In [1]:
```python
## Import Tesnorflow and Keras from Tensorflow ##
import tensorflow as tf
from tensorflow import keras
```

In [2]:
```python
## import other libraries as well
import numpy as np
import matplotlib.pyplot as plt
```

In [3]:
```python
## imoprt the datasets from Google
data = keras.datasets.fashion_mnist
```

In [4]:
```python
## Split the data sets into train and test

(train_images,train_labels), (test_images,test_labels) = data.load_data()
```

In [5]:
```python
## Show the Dimesnsions of train image
train_images.shape
```

Out[5]:
```
(60000, 28, 28)
```

In [6]:
```python
len(train_labels)
```

Out[6]:
```
60000
```

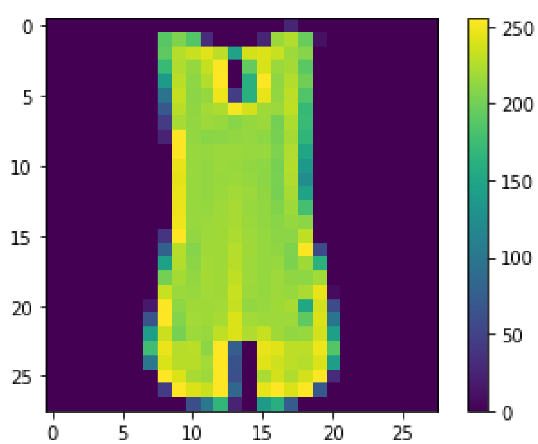
In [7]:
```python
train_labels
```

Out[7]:
```
array([9, 0, 0, ..., 3, 0, 5], dtype=uint8)
```

In [8]:
```python
test_images.shape
```

Out[8]:
```
(10000, 28, 28)
```

In [9]:
```python
## Show the actual image how it look like from the data set
plt.figure()
plt.imshow(train_images[4])
plt.colorbar()
plt.grid(False)
plt.show()
```



In [10]:

```
## Scale these values to a range of 0 to 1 before feeding them to the neural network model.
train_images = train_images / 255.0

test_images = test_images / 255.0
```

In [11]:

```
print(train_labels[0])
```

```
9
```

In [12]:
```
## It shows the IMage value in RGB format which is (28,28) matrix represenatation of an ima
print(train_images[0])
```

