In [1]:
```python
# TensorFlow and tf.keras
import tensorflow as tf

# Helper libraries
import numpy as np
import matplotlib.pyplot as plt

print(tf.__version__)
```
```
2.4.0
```

In [2]:
```python
#This guide uses the Fashion MNIST dataset which contains
#70,000 grayscale images in 10 categories.
#The images show individual articles of clothing at low resolution  (28 by 28 pixels), as seen here:
```

In [3]:
```python
fashion_mnist = tf.keras.datasets.fashion_mnist

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

In [4]:
```python
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
```

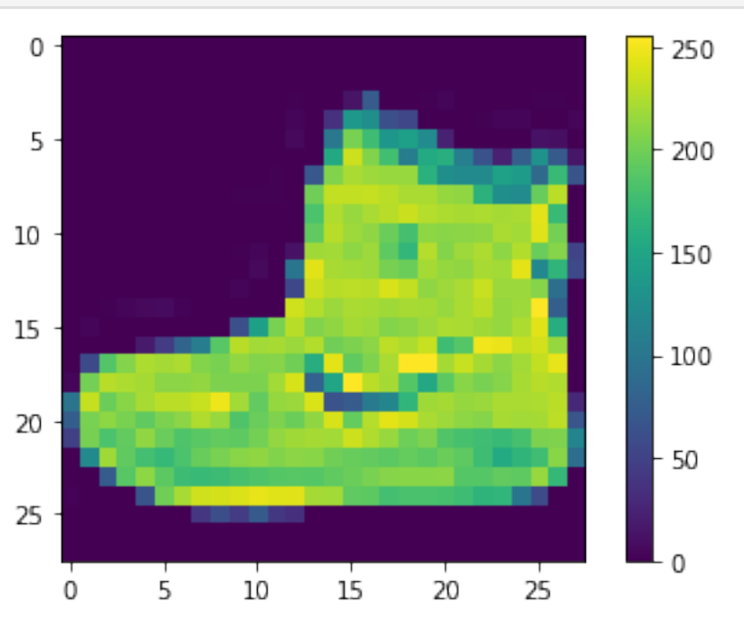
In [5]:
```python
train_images.shape
```
Out[5]:
```
(60000, 28, 28)
```

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
plt.figure()
plt.imshow(train_images[0])
plt.colorbar()
plt.grid(False)
plt.show()
```



In [10]:
```python
train_images = train_images / 255.0

test_images = test_images / 255.0
```

In [11]:
```python
plt.figure(figsize=(10,10))
for i in range(16):
    plt.subplot(4,4,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]:
```python
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(10)
])
```

In [21]:
```python
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

In [22]:
```python
model.fit(train_images, train_labels, epochs=10)
```
```
Epoch 1/10
1875/1875 [==============================] - 2s 756us/step - loss: 0.6318 - accuracy: 0.7788
Epoch 2/10
1875/1875 [==============================] - 1s 746us/step - loss: 0.3917 - accuracy: 0.8603
Epoch 3/10
1875/1875 [==============================] - 1s 744us/step - loss: 0.3414 - accuracy: 0.8756
Epoch 4/10
1875/1875 [==============================] - 1s 769us/step - loss: 0.3138 - accuracy: 0.8869
Epoch 5/10
1875/1875 [==============================] - 1s 752us/step - loss: 0.2929 - accuracy: 0.8926
Epoch 6/10
1875/1875 [==============================] - 1s 766us/step - loss: 0.2774 - accuracy: 0.8977
Epoch 7/10
1875/1875 [==============================] - 1s 747us/step - loss: 0.2660 - accuracy: 0.9021
Epoch 8/10
1875/1875 [==============================] - 2s 804us/step - loss: 0.2522 - accuracy: 0.9070
Epoch 9/10
1875/1875 [==============================] - 1s 746us/step - loss: 0.2520 - accuracy: 0.9059
Epoch 10/10
1875/1875 [==============================] - 1s 780us/step - loss: 0.2421 - accuracy: 0.9105
```
Out[22]:
```
<tensorflow.python.keras.callbacks.History at 0x179a78a8b20>
```

In [24]:
```python
test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)

print('\nTest accuracy:', test_acc)
```
```
313/313 - 0s - loss: 0.3378 - accuracy: 0.8825

Test accuracy: 0.8824999928474426
```

In [25]:
```python
probability_model = tf.keras.Sequential([model, tf.keras.layers.Softmax()])
```

