0.1 Question 1

Discuss one attribute or characteristic you notice that is different between the two emails that may allow you to uniquely identify a spam email.

The Spam email is in HTML form, whereas the Ham email looks more like it is written in a more regular format. Spam emails may be more likely to come in HTML code for mass emailing, whereas regular emails will be written in a more humanoid format because they are only meant to be sent out once.

Create your bar chart in the following cell:

In [14]:
```
# Want to generate the proportion of ham/spam emails that contain the word.
# First, sum up the number of emails where the word is found, then divide by the total number
# np.sum along the index axis (sum the columns, not rows) because each column from the output
# are the occurrences for that word.
spam_words = ['href', 'free', 'act', 'congratulations', 'debt', 'income']

# all_ham_emails = original_training_data.loc[original_training_data['spam'] == 0, 'email']
train_ham_emails = train.loc[train['spam'] == 0, 'email']
train_spam_emails = train.loc[train['spam'] == 1, 'email']
```

In [15]:
```
print(len(all_ham_emails))
print(len(train_ham_emails))
```

```
6208
5595
```

In [16]:
```
words_in_ham = words_in_texts(spam_words, train_ham_emails)
summed_ham_words = np.sum(words_in_ham, axis=0)
summed_ham_words
```

Out[16]:
```
array([ 326, 1729, 2194,   15,   33,   63])
```

In [18]:
```
words_in_spam = words_in_texts(spam_words, train_spam_emails)
summed_spam_words = np.sum(words_in_spam, axis=0)
summed_spam_words
```

Out[18]:
```
array([ 976,  943, 1054,   41,  140,  170])
```

In [19]:
```
prop_ham_words = summed_ham_words / len(train_ham_emails)
prop_spam_words = summed_spam_words / len(train_spam_emails)
print(prop_ham_words, prop_spam_words)
```

```
[0.05826631 0.30902592 0.39213584 0.00268097 0.00589812 0.01126005] [0.5088634  0.49165798 0.54953076 0
```

In [20]:
```
ham_df = pd.DataFrame(prop_ham_words.reshape(1, -1), columns=spam_words)
ham_df['type'] = 'Ham'
ham_df
```

Out[20]:
```
       href      free       act  congratulations      debt   income type
0  0.058266  0.309026  0.392136         0.002681  0.005898  0.01126  Ham
```

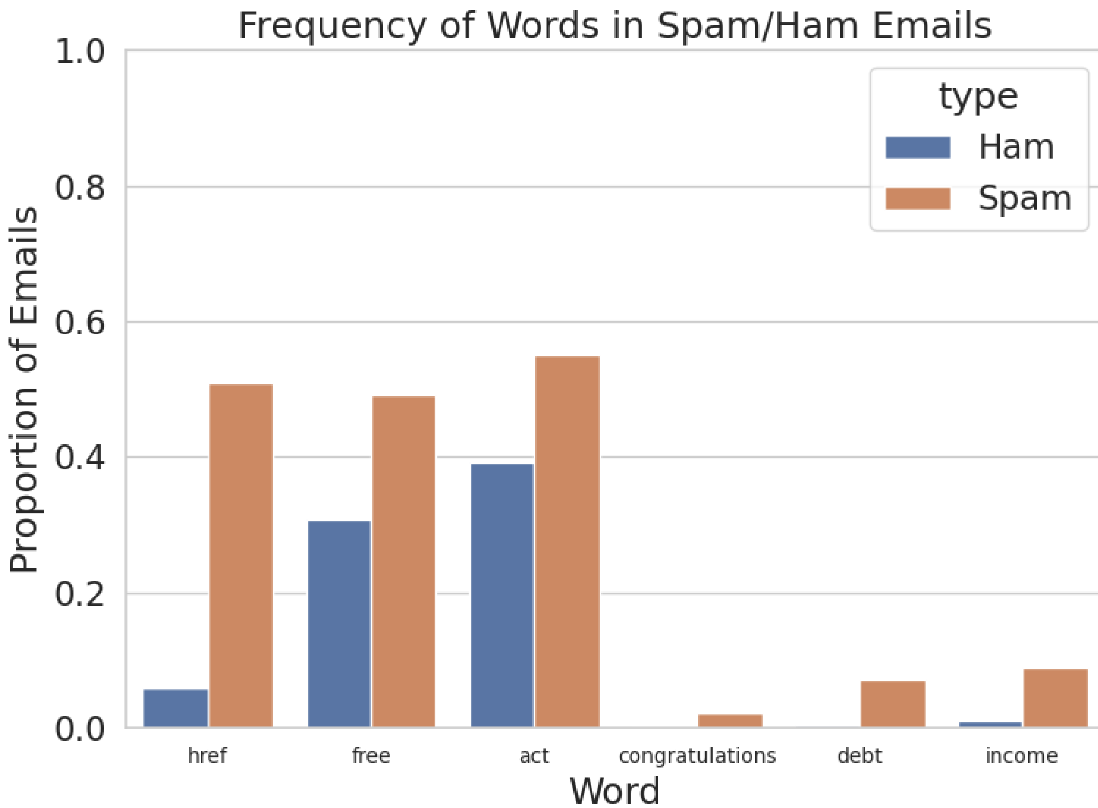
In [21]:
```
spam_df = pd.DataFrame(prop_spam_words.reshape(1, -1), columns=spam_words)
spam_df['type'] = 'Spam'
spam_df
```