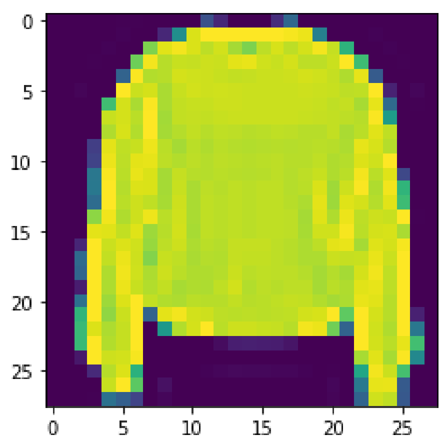
```
[[0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.00392157 0.         0.         0.05098039 0.28627451 0.
  0.         0.00392157 0.01568627 0.         0.         0.
  0.         0.00392157 0.00392157 0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.01176471 0.         0.14117647 0.53333333 0.49803922 0.24313725
  0.21176471 0.         0.         0.         0.00392157 0.01176471
  0.01568627 0.         0.         0.01176471]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.02352941 0.         0.4        0.8        0.69019608 0.5254902
  0.56470588 0.48235294 0.09019608 0.         0.         0.
  0.         0.04705882 0.03921569 0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.60784314 0.9254902  0.81176471 0.69803922
  0.41960784 0.61176471 0.63137255 0.42745098 0.25098039 0.09019608
  0.30196078 0.50980392 0.28235294 0.05882353]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.00392157
  0.         0.27058824 0.81176471 0.8745098  0.85490196 0.84705882
  0.84705882 0.63921569 0.49803922 0.4745098  0.47843137 0.57254902
  0.55294118 0.34509804 0.6745098  0.25882353]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.00392157 0.00392157 0.00392157
  0.         0.78431373 0.90980392 0.90980392 0.91372549 0.89803922
  0.8745098  0.8745098  0.84313725 0.83529412 0.64313725 0.49803922
  0.48235294 0.76862745 0.89803922 0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.71764706 0.88235294 0.84705882 0.8745098  0.89411765
  0.92156863 0.89019608 0.87843137 0.87058824 0.87843137 0.86666667
  0.8745098  0.96078431 0.67843137 0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.75686275 0.89411765 0.85490196 0.83529412 0.77647059
  0.70588235 0.83137255 0.82352941 0.82745098 0.83529412 0.8745098
  0.8627451  0.95294118 0.79215686 0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.00392157 0.01176471 0.
  0.04705882 0.85882353 0.8627451  0.83137255 0.85490196 0.75294118
  0.6627451  0.89019608 0.81568627 0.85490196 0.87843137 0.83137255
  0.88627451 0.77254902 0.81960784 0.20392157]
```

```
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.02352941 0.
 0.38823529 0.95686275 0.87058824 0.8627451  0.85490196 0.79607843
 0.77647059 0.86666667 0.84313725 0.83529412 0.87058824 0.8627451
 0.96078431 0.46666667 0.65490196 0.21960784]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.01568627 0.         0.
 0.21568627 0.9254902  0.89411765 0.90196078 0.89411765 0.94117647
 0.90980392 0.83529412 0.85490196 0.8745098  0.91764706 0.85098039
 0.85098039 0.81960784 0.36078431 0.        ]
[0.         0.         0.00392157 0.01568627 0.02352941 0.02745098
 0.00784314 0.         0.         0.         0.         0.
 0.92941176 0.88627451 0.85098039 0.8745098  0.87058824 0.85882353
 0.87058824 0.86666667 0.84705882 0.8745098  0.89803922 0.84313725
 0.85490196 1.         0.30196078 0.        ]
[0.         0.01176471 0.         0.         0.         0.
 0.         0.         0.         0.24313725 0.56862745 0.8
 0.89411765 0.81176471 0.83529412 0.86666667 0.85490196 0.81568627
 0.82745098 0.85490196 0.87843137 0.8745098  0.85882353 0.84313725
 0.87843137 0.95686275 0.62352941 0.        ]
[0.         0.         0.         0.         0.07058824 0.17254902
 0.32156863 0.41960784 0.74117647 0.89411765 0.8627451  0.87058824
 0.85098039 0.88627451 0.78431373 0.80392157 0.82745098 0.90196078
 0.87843137 0.91764706 0.69019608 0.7372549  0.98039216 0.97254902
 0.91372549 0.93333333 0.84313725 0.        ]
[0.         0.22352941 0.73333333 0.81568627 0.87843137 0.86666667
 0.87843137 0.81568627 0.8        0.83921569 0.81568627 0.81960784
 0.78431373 0.62352941 0.96078431 0.75686275 0.80784314 0.8745098
 1.         1.         0.86666667 0.91764706 0.86666667 0.82745098
 0.8627451  0.90980392 0.96470588 0.        ]
[0.01176471 0.79215686 0.89411765 0.87843137 0.86666667 0.82745098
 0.82745098 0.83921569 0.80392157 0.80392157 0.80392157 0.8627451
 0.94117647 0.31372549 0.58823529 1.         0.89803922 0.86666667
 0.7372549  0.60392157 0.74901961 0.82352941 0.8        0.81960784
 0.87058824 0.89411765 0.88235294 0.        ]
[0.38431373 0.91372549 0.77647059 0.82352941 0.87058824 0.89803922
 0.89803922 0.91764706 0.97647059 0.8627451  0.76078431 0.84313725
 0.85098039 0.94509804 0.25490196 0.28627451 0.41568627 0.45882353
 0.65882353 0.85882353 0.86666667 0.84313725 0.85098039 0.8745098
 0.8745098  0.87843137 0.89803922 0.11372549]
[0.29411765 0.8        0.83137255 0.8        0.75686275 0.80392157
 0.82745098 0.88235294 0.84705882 0.7254902  0.77254902 0.80784314
 0.77647059 0.83529412 0.94117647 0.76470588 0.89019608 0.96078431
 0.9372549  0.8745098  0.85490196 0.83137255 0.81960784 0.87058824
 0.8627451  0.86666667 0.90196078 0.2627451 ]
[0.18823529 0.79607843 0.71764706 0.76078431 0.83529412 0.77254902
 0.7254902  0.74509804 0.76078431 0.75294118 0.79215686 0.83921569
 0.85882353 0.86666667 0.8627451  0.9254902  0.88235294 0.84705882
 0.78039216 0.80784314 0.72941176 0.70980392 0.69411765 0.6745098
 0.70980392 0.80392157 0.80784314 0.45098039]
[0.         0.47843137 0.85882353 0.75686275 0.70196078 0.67058824
 0.71764706 0.76862745 0.8        0.82352941 0.83529412 0.81176471
 0.82745098 0.82352941 0.78431373 0.76862745 0.76078431 0.74901961
 0.76470588 0.74901961 0.77647059 0.75294118 0.69019608 0.61176471
 0.65490196 0.69411765 0.82352941 0.36078431]
[0.         0.         0.29019608 0.74117647 0.83137255 0.74901961
 0.68627451 0.6745098  0.68627451 0.70980392 0.7254902  0.7372549
 0.74117647 0.7372549  0.75686275 0.77647059 0.8        0.81960784
 0.82352941 0.82352941 0.82745098 0.7372549  0.7372549  0.76078431
 0.75294118 0.84705882 0.66666667 0.        ]
[0.00784314 0.         0.         0.         0.25882353 0.78431373
 0.87058824 0.92941176 0.9372549  0.94901961 0.96470588 0.95294118
 0.95686275 0.86666667 0.8627451  0.75686275 0.74901961 0.70196078
 0.71372549 0.71372549 0.70980392 0.69019608 0.65098039 0.65882353
```