In [26]:
```python
predictions = probability_model.predict(test_images)
```

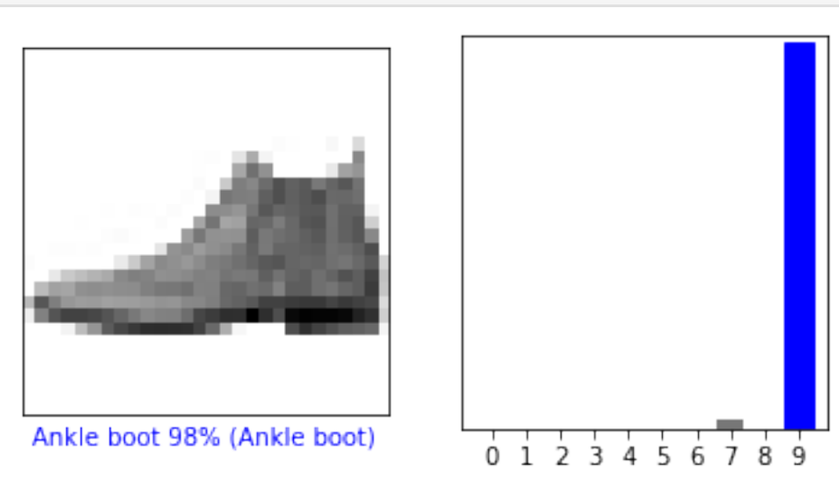
In [28]:
```python
predictions[0]
```
Out[28]:
```
array([3.6937859e-08, 1.2508876e-09, 2.4563775e-07, 1.7621596e-07,
       3.5072011e-07, 1.1662538e-03, 7.7399413e-08, 1.0616244e-02,
       6.6029742e-07, 9.8821592e-01], dtype=float32)
```

In [29]:
```python
np.argmax(predictions[0])
```
Out[29]:
```
9
```

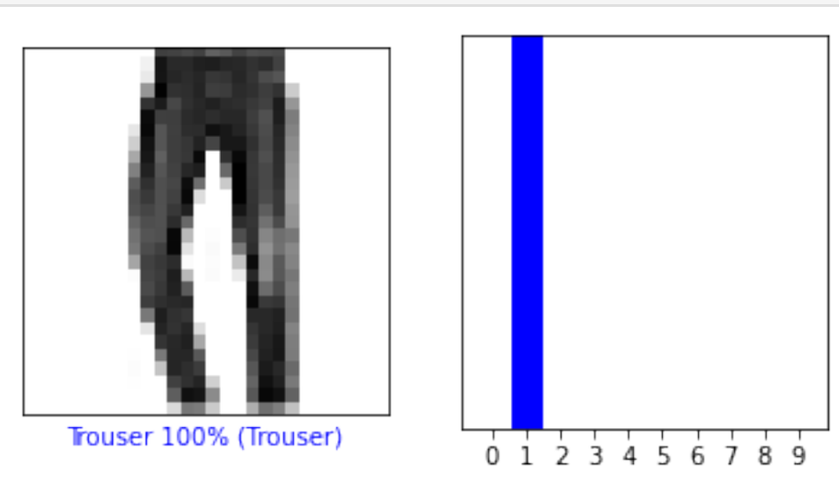
In [30]:
```python
test_labels[0]
```
Out[30]:
```
9
```

In [31]:
```python
def plot_image(i, predictions_array, true_label, img):
  true_label, img = 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):
  true_label = 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')
```

