

# Step-by-Step Guide: Fixing Django Ticket #35924

This guide demonstrates the complete process of fixing Django ticket #35924 (FilteredSelectMultiple widget choose/remove all button arrows not responsive). We'll walk through each step in detail, from setting up the environment to submitting a pull request.

## Step 1: Setting Up the Development Environment

First, let's set up our Django development environment:

```
# Fork the Django repository on GitHub (via the web interface)

# Clone your fork locally
git clone https://github.com/YOUR-USERNAME/django.git
cd django

# Add the upstream remote
git remote add upstream https://github.com/django/django.git

# Create and activate a virtual environment
python -m venv django-dev
source django-dev/bin/activate  # On Windows: django-dev\Scripts\activate

# Install Django in development mode
pip install -e .

# Install testing requirements
pip install -r requirements/testing.txt
```

## Step 2: Understanding the Issue

Let's examine ticket #35924 in detail:

The issue is with the FilteredSelectMultiple widget in Django's admin interface. This widget has buttons to move items between two select boxes. When viewed on mobile devices, the individual item selection arrows correctly change from left/right to up/down, but the "Choose all" and "Remove all" buttons don't adapt their arrows.

## Step 3: Locating the Relevant Code

Let's find the code we need to modify:

```
# Search for FilteredSelectMultiple in the codebase
grep -r "FilteredSelectMultiple" django/
```

Key files we need to examine:

1. [django/contrib/admin/widgets.py](#) - Contains the widget definition
2. [django/contrib/admin/static/admin/js>SelectFilter2.js](#) - JavaScript functionality
3. [django/contrib/admin/static/admin/css/widgets.css](#) - CSS styling

Let's look at the CSS file first to understand how the arrows are currently styled:

```
/* In django/contrib/admin/static/admin/css/widgets.css */

/* Current styling for the arrows */
.selector .selector-available h2 {
    background: var(--primary);
    color: var(--header-link-color);
}

.selector .selector-available h2, .selector .selector-chosen h2 {
    border: 1px solid var(--border-color);
    border-radius: 4px 4px 0 0;
}

.selector .selector-filter {
    background: white;
    border: 1px solid var(--border-color);
    border-width: 0 1px;
    padding: 8px;
}

.selector .selector-chosen .selector-filter {
    padding: 8px;
    background: var(--darkened-bg);
    border: 1px solid var(--border-color);
    border-width: 0 1px;
}

/* Current styling for the choose/remove all buttons */
.selector .selector-chooseall, .selector .selector-clearall {
    display: inline-block;
    height: 16px;
    text-align: left;
    margin: 6px auto 3px;
    font-weight: bold;
    color: var(--body-quiet-color);
    text-decoration: none;
    padding: 0 18px 0 0;
    border-radius: 5px;
}

/* The arrows are defined using pseudo-elements */
.selector .selector-chooseall:before, .selector .selector-clearall:before {
    content: "\00BB"; /* Right-pointing double angle quotation mark */
    font-weight: bold;
    padding: 0 5px;
}
```

```
.selector .selector-clearall:after {
    content: "\00BB"; /* Right-pointing double angle quotation mark */
    font-weight: bold;
    padding: 0 5px;
}

/* Mobile view adjustments for individual arrows */
@media (max-width: 767px) {
    .selector > table {
        display: flex;
        flex-direction: column;
    }

    .selector > table > tbody, .selector > table > tbody > tr {
        display: flex;
        flex-direction: column;
    }

    .selector > table > tbody > tr > * {
        display: block;
        width: auto;
        max-width: none;
    }
}

.selector .selector-available, .selector .selector-chosen {
    width: auto;
}

.selector .selector-available {
    margin-bottom: 10px;
}

.selector .selector-add, .selector .selector-remove {
    width: 8em;
    margin: 0 auto 10px;
    display: block;
}

/* Here's where the individual arrows are changed for mobile */
.selector .selector-add {
    background-position: 1em center;
}

.selector .selector-remove {
    background-position: 1em center;
}

/* But there's no equivalent for the choose/remove all buttons */
}
```

