Top Down Parser

In [1]:

```
import spacy
from spacy import displacy
```

In [2]:

```
nlp = spacy.load('en_core_web_sm')
```

In [3]:

```
text = 'She drove the Greek piano'
```

In [4]:

```
doc = nlp(text)
```

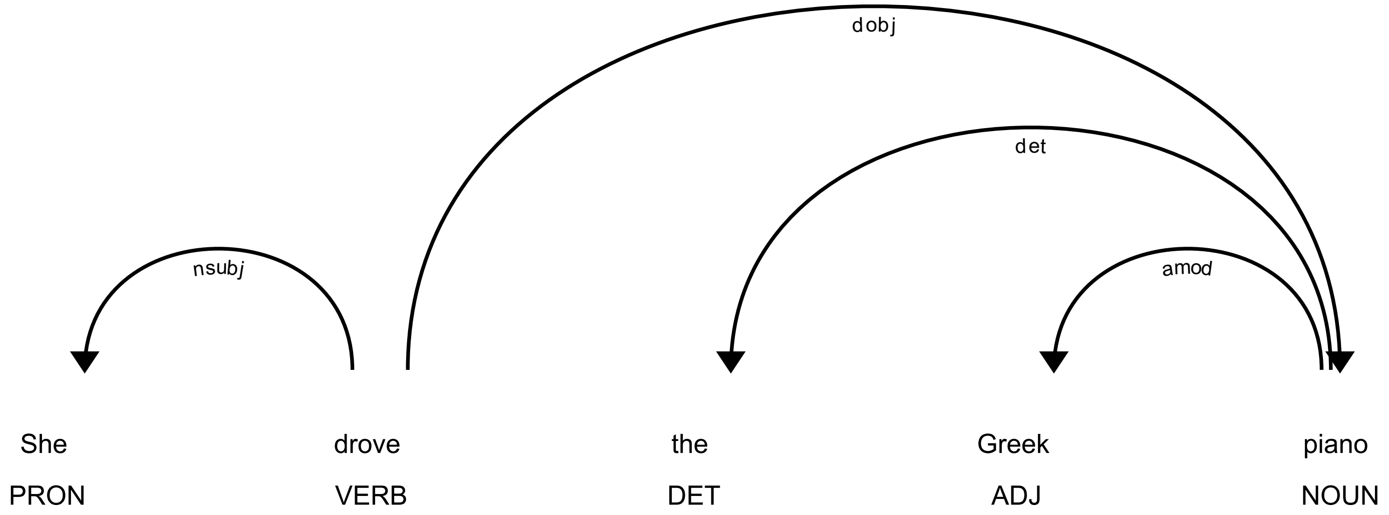
In [5]:

```
for token in doc:
    print(token.text, # the token is text
          token.dep_, # the token's dependency
          token.head.text, # the text of the token's father
          token.pos_, # the token's part of speech
          [child for child in token.children]) # the token's children
```

```
She nsubj drove PRON []
drove ROOT drove VERB [She, piano]
the det piano DET []
Greek amod piano ADJ []
piano dobj drove NOUN [the, Greek]
```