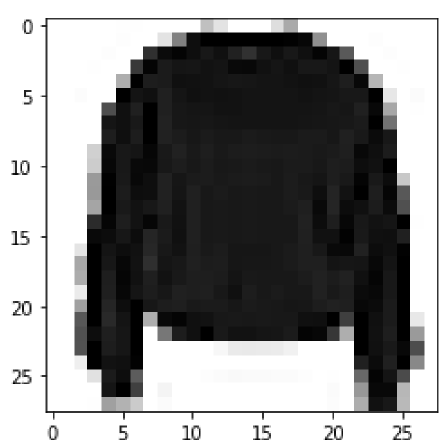
```
  0.38823529 0.22745098 0.         0.         ]
 [0.         0.         0.         0.         0.         0.
  0.         0.15686275 0.23921569 0.17254902 0.28235294 0.16078431
  0.1372549  0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         ]]
```

In [13]:
```
## Define the class labes as an when required for classification
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
```

In [14]:
```
plt.imshow(train_images[7])
plt.show()
```



In [15]:
```
plt.imshow(train_images[7], cmap=plt.cm.binary)
plt.show()
```



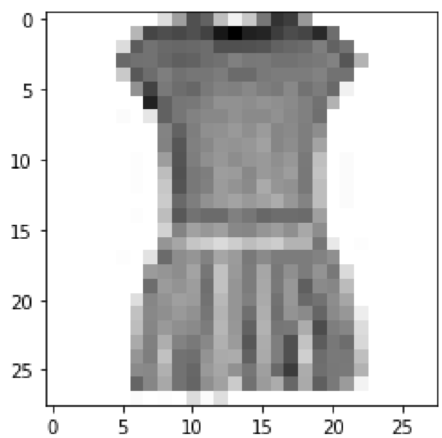
In [16]:
```
print(train_labels[3])
```

```
3
```

In [17]:
```
print(train_images[3])
```

```
 0.46666667 0.43137255 0.45882353 0.45882353 0.43137255 0.46666667
 0.49803922 0.56470588 0.         0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.48235294 0.61176471 0.50588235 0.43921569
 0.43137255 0.4        0.43921569 0.39215686 0.4745098  0.45882353
 0.50588235 0.44705882 0.         0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.49019608 0.6627451  0.49803922 0.46666667
 0.41568627 0.42352941 0.40784314 0.36862745 0.4745098  0.44705882
 0.50588235 0.35686275 0.         0.         0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
 0.00784314 0.         0.38431373 0.67058824 0.50588235 0.43921569
 0.40784314 0.44705882 0.41568627 0.4        0.43921569 0.40784314
 0.52156863 0.25098039 0.         0.01568627 0.         0.
 0.         0.         0.         0.        ]
[0.         0.         0.         0.         0.         0.
```

In [18]:
```
plt.imshow(train_images[3], cmap=plt.cm.binary)
plt.show()
```



In [19]:
```
## let's display the first 25 images from the training set
plt.figure(figsize=(12,12))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[train_labels[i]])
plt.show()
```

