In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import pandas_datareader as web
import math
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense, LSTM
plt.style.use('fivethirtyeight')
```

In [2]:
```
df = web.DataReader('AAPL', data_source = 'yahoo', start='2012-01-01', end='2019-
df
```

Out[2]:

Date	High	Low	Open	Close	Volume	Adj Close
2012-01-03	14.732143	14.607143	14.621429	14.686786	302220800.0	12.575914
2012-01-04	14.810000	14.617143	14.642857	14.765714	260022000.0	12.643497
2012-01-05	14.948214	14.738214	14.819643	14.929643	271269600.0	12.783868
2012-01-06	15.098214	14.972143	14.991786	15.085714	318292800.0	12.917509
2012-01-09	15.276786	15.048214	15.196429	15.061786	394024400.0	12.897017
...	...	...	...	...	...	...
2019-12-11	67.775002	67.125000	67.202499	67.692497	78756800.0	66.616837
2019-12-12	68.139999	66.830002	66.945000	67.864998	137310400.0	66.786591
2019-12-13	68.824997	67.732498	67.864998	68.787498	133587600.0	67.694443
2019-12-16	70.197502	69.245003	69.250000	69.964996	128186000.0	68.853218
2019-12-17	70.442497	69.699997	69.892502	70.102501	114158400.0	68.988541

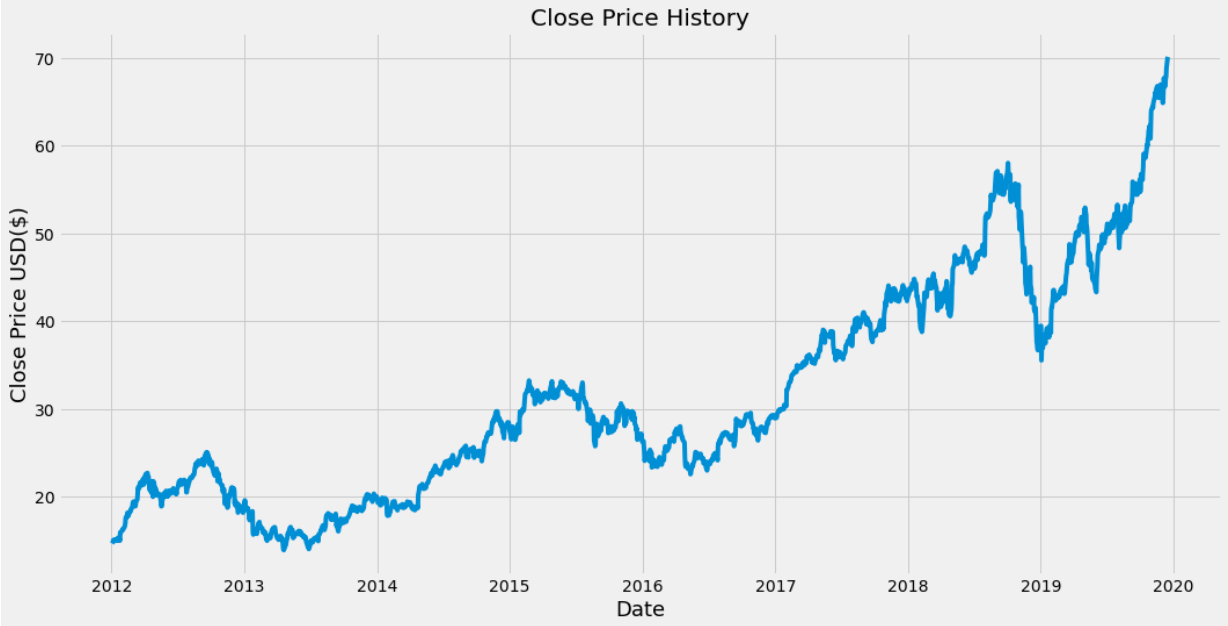
2003 rows × 6 columns

In [3]:
```
df.shape
```

Out[3]:
```
(2003, 6)
```

In [4]:
```
plt.figure(figsize=(16,8))
plt.title('Close Price History')
plt.plot(df['Close'])
plt.xlabel('Date', fontsize=18)
plt.ylabel('Close Price USD($)', fontsize=18)
```

Out[4]: Text(0, 0.5, 'Close Price USD($)')



In [5]:
```
data = df.filter(['Close'])
dataset = data.values
training_data_len = math.ceil(len(dataset)* 0.8)
training_data_len
```

Out[5]: 1603

In [6]:
```
scaler = MinMaxScaler(feature_range=(0,1))
scaled_data = scaler.fit_transform(dataset)
scaled_data
```

Out[6]:
```
array([[0.01316509],
       [0.01457063],
       [0.01748985],
       ...,
       [0.97658263],
       [0.99755134],
       [1.        ]])
```

In [8]:
```
train_data = scaled_data[0:training_data_len,:]
# Split the data into x_train and y_train data sets
x_train = []
y_train = []

for i in range(60,len(train_data)):
    x_train.append(train_data[i-60:i,0])
    y_train.append(train_data[i,0])
    if i<= 61:
        print(x_train)
        print(y_train)
        print()
```

```
[array([0.01316509, 0.01457063, 0.01748985, 0.02026915, 0.01984303,
       0.02080338, 0.02036454, 0.01962679, 0.01862191, 0.02173194,
       0.02453668, 0.02367172, 0.01893355, 0.02345548, 0.01900352,
       0.03569838, 0.03440732, 0.0360927 , 0.03973694, 0.04194384,
       0.0417594 , 0.0410789 , 0.04397903, 0.04670744, 0.04979839,
       0.05479095, 0.0652785 , 0.06543749, 0.07127594, 0.07563885,
       0.06814049, 0.07102789, 0.07097066, 0.07906688, 0.07791571,
       0.08004628, 0.08387497, 0.08600558, 0.09214292, 0.09661394,
       0.09790501, 0.09835659, 0.09071194, 0.08886753, 0.08914103,
       0.09632778, 0.09835024, 0.10269409, 0.11293358, 0.12659476,
       0.12403805, 0.1240444 , 0.13392141, 0.13701237, 0.13481179,
       0.13280207, 0.13070964, 0.13766105, 0.14243103, 0.14442805])]
[0.13949272033425864]