Out[21]:
```
       href      free       act  congratulations      debt    income  type
0  0.508863  0.491658  0.549531         0.021376  0.072993  0.088634  Spam
```

In [22]:
```
spam_ham_df = pd.concat([ham_df, spam_df]).melt('type')
spam_ham_df
```

Out[22]:
```
    type         variable     value
0    Ham             href  0.058266
1   Spam             href  0.508863
2    Ham             free  0.309026
3   Spam             free  0.491658
4    Ham              act  0.392136
5   Spam              act  0.549531
6    Ham  congratulations  0.002681
7   Spam  congratulations  0.021376
8    Ham             debt  0.005898
9   Spam             debt  0.072993
10   Ham           income  0.011260
11  Spam           income  0.088634
```

In [23]:
```
train = train.reset_index(drop=True) # We must do this in order to preserve the ordering of em
plt.figure(figsize=(8,6))
sns.barplot(data=spam_ham_df, x='variable', y='value', hue='type')

ax = plt.gca()
ax.set_ylim(0, 1)
ax.set_ylabel('Proportion of Emails')

ax.tick_params(axis='x', which='major', labelsize=10)
ax.set_xlabel('Word')

plt.title('Frequency of Words in Spam/Ham Emails')

plt.tight_layout()
plt.show()
```


Frequency of Words in Spam/Ham Emails
type
Ham
Spam
Proportion of Emails
1.0
0.8
0.6
0.4
0.2
0.0
href
free
act
congratulations
debt
income
Word

0.2 Question 6c

Explain your results in `q6a` and `q6b`. How did you know what to assign to `zero_predictor_fp`, `zero_predictor_fn`, `zero_predictor_acc`, and `zero_predictor_recall`?

Because the zero predictor only ever classifies emails as zeroes/Ham/Negative, that means that none of the emails are classified as 1/Spam/**Positive**. Therefore, the number of True Positives *and* False Positives is 0. Therefore, `zero_predictor_fp = 0`. Because recall relies on the number of True Positives in the numerator, it is also true that `zero_predictor_recall = 0`.

For false negatives, since zero_predictor only classifies emails as negatives, the only *false* negatives are the emails that are actually spam. Therefore, the number of false negatives are the number of emails in the dataset that have the true classification of spam.

Finally, the accuracy remains still as the number of labels that the zero_predictor predicted correctly. Since the zero_predictor only predicts 0's, the correct predictions are for the emails whose true labels are zero. Therefore, it is the number of labels in the training set that are actually Ham divided by the total number of emails, ie. the proportion of emails that are actually Ham in the training set.

0.3 Question 6f

How does the accuracy of the logistic regression classifier `my_model` compare to the accuracy of the zero predictor?

```
In [51]: logistic_predictor_accuracy = np.sum((Y_train_hat == Y_train)) / len(Y_train)
         print(f'logistic_predictor_accuracy: {logistic_predictor_accuracy}, zero_predictor_accuracy: {z
         print(logistic_predictor_accuracy - zero_predictor_acc)
```

```
logistic_predictor_accuracy: 0.7576201251164648, zero_predictor_accuracy: 0.7447091707706642
0.012910954345800585
```

The accuracy of my_model is only slightly greater than the zero_predictor. my_model has an accuracy of 75.76%, while the zero_predictor has an accuracy of 74.47%. Thus, the logistic predictor only has a greater accuracy of 1.29%, which seems relatively minimal, considering our logistic predictor should in theory be better than just never classifying an email as spam.

0.4 Question 6g

Given the word features provided in Question 4, discuss why the logistic regression classifier `my_model` may be performing poorly.

Hint: Think about how prevalent these words are in the email set.

```
In [52]: some_words
```

```
Out[52]: ['drug', 'bank', 'prescription', 'memo', 'private']
```