In [20]:

```
## Flatten : transforms the format of the images from a two-dimensional array (28*28) to a
## The first Dense layer has 128 nodes or neurons
## he second Dense layer is a 10-node softmax layer that returns an array of 10 probability

model = keras.Sequential([keras.layers.Flatten(input_shape=(28,28)),
                          keras.layers.Dense(128,activation="relu"),
                          keras.layers.Dense(10,activation="softmax")])
```

```
WARNING:tensorflow:From C:\Users\nilesh\Anaconda3\lib\site-packages\tensorflow\python\ops
\init_ops.py:1251: calling VarianceScaling.__init__ (from tensorflow.python.ops.init_ops)
with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
```

In [21]:

```
## Loss function —This measures how accurate the model is during training
## Optimizer —This is how the model is updated based on the data it sees and its loss funct
## Metrics —Used to monitor the training and testing steps
## accuracy- The fraction of the images that are correctly classified.

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

In [22]:
```
## Training the neural network model
## 1. Feed the training data to the model, images with its labels
## 2. model learns by associate images and it's labels
## 3. You ask the model to make predictions, test_images array.
## 4. Verify that the predictions match the labels from the test_labels array.

model.fit(train_images, train_labels, epochs=10)
```

```
Epoch 1/10
60000/60000 [==============================] - 5s 91us/sample - loss: 0.4983 - acc: 0.8248
Epoch 2/10
60000/60000 [==============================] - 5s 80us/sample - loss: 0.3738 - acc: 0.8648
Epoch 3/10
60000/60000 [==============================] - 5s 80us/sample - loss: 0.3346 - acc: 0.8781
Epoch 4/10
60000/60000 [==============================] - 5s 87us/sample - loss: 0.3114 - acc: 0.8861
Epoch 5/10
60000/60000 [==============================] - 5s 80us/sample - loss: 0.2942 - acc: 0.8917
Epoch 6/10
60000/60000 [==============================] - 5s 80us/sample - loss: 0.2789 - acc: 0.8967
Epoch 7/10
60000/60000 [==============================] - 5s 83us/sample - loss: 0.2688 - acc: 0.8998
Epoch 8/10
60000/60000 [==============================] - 5s 83us/sample - loss: 0.2566 - acc: 0.9051
Epoch 9/10
60000/60000 [==============================] - 5s 79us/sample - loss: 0.2480 - acc: 0.9085
Epoch 10/10
60000/60000 [==============================] - 5s 81us/sample - loss: 0.2399 - acc: 0.9107
```

Out[22]:
```
<tensorflow.python.keras.callbacks.History at 0xcf35bb9550>
```

In [23]:
```
## compare how the model performs on the test dataset (unseen data,New data)

test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)

print('\nTest accuracy:', test_acc)
```

```
10000/10000 - 0s - loss: 0.3269 - acc: 0.8849

Test accuracy: 0.8849
```

In [24]:
```
## With the model trained, you can use it to make predictions about some images.
predictions = model.predict(test_images)
```

In [25]:
```
predictions[0]
```

Out[25]:
```
array([2.4247992e-08, 4.5656640e-10, 4.2555897e-08, 1.1614337e-08,
       1.0836500e-07, 5.1758192e-03, 3.7873686e-08, 1.3519106e-02,
       1.6351271e-07, 9.8130476e-01], dtype=float32)
```

In [26]:
```
## A prediction is an array of 10 numbers. They represent the model's "confidence" that the
## each of the 10 different articles of clothing.
predictions[1]
```

Out[26]:
```
array([1.7410640e-04, 7.7553193e-11, 9.9636251e-01, 3.0907454e-09,
       1.1336243e-03, 3.2112167e-11, 2.3298159e-03, 1.8075490e-14,
       2.2551484e-08, 5.8282356e-13], dtype=float32)
```

In [27]:
```
np.argmax(predictions[0])
```

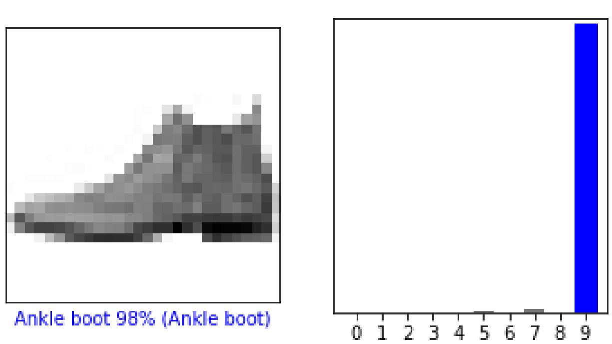
Out[27]:
```
9
```

In [28]:
```python
np.argmax(predictions[1])
```
Out[28]: 2

In [29]:
```python
test_labels[0]
```
Out[29]: 9

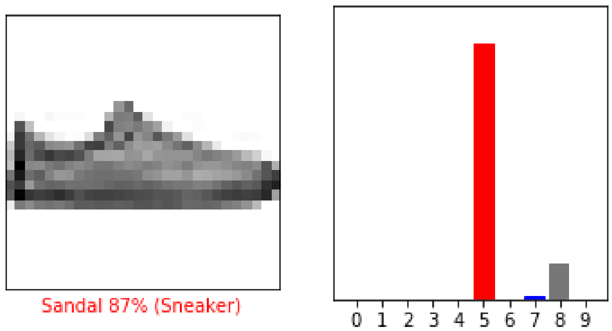
In [30]:
```python
### Graph this to look at the full set of 10 class predictions. ###