Now let's check the JavaScript file to understand how the buttons are created:

```
// In django/contrib/admin/static/admin/js/SelectFilter2.js

// This is how the choose all and remove all buttons are created
quickElement(
  'a',
  quickElement('div', selector_chosen_box, 'selector-chooseall'),
  gettext('Choose all'),
  'href', 'javascript:void(0);',
  'title', interpolate(gettext('Choose all of the %s at once.'),
  [field_name]),
  'id', field_id + '_add_all_link'
);

quickElement(
  'a',
  quickElement('div', selector_chosen_box, 'selector-clearall'),
  gettext('Remove all'),
  'href', 'javascript:void(0);',
  'title', interpolate(gettext('Remove all chosen %s at once.'),
  [field_name]),
  'id', field_id + '_remove_all_link'
);
```

## Step 4: Creating a Development Branch

Let's create a branch for our fix:

```
# Make sure we're on the main branch and up-to-date
git checkout main
git pull upstream main

# Create a new branch
git checkout -b fix-35924-filteredselectmultiple-arrows
```

## Step 5: Implementing the Fix

Now, let's modify the CSS to make the arrows responsive. We'll add CSS rules to change the arrow direction on mobile devices:

```
/* Add this to django/contrib/admin/static/admin/css/widgets.css */

@media (max-width: 767px) {
  /* Existing mobile styles... */

  /* New styles for choose/remove all buttons */
  .selector .selector-chooseall:before {
    content: "\2191"; /* Up arrow */
  }
}
```

```
.selector .selector-clearall:after {  
    content: "\2193"; /* Down arrow */  
}  
}
```

Here's the complete diff for our changes:

```
--- a/django/contrib/admin/static/admin/css/widgets.css  
+++ b/django/contrib/admin/static/admin/css/widgets.css  
@@ -178,6 +178,14 @@  
     .selector .selector-remove {  
         background-position: 1em center;  
     }  
+/* Change arrow direction for choose all/remove all buttons on mobile */  
+    .selector .selector-chooseall:before {  
+        content: "\2191"; /* Up arrow */  
+    }  
+    .selector .selector-clearall:after {  
+        content: "\2193"; /* Down arrow */  
+    }  
/* STACKED WIDGETS */
```

## Step 6: Testing Our Changes

Let's test our changes to ensure they work correctly:

```
# Run the specific tests for admin widgets  
python tests/runtests.py admin_widgets
```

For this UI change, we also need to manually test it. Let's create a test project:

```
# Create a test project  
mkdir -p /tmp/django-test  
cd /tmp/django-test  
django-admin startproject testproject  
cd testproject  
  
# Create a test app  
python manage.py startapp testapp
```

Now, let's create a model with a ManyToManyField to test the widget:

```
# In testapp/models.py
from django.db import models

class Category(models.Model):
    name = models.CharField(max_length=100)

    def __str__(self):
        return self.name

class Article(models.Model):
    title = models.CharField(max_length=200)
    categories = models.ManyToManyField(Category)

    def __str__(self):
        return self.title
```

Register the models in the admin:

```
# In testapp/admin.py
from django.contrib import admin
from .models import Category, Article

admin.site.register(Category)
admin.site.register(Article)
```

Update settings:

```
# In testproject/settings.py
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'testapp',
]
```

Create and apply migrations:

```
python manage.py makemigrations
python manage.py migrate
python manage.py createsuperuser
```

Run the development server:

```
python manage.py runserver
```

Now we can test our changes by:

1. Opening the admin site at <http://127.0.0.1:8000/admin/>
2. Going to Articles > Add Article
3. Testing the categories widget in both desktop and mobile views (using browser developer tools to simulate mobile)
4. Verifying that the arrows in the "Choose all" and "Remove all" buttons change direction in mobile view

## Step 7: Committing Our Changes

Once we've confirmed our fix works, let's commit the changes:

```
# Go back to our Django repository
cd /path/to/django/repo

# Add the modified file
git add django/contrib/admin/static/admin/css/widgets.css

# Commit with a descriptive message
git commit -m "Fixed #35924: Made FilteredSelectMultiple choose/remove all button arrows responsive on mobile"
```