In [6]:

```
displacy.render(doc, style='dep', jupyter=True)
```



Refining parts-of-speech

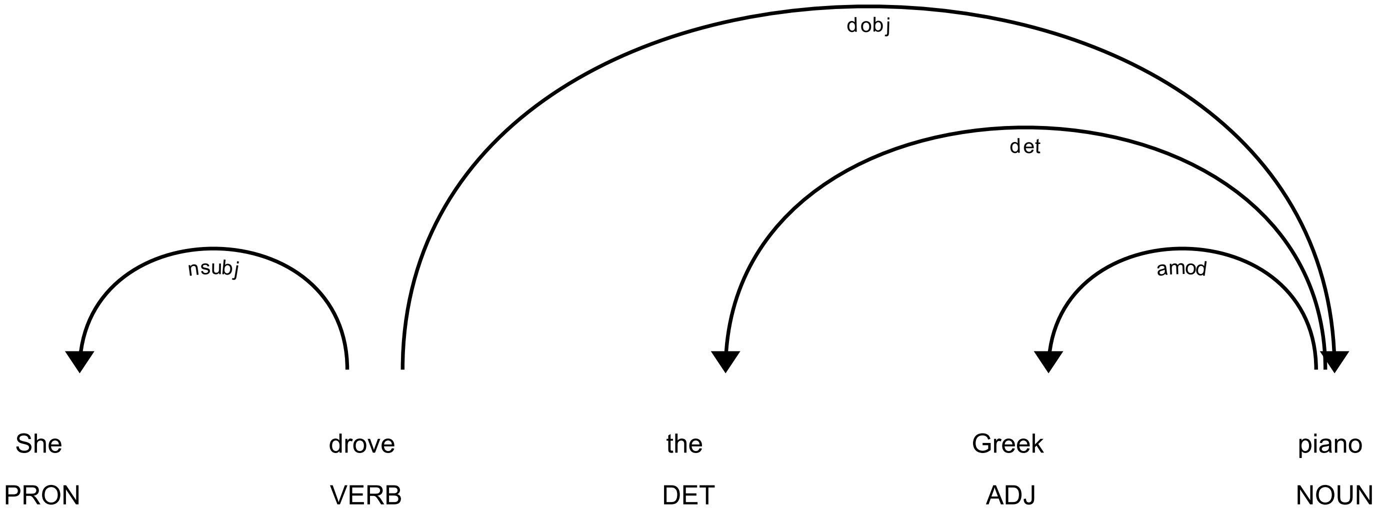
In [7]:

```
for token in doc:
    print(token.text,
          token.dep_,
          token.head.text,
          token.tag_,
          [child for child in token.children])
```

```
She nsubj drove PRP []
drove ROOT drove VBD [She, piano]
the det piano DT []
Greek amod piano JJ []
piano dobj drove NN [the, Greek]
```

In [8]:

```
displacy.render(doc, style='dep', jupyter=True, options = {"fine-grained":True})
```



