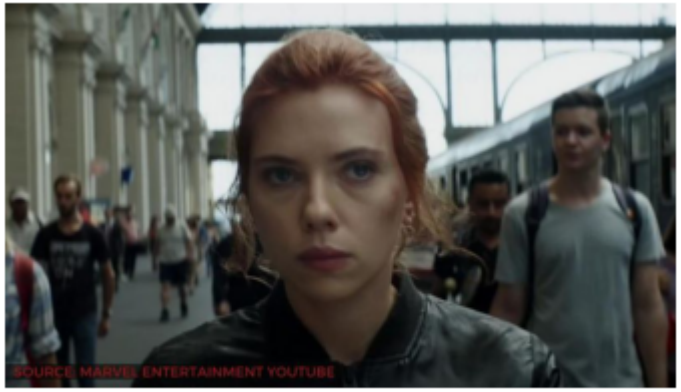
Analyzing an local image

In [3]:
```
import os
import sys
import requests
%matplotlib inline
import matplotlib.pyplot as plt
from PIL import Image
from io import BytesIO

subscription_key = "16d670a0ebd7461db042805430872e6f"
endpoint = "https://sjreddy.cognitiveservices.azure.com/"
analyze_url = endpoint + "vision/v3.1/analyze"

image_path = r"C:\Users\Jaswanth Reddy\Pictures\Images downloaded\black_widow.jpeg"

# Read the image into a byte array
image_data = open(image_path, "rb").read()
headers = {'Ocp-Apim-Subscription-Key': subscription_key,
           'Content-Type': 'application/octet-stream'}
params = {'visualFeatures': 'Categories,Description,Color'}
response = requests.post(
    analyze_url, headers=headers, params=params, data=image_data)
response.raise_for_status()

# The 'analysis' object contains various fields that describe the image. The most
# relevant caption for the image is obtained from the 'description' property.
analysis = response.json()
print(analysis)
```

```
{'categories': [{'name': 'people_portrait', 'score': 0.484375, 'detail': {'celebrities': [{'name': 'Scarlett J
ohansson', 'confidence': 0.9995917677879333, 'faceRectangle': {'left': 279, 'top': 149, 'width': 207, 'heigh
t': 207}}]}}], 'color': {'dominantColorForeground': 'White', 'dominantColorBackground': 'Black', 'dominantColo
rs': ['Grey', 'Black', 'White'], 'accentColor': '425157', 'isBwImg': False, 'isBWImg': False}, 'description':
{'tags': ['person'], 'captions': [{'text': 'Scarlett Johansson taking a selfie', 'confidence': 0.5668699145317
078}]}, 'requestId': '7ede61f3-cafe-4012-a625-57907f717af4', 'metadata': {'height': 464, 'width': 812, 'forma
t': 'Jpeg'}}
```

In [4]:
```
image_caption = analysis["description"]["captions"][0]["text"].capitalize()

# Display the image and overlay it with the caption.
image = Image.open(BytesIO(image_data))
plt.imshow(image)
plt.axis("off")
_ = plt.title(image_caption, size="x-large", y=-0.1)
plt.show()
```



Scarlett johansson taking a selfie

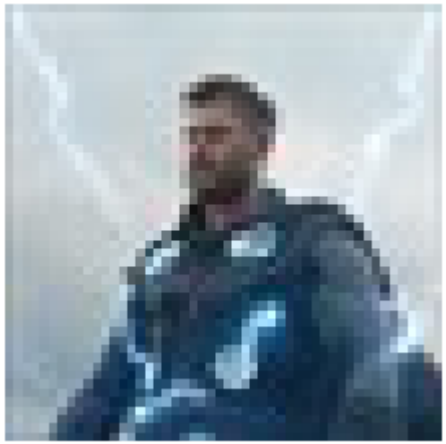
Generate a thumbnail

In [23]:
```
import os
import sys
import requests
import matplotlib.pyplot as plt
%matplotlib inline
from PIL import Image
from io import BytesIO

# Add your Computer Vision subscription key and endpoint to your environment variables.
subscription_key = "16d670a0ebd7461db042805430872e6f"
endpoint = "https://sjreddy.cognitiveservices.azure.com/"

thumbnail_url = endpoint + "vision/v3.1/generateThumbnail"

# Set image_url to the URL of an image that you want to analyze.
image_url = "https://i.guim.co.uk/img/media/933249d24608932fc897fcaa5e8c8bb2bdc9e977/124_0_1800_1080/master/1800


# Construct URL
headers = {'Ocp-Apim-Subscription-Key': subscription_key}
params = {'width': '50', 'height': '50', 'smartCropping': 'true'}
data = {'url': image_url}
# Call API
response = requests.post(thumbnail_url, headers=headers, params=params, json=data)
response.raise_for_status()

# Open the image from bytes
thumbnail = Image.open(BytesIO(response.content))

# Verify the thumbnail size.
print("Thumbnail is {0}-by-{1}".format(*thumbnail.size))

# Save thumbnail to file
thumbnail.save('thumbnail.png')

# Display image
thumbnail.show()
```

```
Thumbnail is 50-by-50
```