## Step 8: Writing Tests

For UI changes like this, we should consider adding JavaScript tests. Django uses QUnit for JavaScript testing:

```
// In js_tests/admin/SelectFilter2.test.js (create if it doesn't exist)

QUnit.module('admin.SelectFilter2');

QUnit.test('Arrow direction changes on mobile view', function(assert) {
    // Set up a mock environment
    const fixture = document.getElementById('qunit-fixture');

    // Create a FilteredSelectMultiple widget
    // ... (setup code)

    // Test desktop view
    assert.equal(
        window.getComputedStyle(chooseAllButton, ':before').content,
        '\u21ba',
        'Choose all button has right arrow on desktop'
```

```
);

    // Simulate mobile view
    // ... (code to simulate mobile view)

    // Test mobile view
    assert.equal(
        window.getComputedStyle(chooseAllButton, ':before').content,
        '\u2191',
        'Choose all button has up arrow on mobile'
    );
};

});
```

However, since this is a CSS-only change and the functionality remains the same, we might not need additional tests beyond the existing ones.

## Step 9: Pushing Changes and Creating a Pull Request

Let's push our changes to GitHub:

```
git push origin fix-35924-filteredselectmultiple-arrows
```

Now, go to your fork on GitHub and create a pull request:

1. Click "Pull Request"
2. Set the base repository to django/django and the base branch to main
3. Set your repository and branch as the compare
4. Add a descriptive title: "Fixed #35924: Made FilteredSelectMultiple choose/remove all button arrows responsive"
5. Add a detailed description:

```
The arrow buttons to choose or remove selected items correctly change direction when in a mobile view to be up/down rather than left/right. However, in the choose all/remove all buttons the arrow icons were always left/right.
```

```
This PR fixes the issue by making the arrows in the choose all/remove all buttons responsive to match the behavior of the individual item buttons.
```

```
Fixes #35924
```

6. Submit the pull request

## Step 10: Responding to Review Feedback

After submitting your PR, Django maintainers will review it. They might suggest changes:

Reviewer: "Thanks for the PR! Could you also add a comment in the CSS explaining why these arrows need to be different on mobile?"

Let's make the requested change:

```
# Edit the file again  
vim django/contrib/admin/static/admin/css/widgets.css  
  
# Add a comment explaining the change
```

```
--- a/django/contrib/admin/static/admin/css/widgets.css  
+++ b/django/contrib/admin/static/admin/css/widgets.css  
@@ -178,6 +178,7 @@  
     background-position: 1em center;  
 }  
  
+ /* Change arrow direction to match the vertical layout on mobile  
devices */  
/* Change arrow direction for choose all/remove all buttons on mobile  
*/  
.selector .selector-chooseall:before {  
    content: "\2191"; /* Up arrow */
```

Commit and push the changes:

```
git add django/contrib/admin/static/admin/css/widgets.css  
git commit -m "Added explanatory comment for mobile arrow direction"  
git push origin fix-35924-filteredselectmultiple-arrows
```

The PR will be automatically updated with your new commit.

## Step 11: Final Acceptance and Merge

Once the reviewers are satisfied with your changes, they will approve the PR and merge it into Django's main branch. Congratulations! You've successfully contributed to Django.

## Key Learnings from This Process

- 1. Understanding the Issue:** Taking time to fully understand the problem before diving into code
- 2. Locating Relevant Code:** Using grep and code exploration to find where changes are needed
- 3. Making Minimal Changes:** Fixing the issue with the smallest necessary change
- 4. Testing Thoroughly:** Running automated tests and manual testing
- 5. Clear Communication:** Writing clear commit messages and PR descriptions

6. **Responding to Feedback:** Being receptive to reviewer suggestions

7. **Documentation:** Adding comments to explain the changes

This process demonstrates how even a small UI fix follows the same structured contribution workflow as larger changes. By starting with beginner-friendly tickets like this one, you can become familiar with the contribution process before tackling more complex issues.