[array([0.01316509, 0.01457063, 0.01748985, 0.02026915, 0.01984303,
       0.02080338, 0.02036454, 0.01962679, 0.01862191, 0.02173194,
       0.02453668, 0.02367172, 0.01893355, 0.02345548, 0.01900352,
       0.03569838, 0.03440732, 0.0360927 , 0.03973694, 0.04194384,
       0.0417594 , 0.0410789 , 0.04397903, 0.04670744, 0.04979839,
       0.05479095, 0.0652785 , 0.06543749, 0.07127594, 0.07563885,
       0.06814049, 0.07102789, 0.07097066, 0.07906688, 0.07791571,
       0.08004628, 0.08387497, 0.08600558, 0.09214292, 0.09661394,
       0.09790501, 0.09835659, 0.09071194, 0.08886753, 0.08914103,
       0.09632778, 0.09835024, 0.10269409, 0.11293358, 0.12659476,
       0.12403805, 0.1240444 , 0.13392141, 0.13701237, 0.13481179,
       0.13280207, 0.13070964, 0.13766105, 0.14243103, 0.14442805]), array([0.0
1457063, 0.01748985, 0.02026915, 0.01984303, 0.02080338,
       0.02036454, 0.01962679, 0.01862191, 0.02173194, 0.02453668,
       0.02367172, 0.01893355, 0.02345548, 0.01900352, 0.03569838,
       0.03440732, 0.0360927 , 0.03973694, 0.04194384, 0.0417594 ,
       0.0410789 , 0.04397903, 0.04670744, 0.04979839, 0.05479095,
       0.0652785 , 0.06543749, 0.07127594, 0.07563885, 0.06814049,
       0.07102789, 0.07097066, 0.07906688, 0.07791571, 0.08004628,
       0.08387497, 0.08600558, 0.09214292, 0.09661394, 0.09790501,
       0.09835659, 0.09071194, 0.08886753, 0.08914103, 0.09632778,
       0.09835024, 0.10269409, 0.11293358, 0.12659476, 0.12403805,
       0.1240444 , 0.13392141, 0.13701237, 0.13481179, 0.13280207,
       0.13070964, 0.13766105, 0.14243103, 0.14442805, 0.13949272])]
[0.13949272033425864, 0.13293562570222134]
```

In [9]:
```
x_train, y_train = np.array(x_train), np.array(y_train)
```

In [13]:
```
# Reshape the data
#
x_train = np.reshape(x_train,(x_train.shape[0],x_train.shape[1],1))
x_train.shape
```

Out[13]:
```
(1543, 60, 1)
```

In [15]:
```
# build the LSTM model
model = Sequential()
model.add(LSTM(50, return_sequences=True, input_shape=(x_train.shape[1],1)))
model.add(LSTM(50, return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))
```

In [16]:
```
# compile the model
model.compile(optimizer = 'adam', loss = 'mean_squared_error')
```

In [17]:
```
#traiin the model
model.fit(x_train,y_train, batch_size=1, epochs= 1)
```

```
1543/1543 [==============================] - 36s 22ms/step - loss: 6.4344e-04
```

Out[17]:
```
<keras.callbacks.History at 0x29758c60d60>
```

In [26]:
```
# create the testing data set
# create a new array containing scaled values from index 1543 to 2003
test_data = scaled_data[training_data_len-60: ,:]
# create the data sets x_test and y_test
x_test = []
y_test = dataset[training_data_len:,:]

for i in range(60,len(test_data)):
    x_test.append(test_data[i-60:i,0])
```

In [27]:
```
# convert the data to a numpy araray
x_test = np.array(x_test)
```

In [28]:
```
x_test.shape
```

Out[28]:
```
(400, 60)
```

In [29]:
```
x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1))
```

In [30]:
```
x_test.shape
```

Out[30]:
```
(400, 60, 1)
```

In [31]:
```
# get the models predicted price values
predictions = model.predict(x_test)
predictions = scaler.inverse_transform(predictions)
```

In [32]:
```
#evaluate our model
# get the root mean squared error (RMSE)
rmse = np.sqrt(np.mean(predictions - y_test )**2 )
rmse
```

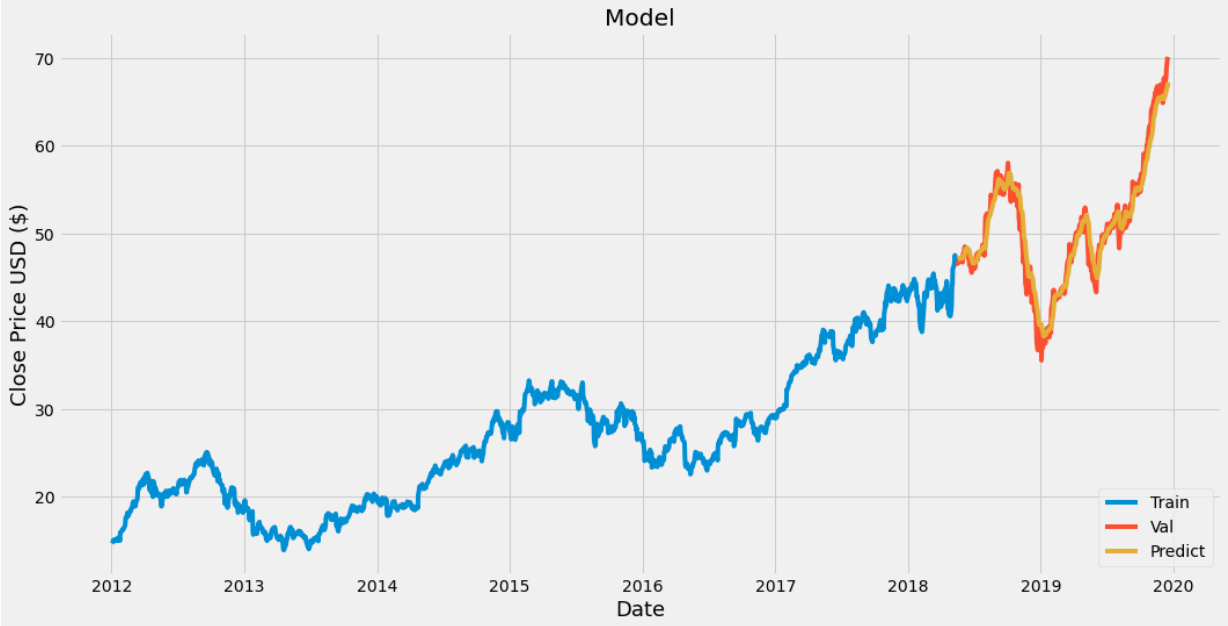
Out[32]:
```
0.005707483291625976
```

In [36]:
```
# plot the data
train = data[:training_data_len]
valid = data[training_data_len:]
valid['Predictions'] = predictions
# visualize the model
plt.figure(figsize=(16,8))
plt.title('Model')
plt.xlabel('Date', fontsize = 18)
plt.ylabel('Close Price USD ($)', fontsize= 18)
plt.plot(train['Close'])
plt.plot(valid[['Close', 'Predictions']])
plt.legend(['Train', 'Val', 'Predict'], loc = 'lower right')
```