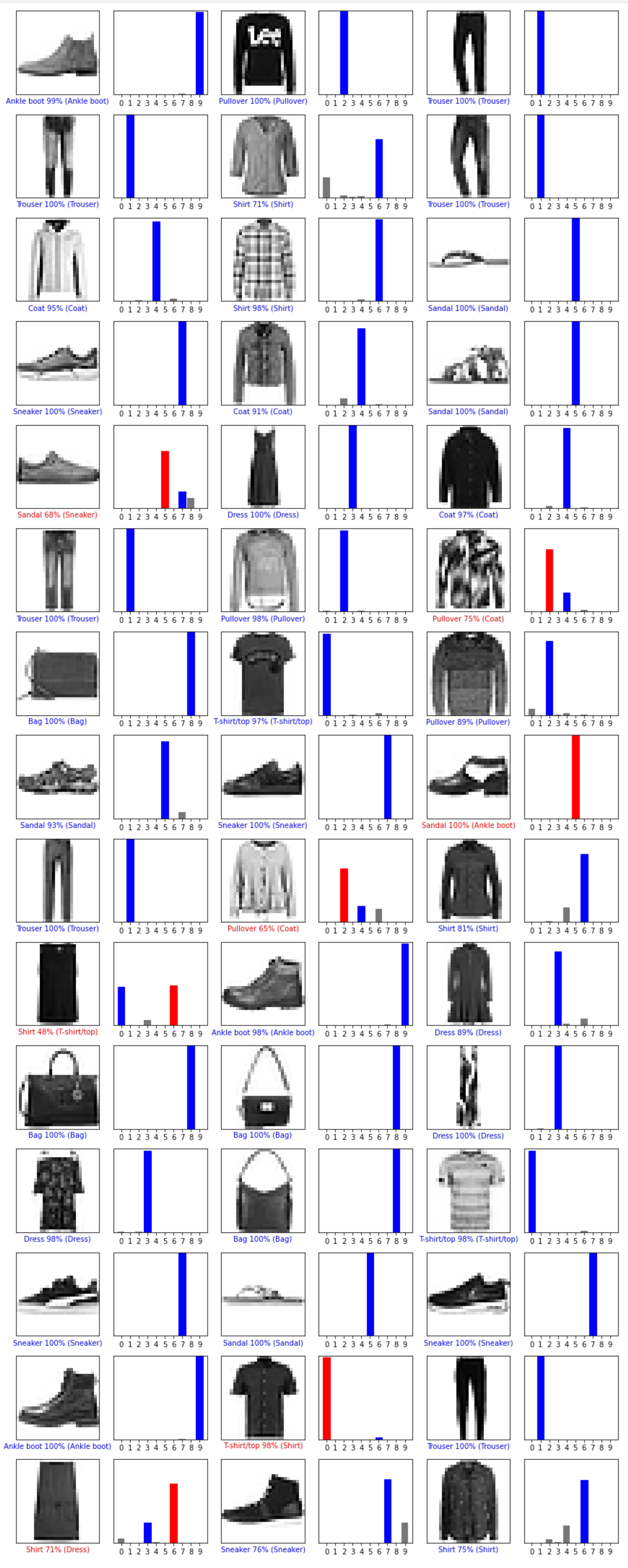
In [22]:
```python
i = 0
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions[i], test_labels, test_images)
plt.subplot(1,2,2)
plot_value_array(i, predictions[i],  test_labels)
plt.show()
```



In [33]:
```python
i = 5
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 [35]:
```python
# 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 = 15
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()
```

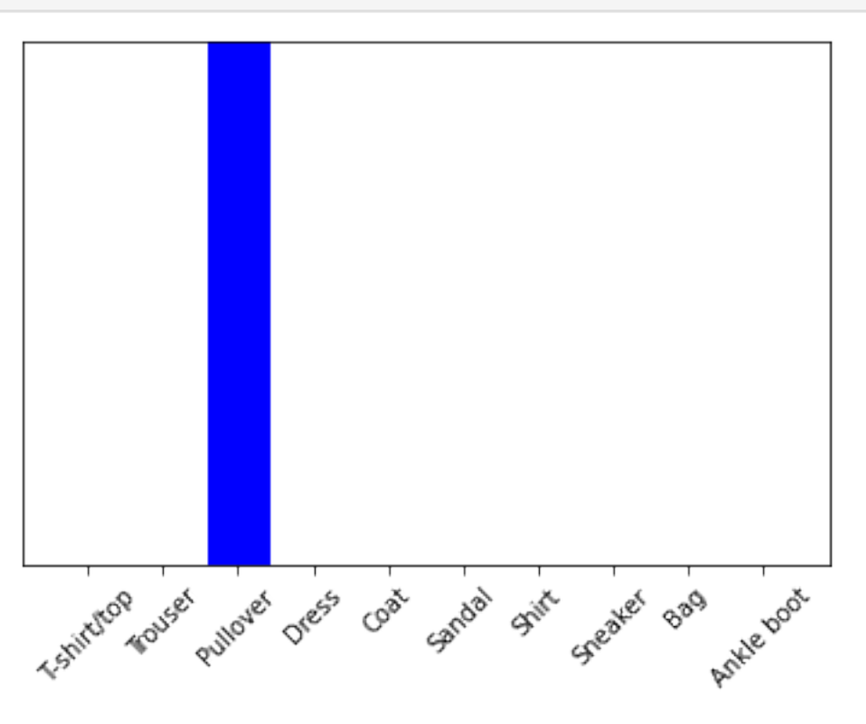


In [36]:
```python
# Grab an image from the test dataset.
img = test_images[1]

print(img.shape)
```
```
(28, 28)
```

In [37]:
```python
# 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 [38]:
```python
predictions_single = probability_model.predict(img)

print(predictions_single)
```
```
[[1.5250924e-04 9.2520427e-12 9.9820948e-01 1.3493254e-12 1.6040835e-03
  1.4625620e-13 3.3938726e-05 2.8209379e-20 1.6043294e-09 3.9804757e-16]]
```

In [39]:
```python
plot_value_array(1, predictions_single[0], test_labels)
_ = plt.xticks(range(10), class_names, rotation=45)
```



In [29]:
```python
np.argmax(predictions_single[0])
```
Out[29]:
```
2
```

In [ ]: