The Effect of US Presidents on EUR/USD Movement ¶

The Euro/Dollar Pair

Whatever asset you decide to trade, one of the first lesson your mentor will teach you is to *look at the Euro and Dollar long term trend*. Traders persistently try to predict the EURO/USD long term movements, sometimes ending up in circles of uncertainty.

It wasn't until 1999 that the euro really began its journey, when 11 countries (Austria, Belgium, Finland, France, Germany, Ireland, Italy, Luxembourg, the Netherlands, Portugal and Spain) fixed their exchange rates and created a new currency with monetary policy passed to the European Central Bank. The currency has withstood the test of time, and today the Euro is over 20 years old. US Financial data providers now ask at the start of every year – what will be the direction of the EURO/USD pair?

This project will attempt to use a storytelling perspective to unfold the relationship between US Election results and the EURO/USD pair.

Dataset Information

The dataset we'll use describes Euro daily exchange rates between 1999 and 2021. It contains date and Euro rates corresponding to 40 currencies all over the world. This data has been put together and uploaded on [Kaggle (https://www.kaggle.com/lsind18/euro-exchange-daily-rates-19992020)](https://www.kaggle.com/lsind18/euro-exchange-daily-rates-19992020) by a user called *Daria Chemkaeva*. The data is constantly updated from its original source — the European Central Bank.

Summary of Findings

Whether the EURO/USD rate will appreciate or depreciate in the months after an election depends on the candidate who wins. If a Republican wins, the EURO/USD rates will rise, signifying a weaker dollar. If the Democratic party wins, EUR/USD rates fall due to the strengthening of the dollar. This movement is explained in part by differentiated fiscal and monetary policies adopted by the political parties.

Before we get started, what exactly is an exchange rate?

An exchange rate is the rate at which a country's currency will be exchanged for another country's currency. For example, if the exchange rate of the euro to the US dollar is 1.5, **you will get 1.5 US dollars if you pay 1.0 euro**. This means that the euro has more value than the US dollar at this exchange rate.

In [51]:
```
import pandas as pd

exchange_rates = pd.read_csv('euro-daily-hist_1999_2020.csv')
exchange_rates.head()
```

Out[51]:

	Period\Unit:	[Australian dollar]	[Bulgarian lev]	[Brazilian real]	[Canadian dollar]	[Swiss franc]	[Chinese yuan renminbi]	[Cypriot pound]	[Czech koruna]
0	2021-01-08	1.5758	1.9558	6.5748	1.5543	1.0827	7.9184	NaN	26.163
1	2021-01-07	1.5836	1.9558	6.5172	1.5601	1.0833	7.9392	NaN	26.147
2	2021-01-06	1.5824	1.9558	6.5119	1.5640	1.0821	7.9653	NaN	26.145
3	2021-01-05	1.5927	1.9558	6.5517	1.5651	1.0803	7.9315	NaN	26.227
4	2021-01-04	1.5928	1.9558	6.3241	1.5621	1.0811	7.9484	NaN	26.141

5 rows × 41 columns

In [52]:
```
exchange_rates.tail(3)
```

Out[52]:

	Period\Unit:	[Australian dollar]	[Bulgarian lev]	[Brazilian real]	[Canadian dollar]	[Swiss franc]	[Chinese yuan renminbi]	[Cypriot pound]	[Cz kor
5696	1999-01-06	1.8820	NaN	NaN	1.7711	1.6116	NaN	0.58200	34.
5697	1999-01-05	1.8944	NaN	NaN	1.7965	1.6123	NaN	0.58230	34.
5698	1999-01-04	1.9100	NaN	NaN	1.8004	1.6168	NaN	0.58231	35.

3 rows × 41 columns

In [53]:
```
exchange_rates.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5699 entries, 0 to 5698
Data columns (total 41 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Period\Unit:               5699 non-null   object
 1   [Australian dollar ]       5699 non-null   object
 2   [Bulgarian lev ]           5297 non-null   object
 3   [Brazilian real ]          5431 non-null   object
 4   [Canadian dollar ]         5699 non-null   object
 5   [Swiss franc ]             5699 non-null   object
 6   [Chinese yuan renminbi ]   5431 non-null   object
 7   [Cypriot pound ]           2346 non-null   object
 8   [Czech koruna ]            5699 non-null   object
 9   [Danish krone ]            5699 non-null   object
 10  [Estonian kroon ]          3130 non-null   object
 11  [UK pound sterling ]       5699 non-null   object
 12  [Greek drachma ]           520 non-null    object
 13  [Hong Kong dollar ]        5699 non-null   object
```

Initial Notes

- *There are 5699 rows and 41 columns in the dataset. The first column corresponds to time, while the others correspond to 40 different currencies.*
- *17 columns have null values. In fact, over 50% of values in some columns are null. Notable examples are the Greek drachma and Slovanian tolar columns.*
- *Since we are dealing with exchange rates, many of these columns should be numeric. However, only three of the currency columns are stored with the correct data type. The period/Unit column should also be stored as a DateTime object rather than a string/object type.*
- *Column names do not follow the Python snake case convention. They also contain some extra characters that might make them challenging to work with.*

In the next few steps we will be correcting some of these irregularities. We are only concerned about the EURO/USD pair, hence, we will direct our cleaning towards the required columns - `Period/Unit` and `[US dollar ]`.

Data Cleaning

Our cleaning will follow three simple steps. First we'll rename the two columns: `Period/Unit` and `[US dollar ]`. Next, we'd extract the columns into a new dataframe called `usd_data`. Finally, we will convert the two columns into the right data types:

In [54]:
```
# Rename relevant columns
exchange_rates.rename(columns={'[US dollar ]': 'US_dollar',
                               'Period\\Unit:': 'Time'},
                      inplace=True)
# Datatype conversion
exchange_rates['Time'] = pd.to_datetime(exchange_rates['Time'])
exchange_rates.sort_values('Time', inplace=True)
exchange_rates.reset_index(drop=True, inplace=True)
exchange_rates.head()
```

Out[54]:

	Time	[Australian dollar]	[Bulgarian lev]	[Brazilian real]	[Canadian dollar]	[Swiss franc]	[Chinese yuan renminbi]	[Cypriot pound]	[Czech koruna]	[Dan kror
0	1999-01-04	1.9100	NaN	NaN	1.8004	1.6168	NaN	0.58231	35.107	7.4
1	1999-01-05	1.8944	NaN	NaN	1.7965	1.6123	NaN	0.58230	34.917	7.4
2	1999-01-06	1.8820	NaN	NaN	1.7711	1.6116	NaN	0.58200	34.850	7.4
3	1999-01-07	1.8474	NaN	NaN	1.7602	1.6165	NaN	0.58187	34.886	7.4
4	1999-01-08	1.8406	NaN	NaN	1.7643	1.6138	NaN	0.58187	34.938	7.4

5 rows × 41 columns

In [55]:
```
# Examine the US_dollar column
euro_to_dollar = exchange_rates[['Time', 'US_dollar']].copy()
euro_to_dollar['US_dollar'].value_counts() # 62 '-' characters
```

Out[55]:
```
-         62
1.2276     9
1.1215     8
1.1305     7
1.1346     6
          ..
1.4038     1
0.9774     1
0.8609     1
1.3503     1
1.0718     1
Name: US_dollar, Length: 3528, dtype: int64
```

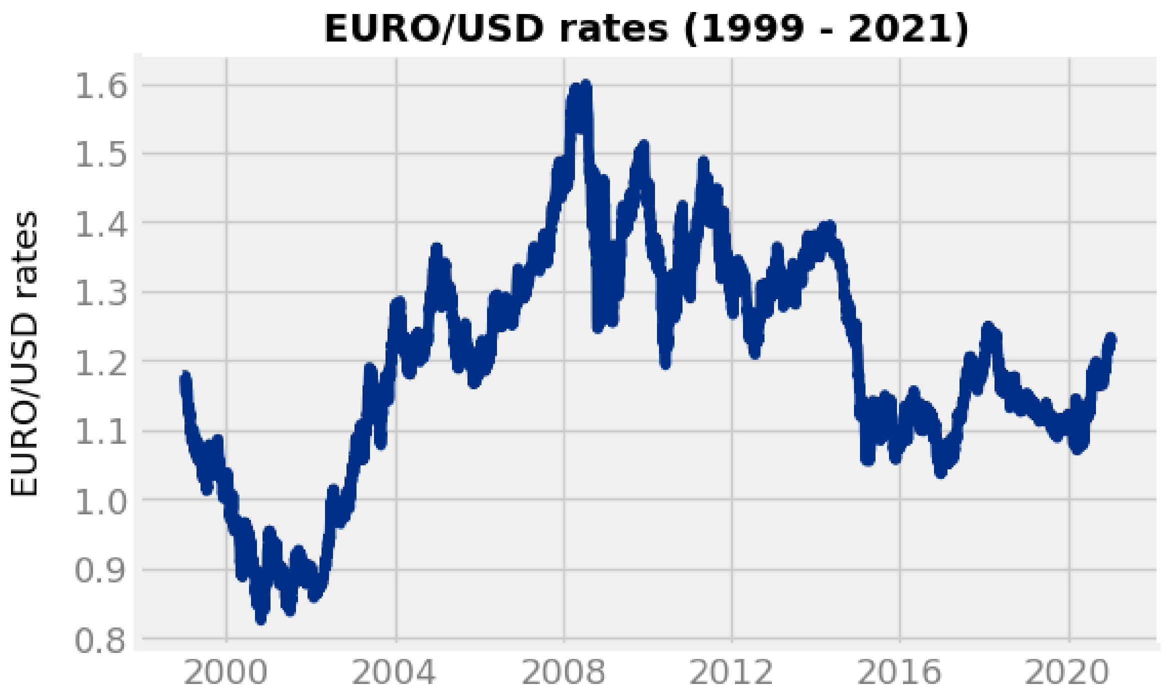
It appears that we have some missing values recorded as '-' in the us_dollar column. These values can cause our conversion process to throw up errors. They can also affect the results of our analysis. We will remove them from our data:

In [56]:
```
euro_to_dollar = euro_to_dollar[euro_to_dollar['US_dollar'] != '-']
euro_to_dollar['US_dollar'] = euro_to_dollar['US_dollar'].astype(float)
euro_to_dollar.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5637 entries, 0 to 5698
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Time       5637 non-null   datetime64[ns]
 1   US_dollar  5637 non-null   float64
dtypes: datetime64[ns](1), float64(1)
memory usage: 132.1 KB
```

Rolling Mean

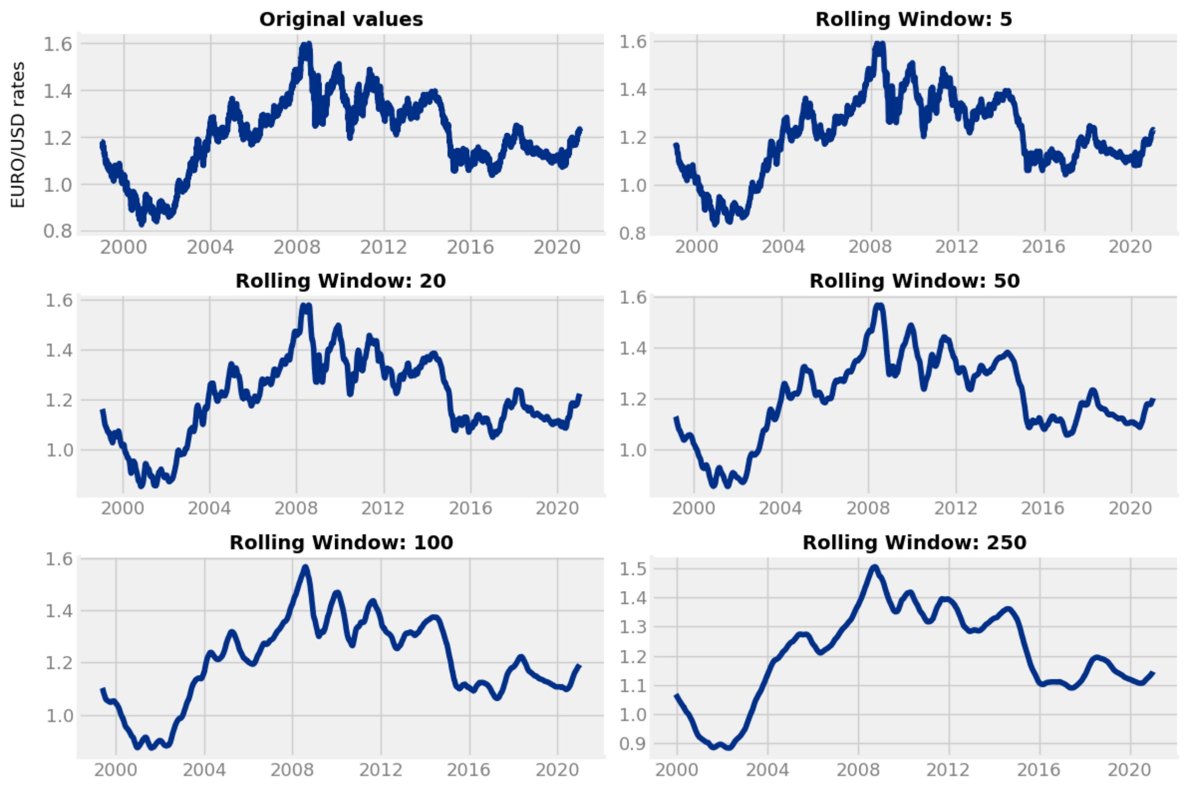
In [101]:
```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
# Enables Jupyter to display graphs

plt.plot(euro_to_dollar['Time'], euro_to_dollar['US_dollar'], color='#002F87')
plt.xticks(size=13, color='grey')
plt.yticks(size=13, color='grey')
plt.ylabel('EURO/USD rates', size=13, labelpad=10)
plt.title('EURO/USD rates (1999 - 2021)', size='14', weight='bold')
sns.despine()

plt.show()
```



Its quite challenging to make sense of this graph. We see many small wiggles — rather than seeing a smooth line. These are the visual representation of the daily variations in exchange rate. The rate goes up and down from day to day, only showing clear upward or downward trends in the longer run (months or years).

Depending on our goals, we may not want to show the daily variations on our graph. If we want to show only the long-term trends, we can use the moving average(rolling mean). To calculate the rolling mean over any period of time, we can use the handy pandas.Series.rolling().mean() method:

In [113]:
```
plt.figure(figsize=(12,8))

plt.subplot(3,2,1)
plt.plot(euro_to_dollar['Time'], euro_to_dollar['US_dollar'], color='#002F87')
plt.xticks(size=14, color='grey')
plt.yticks(size=14, color='grey')
plt.ylabel('EURO/USD rates', size=13, labelpad=10)
plt.title('Original values', weight='bold', size=14)

for i, rolling_mean in zip([2, 3, 4, 5, 6],
                           [5, 20, 50, 100, 250]):
    plt.subplot(3,2,i)
    plt.plot(euro_to_dollar['Time'],
             euro_to_dollar['US_dollar'].rolling(rolling_mean).mean(),  color=
    plt.xticks(size=13, color='grey')
    plt.yticks(size=13, color='grey')
    plt.title('Rolling Window: ' + str(rolling_mean), weight='bold',size=14)
    sns.despine()

plt.tight_layout() # Auto-adjusts the padding between subplots
#Tự động điều chỉnh phần đệm giữa các ô con
plt.show()
```



The 50-day rolling window appears to be a sweet spot. We can see detailed fluctuations in exchange rates, without losing track of long term trends. For most of our analysis, we will stick to the 50-day rolling window. Let's add this column to our dataframe:

In [114]:
```
euro_to_dollar['rolling_mean'] = euro_to_dollar['US_dollar'].rolling(50).mean(
euro_to_dollar
```

Out[114]:

	Time	US_dollar	rolling_mean
0	1999-01-04	1.1789	NaN
1	1999-01-05	1.1790	NaN
2	1999-01-06	1.1743	NaN
3	1999-01-07	1.1632	NaN
4	1999-01-08	1.1659	NaN
...	...	...	...
5694	2021-01-04	1.2296	1.198524
5695	2021-01-05	1.2271	1.199354
5696	2021-01-06	1.2338	1.200392
5697	2021-01-07	1.2276	1.201280
5698	2021-01-08	1.2250	1.202326

5637 rows × 3 columns

Storytelling

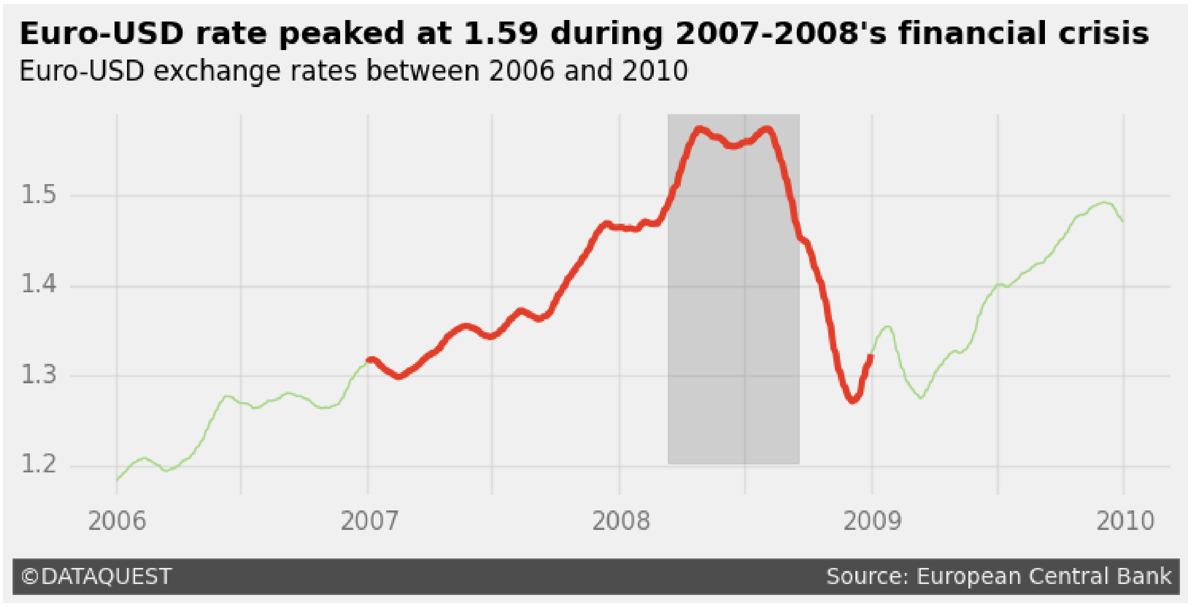
The Influence of The US Elections on EURO/USD Rates

Before investigating for the impact of the Influence of The US Elections on EURO/USD Rates, we start by exploring the change of Exchange rate in the Financial Crisis (2007-2008).

Financial Crisis

In [65]:
```
financial_crisis = euro_to_dollar.copy(
                   )[(euro_to_dollar['Time'].dt.year >= 2006
                   ) & (euro_to_dollar['Time'].dt.year <= 2009)]
financial_crisis_7_8 = euro_to_dollar.copy(
                   )[(euro_to_dollar.Time.dt.year >= 2007
                   ) & (euro_to_dollar.Time.dt.year <= 2008)]
```

In [66]:
```
### Adding the FiveThirtyEight style
import matplotlib.style as style
style.use('fivethirtyeight')

### Adding the plot
fig,ax = plt.subplots(figsize=(8,3))
ax.plot(financial_crisis['Time'],
        financial_crisis['rolling_mean'],
        linewidth=1, color='#A6D785')

### Highlighting the 2007-2008 period
ax.plot(financial_crisis_7_8['Time'],
        financial_crisis_7_8['rolling_mean'],
        linewidth=3, color='#e23d28')

### Highlihting the peak of the crisis
ax.axvspan(xmin=733112.0, xmax=733302.0, ymin=0.09,
           alpha=0.3, color='grey')

### Adding separate tick labels
ax.set_xticklabels([])
ax.set_yticklabels([])

x = 732272.0
for year in ['2006', '2007', '2008', '2009', '2010']:
    ax.text(x, 1.13, year, alpha=0.5, fontsize=11)
    x += 365

y = 1.193
for rate in ['1.2', '1.3', '1.4', '1.5']:
    ax.text(732172.0, y, rate, alpha=0.5, fontsize=11)
    y += 0.1

## Adding a title and a subtitle
ax.text(732172.0, 1.67, "Euro-USD rate peaked at 1.59 during 2007-2008's finan
        weight='bold')
ax.text(732172.0, 1.63, 'Euro-USD exchange rates between 2006 and 2010',
        size=12)

### Adding a signature
ax.text(732172.0, 1.07, '©DATAQUEST' + ' '*94 + 'Source: European Central Bank
        color = '#f0f0f0', backgroundcolor = '#4d4d4d',
        size=10)

### Add some transparency to the grid - độ mờ của lưới
ax.grid(alpha=0.5)  #color = 'green', linestyle = '--',

plt.show()
```



At this time, the EURO/USD exchange rate increased sharply, showing the period of cheap money in the context of the financial bubble that took place in the United States first. When the crisis spread to the whole world, the EU countries printed money, the exchange rate fell again.

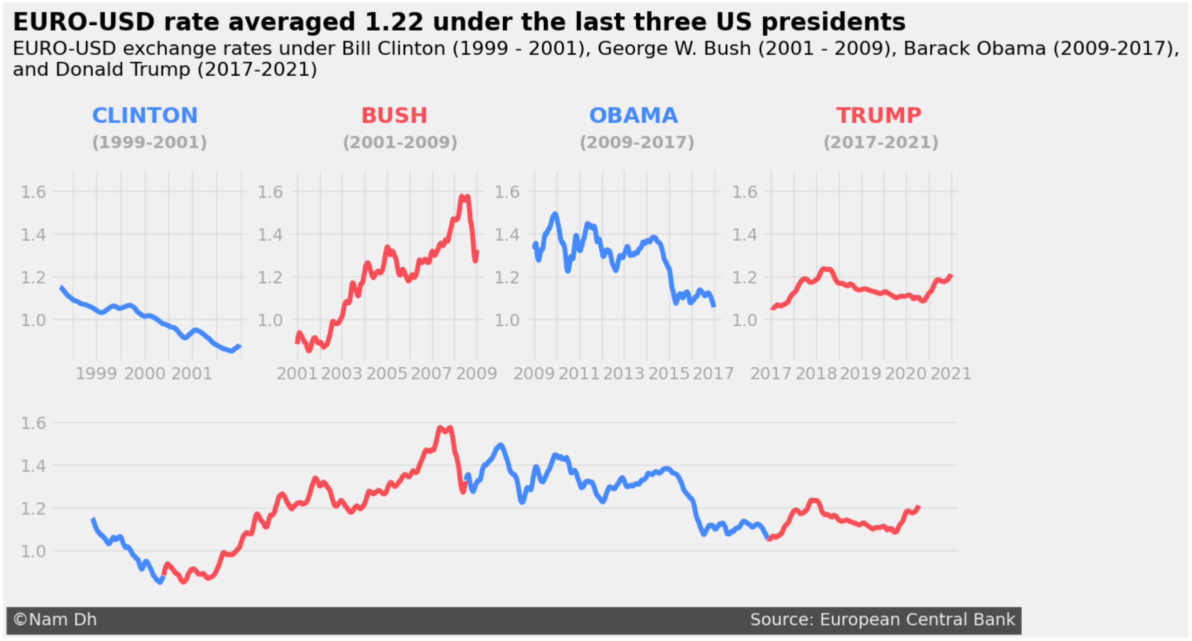
Without getting so deep in politics, the US ruling party may influence the EURO/USD pair. There might be a correlation between a political party's agenda and the USD price. Every ruling party has a different economic plan. In general, Republicans tend to focus inwards, control tax levels, revise agreements with other countries, and increase military spending. Democrats focus on external expansion and decreasing military spending. These policies could stimulate or negatively impact the US economy.

The Three US Presidencies Example

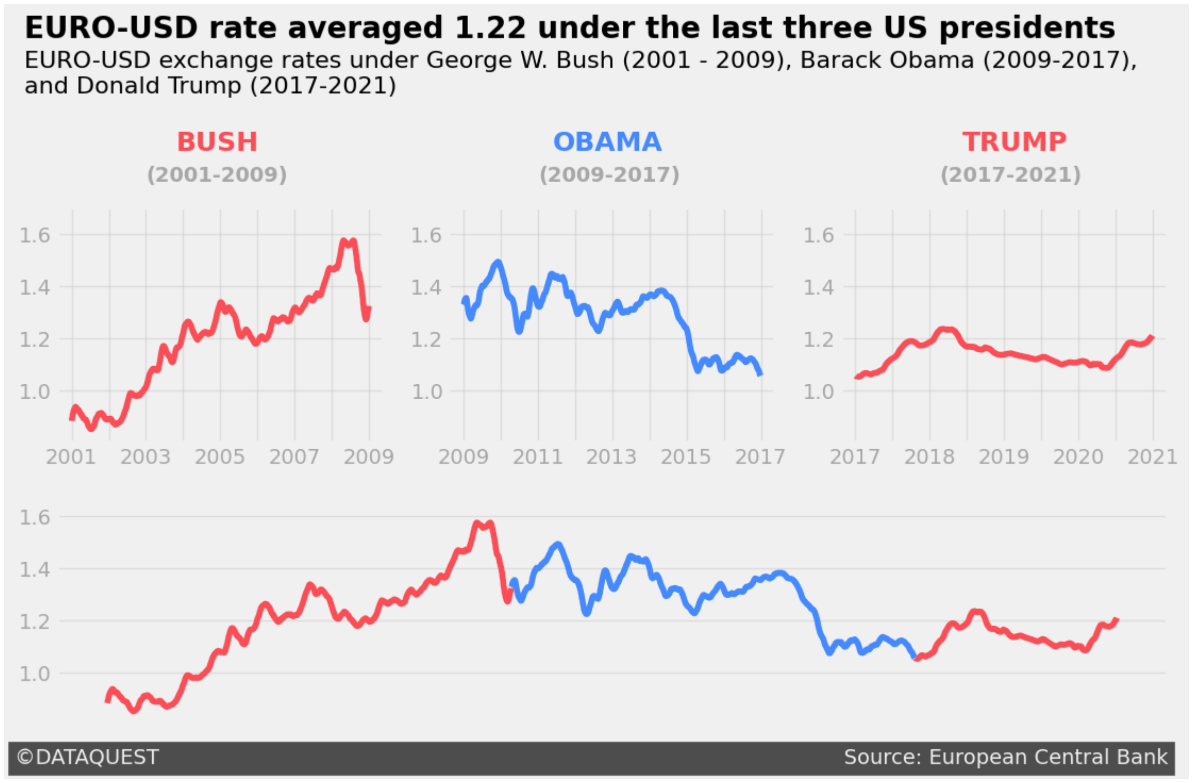
In [79]:
```
bush_obama_trump = euro_to_dollar.copy(
                   )[(euro_to_dollar['Time'].dt.year >= 1999) & (euro_to_dolla

clinton = bush_obama_trump.copy(
       )[(bush_obama_trump['Time'].dt.year >= 1999) & (bush_obama_trump['Time'
obama = bush_obama_trump.copy(
       )[(bush_obama_trump['Time'].dt.year >= 2001) & (bush_obama_trump['Time'
obama = bush_obama_trump.copy(
       )[(bush_obama_trump['Time'].dt.year >= 2009) & (bush_obama_trump['Time'
trump = bush_obama_trump.copy(
       )[(bush_obama_trump['Time'].dt.year >= 2017) & (bush_obama_trump['Time'
```

In [147]:
```
### Adding the FiveThirtyEight style
style.use('fivethirtyeight')

### Adding the subplots
plt.figure(figsize=(12, 6))
ax0 = plt.subplot(2,4,1)
ax1 = plt.subplot(2,4,2)
ax2 = plt.subplot(2,4,3)
ax3 = plt.subplot(2,4,4)
ax4 = plt.subplot(2,1,2)
axes = [ax0, ax1, ax2, ax3, ax4]

### Changes to all the subplots
for ax in axes:
    ax.set_ylim(0.8, 1.7)
    ax.set_yticks([1.0, 1.2, 1.4, 1.6])
    ax.set_yticklabels(['1.0', '1.2','1.4', '1.6'],
                       alpha=0.3)
    ax.grid(alpha=0.5)

### Ax0: Clinton
ax0.plot(clinton['Time'], clinton['rolling_mean'],
        color='#4589ff')
ax0.set_xticklabels(['', '1999', '','2000','', '2001'],
                   alpha=0.3)
ax0.text(729916.0, 1.92, 'CLINTON', fontsize=18, weight='bold',
        color='#4589ff')
ax0.text(729916.0, 1.8, '(1999-2001)', weight='bold',
        alpha=0.3)

### Ax1: Bush
ax1.plot(bush['Time'], bush['rolling_mean'],
        color='#fa4d56')
ax1.set_xticklabels(['', '2001', '', '2003', '', '2005', '',
                     '2007', '', '2009'],
                   alpha=0.3)
ax1.text(731516.0, 1.92, 'BUSH', fontsize=18, weight='bold',
        color='#fa4d56')
ax1.text(731216.0, 1.8, '(2001-2009)', weight='bold',
        alpha=0.3)


### Ax2: Obama
ax2.plot(obama['Time'], obama['rolling_mean'],
        color='#4589ff')
ax2.set_xticklabels(['', '2009', '', '2011', '', '2013', '',
                     '2015', '', '2017'],
                   alpha=0.3)
ax2.text(734288.0, 1.92, 'OBAMA', fontsize=18, weight='bold',
        color='#4589ff')
ax2.text(734138.0, 1.8, '(2009-2017)', weight='bold',
         alpha=0.3)


### Ax3: Trump
ax3.plot(trump['Time'], trump['rolling_mean'],
        color='#fa4d56')
```

```
ax3.set_xticklabels(['2017', '', '2018', '', '2019', '',
                     '2020', '', '2021'],
                   alpha=0.3)
ax3.text(736855.0, 1.92, 'TRUMP', fontsize=18, weight='bold',
        color='#fa4d56')
ax3.text(736745.0, 1.8, '(2017-2021)', weight='bold',
         alpha=0.3)

### Ax4: Bush-Obama-Trump
ax4.plot(clinton['Time'], clinton['rolling_mean'],
        color='#4589ff')
ax4.plot(bush['Time'], bush['rolling_mean'],
        color='#fa4d56')
ax4.plot(obama['Time'], obama['rolling_mean'],
        color='#4589ff')
ax4.plot(trump['Time'], trump['rolling_mean'],
        color='#fa4d56')
ax4.grid(alpha=0.5)
ax4.set_xticks([])

### Adding a title and a subtitle
ax0.text(729616.0, 2.35, 'EURO-USD rate averaged 1.22 under the last three US
         fontsize=20, weight='bold')
ax0.text(729616.0, 2.14, '''EURO-USD exchange rates under Bill Clinton (1999 -
and Donald Trump (2017-2021)''',
        fontsize=16)

### Adding a signature
ax4.text(729016.0, 0.65, '©Nam Dh' + ' '*120 + 'Source: European Central Bank'
        color = '#f0f0f0', backgroundcolor = '#4d4d4d',
        size=14)

plt.show()
```