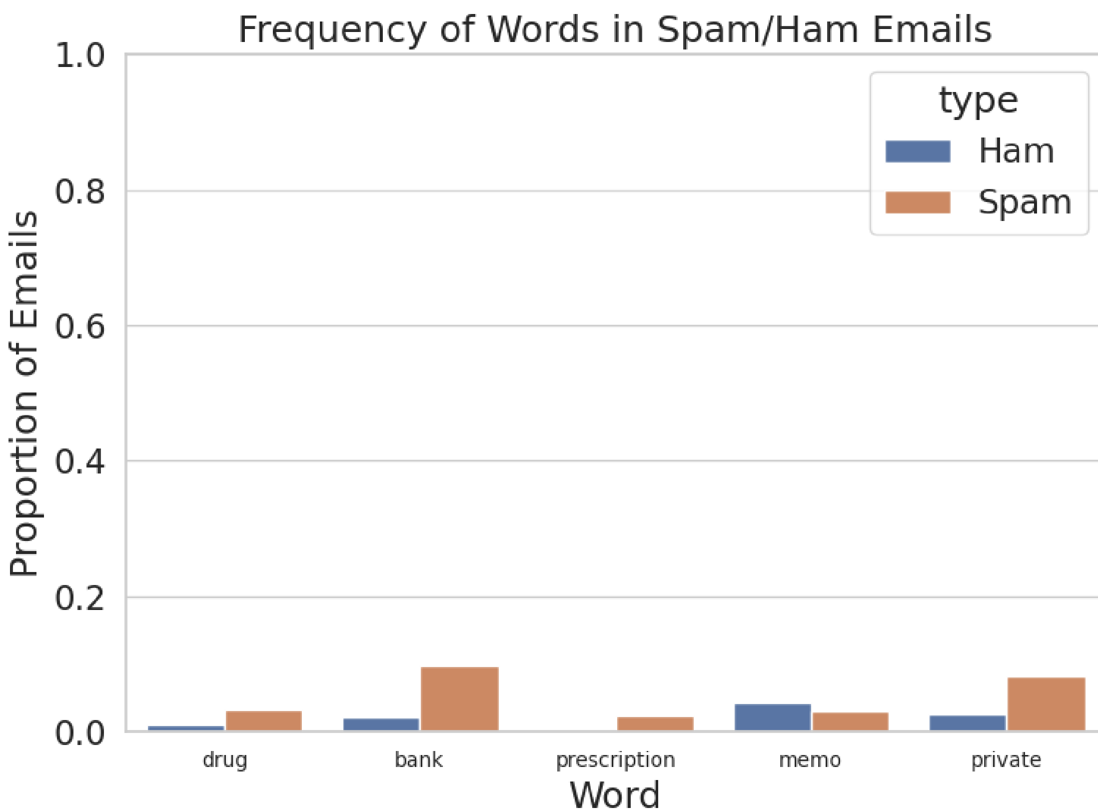
```
In [56]: some_words_in_ham = words_in_texts(some_words, train_ham_emails)
         summed_ham_some_words = np.sum(some_words_in_ham, axis=0)
         some_words_in_spam = words_in_texts(some_words, train_spam_emails)
         summed_spam_some_words = np.sum(some_words_in_spam, axis=0)
         prop_ham_some_words = summed_ham_some_words / len(train_ham_emails)
         prop_spam_some_words = summed_spam_some_words / len(train_spam_emails)
         some_words_ham_df = pd.DataFrame(prop_ham_some_words.reshape(1, -1), columns=some_words)
         some_words_ham_df['type'] = 'Ham'
         some_words_spam_df = pd.DataFrame(prop_spam_some_words.reshape(1, -1), columns=some_words)
         some_words_spam_df['type'] = 'Spam'
         somewords_df = pd.concat([some_words_ham_df, some_words_spam_df]).melt('type')
         somewords_df
```

Out[56]:

	type	variable	value
0	Ham	drug	0.010366
1	Spam	drug	0.033368
2	Ham	bank	0.021269
3	Spam	bank	0.097497
4	Ham	prescription	0.001609
5	Spam	prescription	0.023983
6	Ham	memo	0.044861
7	Spam	memo	0.030761
8	Ham	private	0.026631
9	Spam	private	0.082899

```
In [57]: train = train.reset_index(drop=True) # We must do this in order to preserve the ordering of em
         plt.figure(figsize=(8,6))
         sns.barplot(data=somewords_df, x='variable', y='value', hue='type')

         ax = plt.gca()
```

```
ax.set_ylim(0, 1)
ax.set_ylabel('Proportion of Emails')

ax.tick_params(axis='x', which='major', labelsize=10)
ax.set_xlabel('Word')

plt.title('Frequency of Words in Spam/Ham Emails')

plt.tight_layout()
plt.show()
```



When thinking about the prevalence of these words in an email, the graph alludes that they are not very prevalent. In regular Ham emails, these words appear very infrequently. Even in the Spam emails, the most prevalent word only appears in about 10% of the spam emails, with multiple words appearing much less. Thus, the words chosen for feature engineering for emails may not be optimal and may not be the best for the distribution of emails in our dataset, given that they show such little prevalence in both Ham and Spam emails.

0.5 Question 6h

Would you prefer to use the logistic regression classifier `my_model` or the zero predictor classifier for a spam filter? Why? Describe your reasoning and relate it to at least one of the evaluation metrics you have computed so far.

Given the nature of this problem, I would rather use the zero predictor. This is because the zero predictor has a similar accuracy as the logistic regression classifier while also zero false positives, which could be critical when dealing with emails. I would not want an important email to be misclassified as a spam email (false positive) with only a ~1% increase in accuracy when actually filtering out spam.

Thus, I would rather all emails be not marked at spam and have to filter out the spam manually than have to potentially look through the "Spam" folder for an actually important email that was misclassified.