In [9]:

```
text1 = 'The Queen of England drove the Greek piano'
```

In [10]:

```
doc1 = nlp(text1)
```

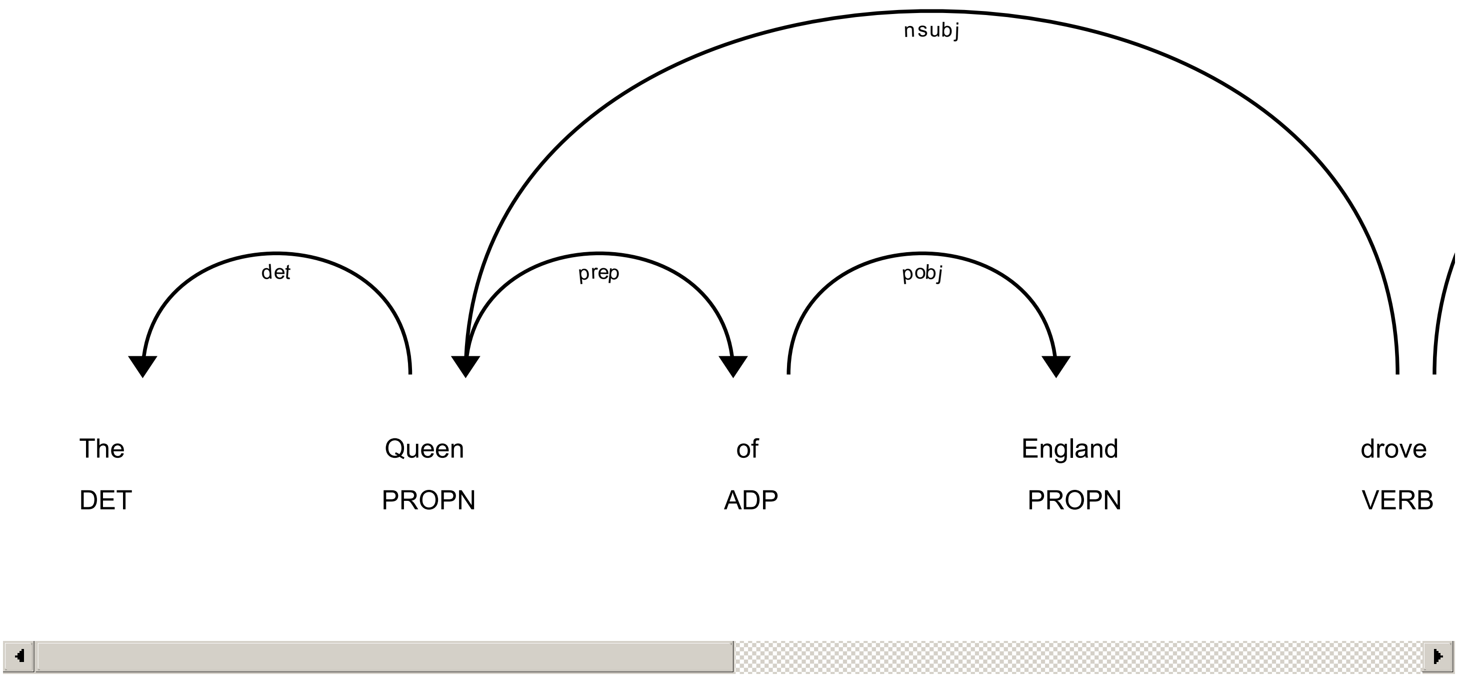
In [11]:

```
for token in doc:
    print(token.text,
          token.dep_,
          token.head.text)
```

```
She nsubj drove
drove ROOT drove
the det piano
Greek amod piano
piano dobj drove
```

In [12]:

```
displacy.render(doc1, style='dep', jupyter=True, options = {"fine-grained":True})
```



In [13]:

```
queen_token = doc[1] # queen is the 2nd token in the sentence

print(queen_token.left_edge.text) # This is the first word in the subtree
print(queen_token.right_edge.text)  # This is the last word in the subtree

span = doc1[queen_token.left_edge.i:queen_token.right_edge.i+1] # all words in the sente
nce

print(span)
```

```
She
piano
The Queen of England drove
```

In [14]:

```
text2 = 'Berfore the war, the King of Norway flew the Dutch Guitar'
doc2 = nlp(text2)

for token in doc2:
    if token.dep_=='nsubj':
        subtree = token.subtree
        break

print([(t.text, t.dep_, t.pos_) for t in subtree])
```

```
[('the', 'det', 'DET'), ('King', 'nsubj', 'PROPN'), ('of', 'prep', 'ADP'), ('Norway', 'po
bj', 'PROPN')]
```

In [15]:

```
import nltk
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
from nltk import pos_tag, word_tokenize, RegexpParser
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\Swapnil\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\Swapnil\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

In [16]:

```
sample_text = "The quick brown fox jumps over the lazy dog"

tagged = pos_tag(word_tokenize(sample_text))

chunker = RegexpParser("""
NP:{<DT>?<JJ>*<NN>} # To extract Noun Phrase
P:{<IN>} # To extract Prepositions
V:{<V.*>} # To extract Verbs
PP:{<p><NP>} # To extract prepositional phrases
VP:{<V><NP|PP>} # to extract verb phrases
""")
```

In [17]:

```
output = chunker.parse(tagged)
print("After Extracting\n", output)
```

```
After Extracting
 (S
  (NP The/DT quick/JJ brown/NN)
  (NP fox/NN)
  (V jumps/VBZ)
  (P over/IN)
  (NP the/DT lazy/JJ dog/NN))
```

In [18]:

```
# draws parse tree
output.draw()
```

In [ ]:

```

```