In [146]:
```
### Adding the FiveThirtyEight style
style.use('fivethirtyeight')

### Adding the subplots
plt.figure(figsize=(12, 6))
ax1 = plt.subplot(2,3,1)
ax2 = plt.subplot(2,3,2)
ax3 = plt.subplot(2,3,3)
ax4 = plt.subplot(2,1,2)
axes = [ax1, ax2, ax3, ax4]

### Changes to all the subplots
for ax in axes:
    ax.set_ylim(0.8, 1.7)
    ax.set_yticks([1.0, 1.2, 1.4, 1.6])
    ax.set_yticklabels(['1.0', '1.2','1.4', '1.6'],
                   alpha=0.3)
    ax.grid(alpha=0.5)


### Ax1: Bush
ax1.plot(bush['Time'], bush['rolling_mean'],
        color='#fa4d56')
ax1.set_xticklabels(['', '2001', '', '2003', '', '2005', '',
                    '2007', '', '2009'],
                   alpha=0.3)
ax1.text(731516.0, 1.92, 'BUSH', fontsize=18, weight='bold',
        color='#fa4d56')
ax1.text(731216.0, 1.8, '(2001-2009)', weight='bold',
        alpha=0.3)


### Ax2: Obama
ax2.plot(obama['Time'], obama['rolling_mean'],
        color='#4589ff')
ax2.set_xticklabels(['', '2009', '', '2011', '', '2013', '',
                    '2015', '', '2017'],
                   alpha=0.3)
ax2.text(734288.0, 1.92, 'OBAMA', fontsize=18, weight='bold',
        color='#4589ff')
ax2.text(734138.0, 1.8, '(2009-2017)', weight='bold',
         alpha=0.3)


### Ax3: Trump
ax3.plot(trump['Time'], trump['rolling_mean'],
        color='#fa4d56')
ax3.set_xticklabels(['2017', '', '2018', '', '2019', '',
                    '2020', '', '2021'],
                   alpha=0.3)
ax3.text(736855.0, 1.92, 'TRUMP', fontsize=18, weight='bold',
        color='#fa4d56')
ax3.text(736745.0, 1.8, '(2017-2021)', weight='bold',
         alpha=0.3)

### Ax4: Bush-Obama-Trump
ax4.plot(bush['Time'], bush['rolling_mean'],
        color='#fa4d56')
ax4.plot(obama['Time'], obama['rolling_mean'],
        color='#4589ff')
ax4.plot(trump['Time'], trump['rolling_mean'],
        color='#fa4d56')
ax4.grid(alpha=0.5)
ax4.set_xticks([])

### Adding a title and a subtitle
ax1.text(730016.0, 2.35, 'EURO-USD rate averaged 1.22 under the last three US
         fontsize=20, weight='bold')
ax1.text(730016.0, 2.14, '''EURO-USD exchange rates under George W. Bush (2001
and Donald Trump (2017-2021)''',
        fontsize=16)

### Adding a signature
ax4.text(729816.0, 0.65, '©DATAQUEST' + ' '*103 + 'Source: European Central Ba
        color = '#f0f0f0', backgroundcolor = '#4d4d4d',
        size=14)

plt.show()
```



Conclusion

When we talk about US dollar performance, we refer to a currency pairing. We examined one of the most important currency pairs in trading - The EURO/USD pair, and compared the long term trends in relation to the US election results. Overall, we learned that the Republican policy builds on the weaker dollar, while the Democrats favor the stronger dollar.

Our explanation of this phenomenon is that Democratic presidents often implement policies that stimulate short-term economic growth and higher consumption, which causes the US dollar to appreciate. A Republican administration, however, is usually identified with a pro-business agenda, which may result in a weaker dollar. President Donald Trump has often said he wants to see a lower US dollar to make US exports more competitive. He has not hesitated to criticise the Federal Reserve for not reducing interest rates, which would cause the greenback to depreciate.

Prompt for Future Exploration

In this analysis, we have only considered the impact of the US elections on the EURO/USD rates. It would be of immense value to also evaluate the effects of other significant events like