```
C:\Users\ehsan\AppData\Local\Temp/ipykernel_16560/2019468251.py:4: SettingWithC
opyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
  valid['Predictions'] = predictions
```

Out[36]: <matplotlib.legend.Legend at 0x2975ea16670>



In [37]:
```
# Show the valid predicted prices
valid
```

Out[37]:

Date	Close	Predictions
2018-05-17	46.747501	47.137703
2018-05-18	46.577499	47.154289
2018-05-21	46.907501	47.115097
2018-05-22	46.790001	47.103394
2018-05-23	47.090000	47.089199
...	...	...
2019-12-11	67.692497	65.918266
2019-12-12	67.864998	66.155891
2019-12-13	68.787498	66.401611
2019-12-16	69.964996	66.743469
2019-12-17	70.102501	67.234062

400 rows × 2 columns

In [38]:
```
# get the quote
apple_quote = web.DataReader('AAPL', data_source='yahoo',start='2012-01-01',end='
# create a new dataframe
new_df = apple_quote.filter(['Close'])
# Get the las 60 day closing price values and convert the dataframe to an array
last_60_days = new_df[-60:].values
# Scale the data to be values between 0 and 1
last_60_days_scaled = scaler.transform(last_60_days)
# create an empty list
X_test = []
# append the past 60 days
X_test.append(last_60_days_scaled)
# Convert the X_test data set to numpy array
X_test = np.array(X_test)
# Reshape the data to be 3D
X_test = np.reshape(X_test,(X_test.shape[0],X_test.shape[1],1))
# get the predicted scaled price
pred_price = model.predict(X_test)
# undo the scaling
pred_price = scaler.inverse_transform(pred_price)
print(pred_price)
```

```
[[67.725784]]
```

In [40]:
```
apple_quote2 = web.DataReader('AAPL', data_source='yahoo',start='2019-12-18',end=
print(apple_quote2['Close'])
```

```
Date
2019-12-18    69.934998
Name: Close, dtype: float64
```

In [ ]:
```

```