In [24]:
```
plt.imshow(thumbnail)
plt.axis("off")
```

Out[24]: (-0.5, 49.5, 49.5, -0.5)



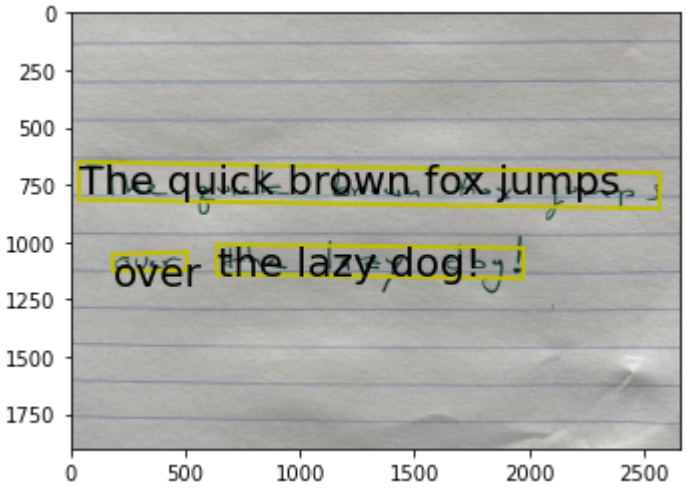
Extract text from images

In [31]:
```
import json
import os
import sys
import requests
import time
import matplotlib.pyplot as plt
from matplotlib.patches import Polygon
from PIL import Image
from io import BytesIO

subscription_key = "16d670a0ebd7461db042805430872e6f"

endpoint = "https://sjreddy.cognitiveservices.azure.com/"


text_recognition_url = endpoint + "/vision/v3.1/read/analyze"

# Set image_url to the URL of an image that you want to recognize.
image_url = "https://raw.githubusercontent.com/MicrosoftDocs/azure-docs/master/articles/cognitive-services/Compu

headers = {'Ocp-Apim-Subscription-Key': subscription_key}
data = {'url': image_url}
response = requests.post(
    text_recognition_url, headers=headers, json=data)
response.raise_for_status()

# Extracting text requires two API calls: One call to submit the
# image for processing, the other to retrieve the text found in the image.

# Holds the URI used to retrieve the recognized text.
operation_url = response.headers["Operation-Location"]

# The recognized text isn't immediately available, so poll to wait for completion.
analysis = {}
poll = True
while (poll):
    response_final = requests.get(
        response.headers["Operation-Location"], headers=headers)
    analysis = response_final.json()

    print(json.dumps(analysis, indent=4))
```

```
    time.sleep(1)
    if ("analyzeResult" in analysis):
        poll = False
    if ("status" in analysis and analysis['status'] == 'failed'):
        poll = False

polygons = []
if ("analyzeResult" in analysis):
    # Extract the recognized text, with bounding boxes.
    polygons = [(line["boundingBox"], line["text"])
                for line in analysis["analyzeResult"]["readResults"][0]["lines"]]
```


```
                            "text": "lazy",
                            "confidence": 0.981
                        },
                        {
                            "boundingBox": [
                                1661,
                                1021,
                                1974,
                                1029,
                                1976,
                                1156,
                                1663,
                                1158
                            ],
                            "text": "dog!",
                            "confidence": 0.559
                        }
                    ]
                }
            ]
```

In [32]:
```
# Display the image and overlay it with the extracted text.
image = Image.open(BytesIO(requests.get(image_url).content))
ax = plt.imshow(image)
for polygon in polygons:
    vertices = [(polygon[0][i], polygon[0][i+1])
                for i in range(0, len(polygon[0]), 2)]
    text = polygon[1]
    patch = Polygon(vertices, closed=True, fill=False, linewidth=2, color='y')
    ax.axes.add_patch(patch)
    plt.text(vertices[0][0], vertices[0][1], text, fontsize=20, va="top")
plt.show()
```



