
from dajax.core import Dajax

def add_css_example(request):
    dajax = Dajax()
    dajax.add_css_class('div .alert', 'red')
    dajax.add_css_class('div .warning', ['big', 'yellow'])
    return dajax.json()
```

1.2.4 remove_css_class(selector, value)

Remove to all elements that matches with the `selector` the CSS class value. `value` could be a string or a list of them.

- **selector**: CSS selector.
- **value**: Any CSS class name or a list of them.

Usage Example:

```
from dajax.core import Dajax

def remove_css_example(request):
    dajax = Dajax()
    dajax.remove_css_class('div .message', 'big-message')
    dajax.remove_css_class('div .total', ['big', 'red'])
    return dajax.json()
```

1.2.5 append(selector, attribute, value)

Append to all elements that matches with the `selector` value to with the desired attribute.

- **selector**: CSS selector.
- **attribute**: Any valid attribute.
- **value**: Any CSS class name or a list of them.

Usage Example:

```
from dajax.core import Dajax

def append_example(request):
    dajax = Dajax()
    dajax.append('#message', 'innerHTML', 'Last message')
    return dajax.json()
```

1.2.6 prepend(selector, attribute, value)

Prepend to all elements that matches with the `selector` value to with the desired `attribute`.

- **selector**: CSS selector.
- **attribute**: Any valid attribute.
- **value**: Any CSS class name or a list of them.

Usage Example:

```
from dajax.core import Dajax

def prepend_example(request):
    dajax = Dajax()
    dajax.prepend('#message', 'innerHTML', 'First message')
    return dajax.json()
```

1.2.7 clear(selector, attribute)

Clear all elements that matches with the `selector` the desired `attribute`.

- **selector**: CSS selector.
- **attribute**: Any valid attribute.

Usage Example:

```
from dajax.core import Dajax

def clear_example(request):
    dajax = Dajax()
    dajax.clear('#message', 'innerHTML')
    return dajax.json()
```

1.2.8 redirect(url, delay=0)

Redirect current page to `url` with a delay of `ms`.

- **url**: Destination URL.
- **delay**: Number of `ms` that the browser should wait before redirecting.

Usage Example:

```
from dajax.core import Dajax

def redirect_example(request):
    dajax = Dajax()
    dajax.redirect('http://google.com', delay=2000)
    return dajax.json()
```

1.2.9 script(code)

Executes code in the browser

- **code**: Code to execute.

Usage Example:

```
from dajax.core import Dajax

def code_example(request):
    dajax = Dajax()
    dajax.code('my_function();')
    return dajax.json()
```

1.2.10 remove(selector)

Remove all elements that matches selector.

- **selector**: CSS selector.

Usage Example:

```
from dajax.core import Dajax

def code_example(request):
    dajax = Dajax()
    dajax.remove('.message')
    return dajax.json()
```

1.2.11 add_data(data, callback_function)

Send data to the browser and call `callback_function` using this data.

- **data**: Data you want to send to your function.
- **callback_function**: Fuction you want to call in the browser.

Usage Example:

```
from dajax.core import Dajax

def data_example(request):
    dajax = Dajax()
    dajax.add_data(range(10), 'my_js_function')
    return dajax.json()
```

1.3 Migrating to 0.9

1.3.1 Static files

Since 0.9 dajax takes advantage of `django.contrib.staticfiles` so deploying a dajax application live is much easier than in previous versions. All the `X.dajax.core.js` flavoured files (jQuery, Prototype, ...) are inside a new folder named `static` instead of `src`.

You need to remember to run `python manage.py collectstatic` before deploying your code live. This command will collect all the static files your application needs into `STATIC_ROOT`. For further information, this is the [Django static files documentation](#)

You should change all your dajax core imports using for example for jQuery:

```
{% static "dajax/jquery.core.js" %}
```

1.3.2 Imports

If you were importing dajax using:

```
from dajax.core.Dajax import Dajax
```

you should change it to:

```
from dajax.core import Dajax
```

1.4 Changelog

1.4.1 0.9.2

- Fix unicode issues
- Fix Internet Explorer issues modifying element's innerHTML

1.4.2 0.9

- Move dajaxice core from `dajaxice.core.Dajax` to `dajax.core`
- Django 1.3 is now a requirement
- dajaxice 0.5 is now a requirement
- Static files are now located inside `static` instead of `src`

1.4.3 0.8.4

- Upgrade to dajaxice 0.1.3 (New `Dajaxice.EXCEPTION`)
- Dajax PEP8 naming style for `addCSSClass` and `removeCSSClass`
- Fixed some bugs in examples
- Fixed unicode problems

1.4.4 0.8.3

- General: New and cleaned setup.py

1.4.5 0.8.2

- General: Upgrade to dajaxice 0.1.1

1.4.6 0.8.0.1

- dajaxice released, now dajax use it as communication core
- cleaned all the code

1.4.7 0.7.5

- Added Dojo support
- Cleaned js files
- Ajax functions outside project folder now supported.
- Flickr in place editor example.

1.4.8 0.7.4.1

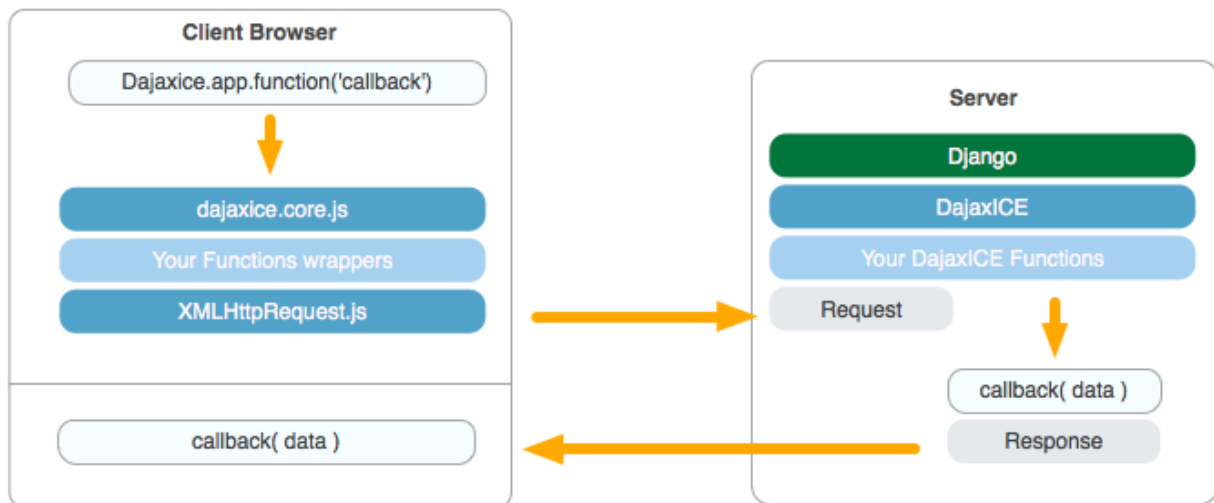
- Typo error importing ajax functions

1.4.9 0.7.4.0

- Typo error importing ajax functions
- Examples: Form validation using new utf-8 support.
- Examples: New deserialize method
- Examples: New DAJAX_CACHE_CONTROL usage in dajax.core.js view.

CHAPTER 2

How does it work?



CHAPTER 3

Example

Once you've installed `dajaxice` and `dajax` you can create ajax functions in your Python code:

```
from dajax.core import Dajax

def assign_test(request):
    dajax = Dajax()
    dajax.assign('#box', 'innerHTML', 'Hello World!')
    dajax.add_css_class('div .alert', 'red')
    return dajax.json()
```

This function will assign to `#box` as `innerHTML` the text `Hello World!` and `Hola!` to every DOM element that matches `.btn`.

You can call this function in your html/js code using:

```
<div onclick="Dajaxice.app.assign_test(Dajax.process);">Click Here!</div>
```

Supported JS Frameworks

Dajax currently support four of the most popular:

- [jQuery 1.7.2](#)
- [Prototype 1.7](#)
- [MooTools 1.4.5](#)
- [Dojo 1.7](#)