

Breaking Changes + Happy Users

Scott González



Breaking Changes

coming soon to a site you're working on

Original Philosophy

- solve everyone's problems
- make every use case as simple as possible



A “Simple” API

```
$( elem ).draggable();
```



A “Simple” API

```
$( elem ).draggable({  
    helper: “clone”  
});
```



A “Simple” API

```
$( elem ).draggable({  
    cursorAt: [ 5, 5 ]  
});
```



A “Simple” API

```
$( elem ).draggable({  
    containment: “#someElem”  
});
```



A “Simple” API

```
$( elem ).draggable({  
    containment: “#someElem”,  
    axis: “x”  
});
```



A “Simple” API

```
$( elem ).draggable({  
    containment: “#someElem”,  
    axis: “x”,  
    grid: [ 20, 50 ]  
});
```



A “Simple” API

```
$( elem ).draggable({  
    containment: “#someElem”,  
    axis: “x”,  
    grid: [ 20, 50 ],  
    cursorAt: [ 5, 5 ],  
    helper: “clone”  
});
```

a road to recovery

Step 1

We admitted we were powerless over feature creep—that our plugins had become unmanageable.

Step 2

Came to believe that an API greater than ours could restore us to sanity.

Step 3

Made a decision to turn our options and our events over to the care of **Simplicity** as we understood it.

Step 4

Made a searching and fearless inventory of our plugins.

Step 5

Admitted to Simplicity, to ourselves, and to our users the exact nature of our wrongs.

Step 6

Were entirely ready to have Simplicity
remove all the cruft of our plugins.

Step 7

Humbly asked *Simplicity* to remove our bloat.

Step 8

Made a list of all plugins we had harmed, and became willing to make amends to them all.

Step 9

Made direct amends to such plugins wherever possible, except when to do so would injure them or others.

Step 10

Continued to take inventory and when we were wrong promptly admitted it.

Step 11

Sought through refactors and redesigns to improve our conscious contact with Simplicity as we understood it, praying only for knowledge of its attainability and the power to carry that out.

Step 12

Having had a philosophical awakening as the result of these steps, we tried to carry this message to developers, and to practice these principles in all our plugins.



keeping users happy

Don't Sacrifice Quality

- design for the future
- deal with design conflicts later

Don't Sacrifice Quality

- 1.8 tabs load event
 - ui.panel is a DOM element
- 1.9 tabs load event
 - ui.panel is a jQuery object
- API CONFLICT!

Make Upgrading Easy

- support old and new APIs simultaneously
- provide an upgrade path
- enable forward-compatibility testing

Best of Both Worlds

- best of new API
 - simpler, more flexible
 - new functionality
- best of old API
 - all existing code uses it

Best of Both Worlds

```
$( elem ).tabs({  
    beforeActivate: function() {  
        // new hotness  
    },  
    select: function() {  
        // old and busted  
    }  
});
```

Upgrade Guide

- show how to upgrade everything
 - explaining the why helps too
- optionally build back-compatible extensions

Done is Done

- provide a way to disable deprecated code
 - alternatively log warnings
- allow users to test for forward-compat

Done is Done

```
<script src="jquery.js"></script>
<script>
    $.uiBackCompat = false;
</script>
<script src="jquery-ui.js"></script>

<!-- new hotness only :- ) -->
```

Thank You

<http://spkr8.com/t/8506>

@scott_gonzalez

scottgonzalez.com