def plot_image(i, predictions_array, true_label, img):
  predictions_array, true_label, img = predictions_array, true_label[i], img[i]
  plt.grid(False)
  plt.xticks([])
  plt.yticks([])

  plt.imshow(img, cmap=plt.cm.binary)

  predicted_label = np.argmax(predictions_array)
  if predicted_label == true_label:
    color = 'blue'
  else:
    color = 'red'

  plt.xlabel("{} {:2.0f}% ({})".format(class_names[predicted_label],
                                100*np.max(predictions_array),
                                class_names[true_label]),
                                color=color)

def plot_value_array(i, predictions_array, true_label):
  predictions_array, true_label = predictions_array, true_label[i]
  plt.grid(False)
  plt.xticks(range(10))
  plt.yticks([])
  thisplot = plt.bar(range(10), predictions_array, color="#777777")
  plt.ylim([0, 1])
  predicted_label = np.argmax(predictions_array)

  thisplot[predicted_label].set_color('red')
  thisplot[true_label].set_color('blue')
```

In [31]:
```python
## With the model trained, you can use it to make predictions about some images ###
i = 0
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions[i], test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions[i],  test_labels)
plt.show()
```

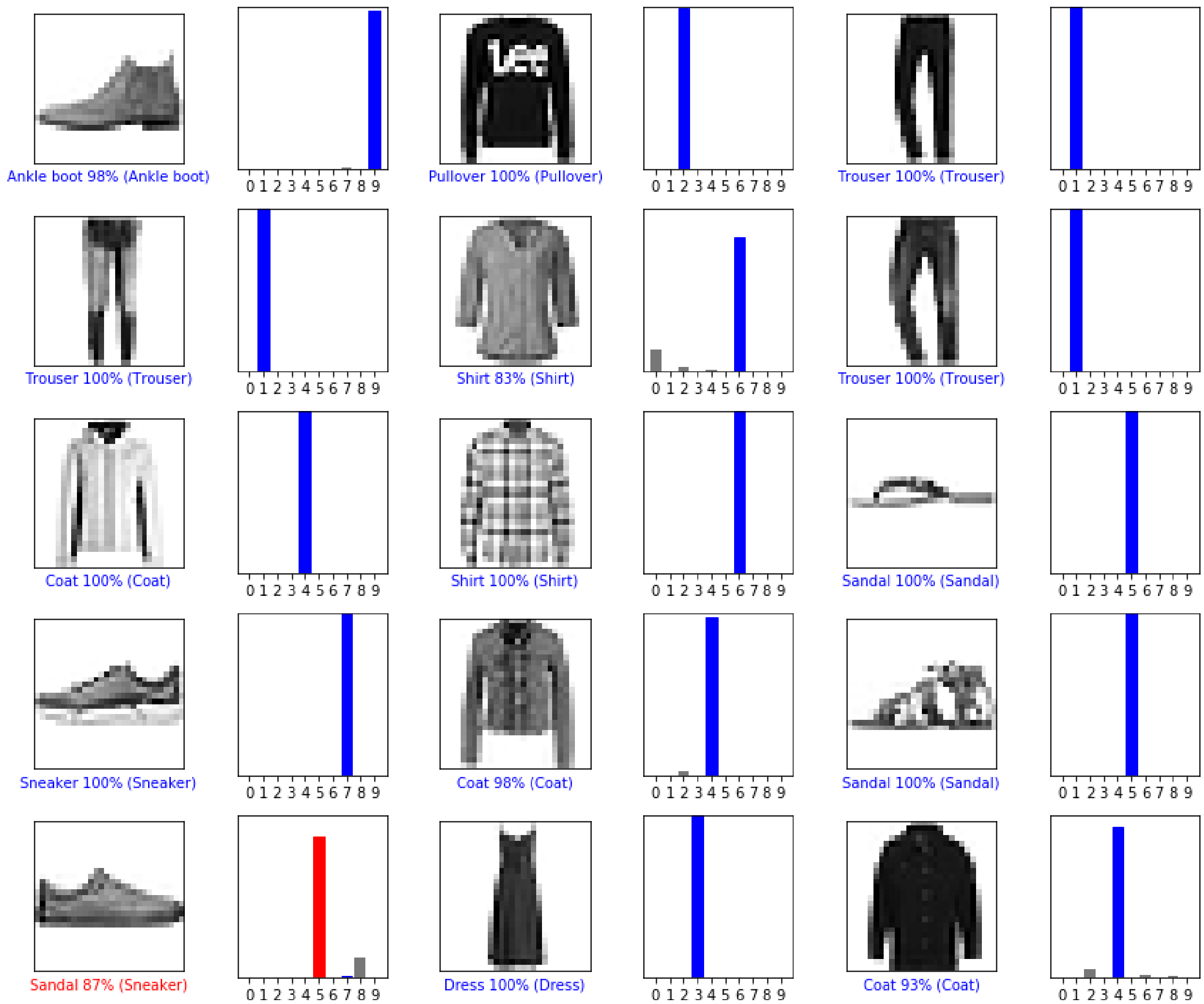


In [32]:

```
i = 12
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions[i], test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions[i],  test_labels)
plt.show()
```



In [33]:
```
# Plot the first X test images, their predicted labels, and the true labels.
# Color correct predictions in blue and incorrect predictions in red.
num_rows = 5
num_cols = 3
num_images = num_rows*num_cols
plt.figure(figsize=(2*2*num_cols, 2*num_rows))
for i in range(num_images):
  plt.subplot(num_rows, 2*num_cols, 2*i+1)
  plot_image(i, predictions[i], test_labels, test_images)
  plt.subplot(num_rows, 2*num_cols, 2*i+2)
  plot_value_array(i, predictions[i], test_labels)
plt.tight_layout()
plt.show()
```



In [34]:
```
# Grab an image from the test dataset.
img = test_images[1]

print(img.shape)
```

```
(28, 28)
```

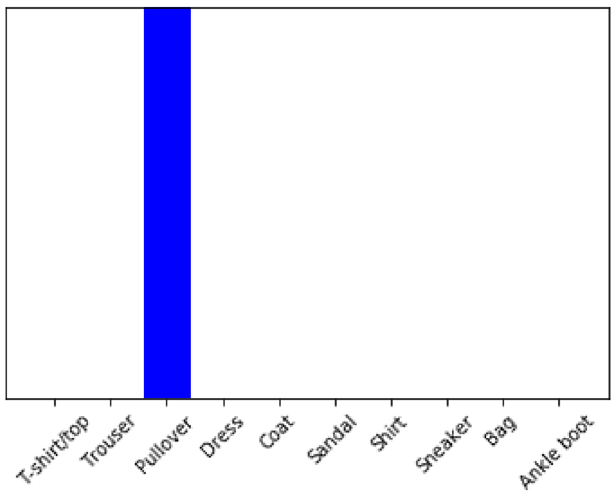
In [35]:
```
# Add the image to a batch where it's the only member.
img = (np.expand_dims(img,0))

print(img.shape)
```

```
(1, 28, 28)
```

In [36]:
```
predictions_single = model.predict(img)

print(predictions_single)
```

```
[[1.7410658e-04 7.7552895e-11 9.9636251e-01 3.0907632e-09 1.1336260e-03
  3.2112167e-11 2.3298236e-03 1.8075490e-14 2.2551571e-08 5.8282020e-13]]
```

In [37]:
```
plot_value_array(1, predictions_single[0], test_labels)
_ = plt.xticks(range(10), class_names, rotation=45)
```



In [38]:
```
## model predicts a label as expected
np.argmax(predictions_single[0])
```

Out[38]: 2