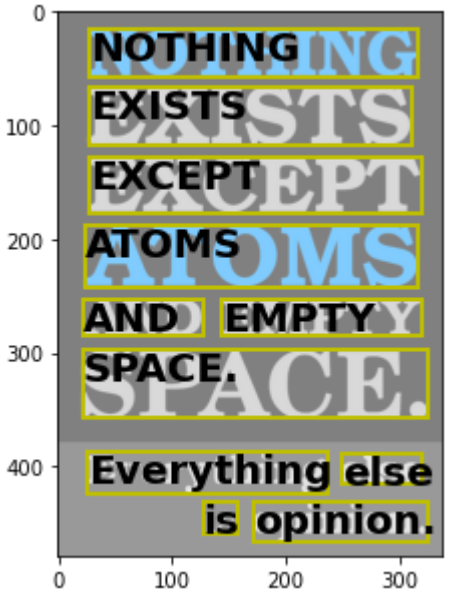
Extract printed text from an image

In [43]:
```
import os
import sys
import requests
# If you are using a Jupyter notebook, uncomment the following line.
# %matplotlib inline
import matplotlib.pyplot as plt
from matplotlib.patches import Rectangle
from PIL import Image
from io import BytesIO

subscription_key = "16d670a0ebd7461db042805430872e6f"

endpoint = "https://sjreddy.cognitiveservices.azure.com/"

ocr_url = endpoint + "vision/v3.1/ocr"

# Set image_url to the URL of an image that you want to analyze.
image_url = "https://upload.wikimedia.org/wikipedia/commons/thumb/a/af/" + \
    "Atomist_quote_from_Democritus.png/338px-Atomist_quote_from_Democritus.png"

headers = {'Ocp-Apim-Subscription-Key': subscription_key}
params = {'language': 'unk', 'detectOrientation': 'true'}
data = {'url': image_url}
response = requests.post(ocr_url, headers=headers, params=params, json=data)
response.raise_for_status()

analysis = response.json()

# Extract the word bounding boxes and text.
line_infos = [region["lines"] for region in analysis["regions"]]
word_infos = []
for line in line_infos:
    for word_metadata in line:
        for word_info in word_metadata["words"]:
            word_infos.append(word_info)
word_infos
```

Out[43]:
```
[{'boundingBox': '28,16,288,41', 'text': 'NOTHING'},
 {'boundingBox': '27,66,283,52', 'text': 'EXISTS'},
 {'boundingBox': '27,128,292,49', 'text': 'EXCEPT'},
```

```
{'boundingBox': '24,188,292,54', 'text': 'ATOMS'},
{'boundingBox': '22,253,105,32', 'text': 'AND'},
{'boundingBox': '144,253,175,32', 'text': 'EMPTY'},
{'boundingBox': '21,298,304,60', 'text': 'SPACE.'},
{'boundingBox': '26,387,210,37', 'text': 'Everything'},
{'boundingBox': '249,389,71,27', 'text': 'else'},
{'boundingBox': '127,431,31,29', 'text': 'is'},
{'boundingBox': '172,431,153,36', 'text': 'opinion.'}]
```

In [44]:
```
# Display the image and overlay it with the extracted text.
plt.figure(figsize=(5, 5))
image = Image.open(BytesIO(requests.get(image_url).content))
ax = plt.imshow(image, alpha=0.5)
for word in word_infos:
    bbox = [int(num) for num in word["boundingBox"].split(",")]
    text = word["text"]
    origin = (bbox[0], bbox[1])
    patch = Rectangle(origin, bbox[2], bbox[3],
                      fill=False, linewidth=2, color='y')
    ax.axes.add_patch(patch)
    plt.text(origin[0], origin[1], text, fontsize=20, weight="bold", va="top")
plt.show()
plt.axis("off")
```



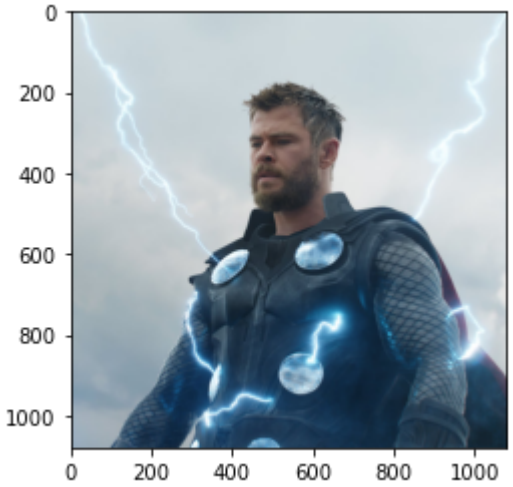
Out[44]: (0.0, 1.0, 0.0, 1.0)

```
# Describing image
```

In [62]:
```
import requests
import matplotlib.pyplot as plt
import os
import json
from PIL import Image
from io import BytesIO

# Add your Computer Vision subscription key and endpoint to your environment variables.

subscription_key = "16d670a0ebd7461db042805430872e6f"
analyze_url = "https://sjreddy.cognitiveservices.azure.com/vision/v3.0/analyze?%s"

# Set image_url to the URL of an image that you want to analyze.
image_url = "https://i.guim.co.uk/img/media/933249d24608932fc897fcaa5e8c8bb2bdc9e977/124_0_1800_1080/master/1800

headers = {'Ocp-Apim-Subscription-Key': subscription_key}
params = {'visualFeatures': 'Categories,Description,Color'}
data = {'url': image_url}
response = requests.post(analyze_url, headers=headers,
                         params=params, json=data)

print(json.dumps(response.json()))
image = Image.open(BytesIO(requests.get(image_url).content))
plt.imshow(image)
```

{"categories": [{"name": "people_", "score": 0.625, "detail": {"celebrities": [{"name": "Chris Hemsworth", "co
nfidence": 0.9203289747238159, "faceRectangle": {"left": 410, "top": 288, "width": 154, "height": 154}}]}}],
"color": {"dominantColorForeground": "White", "dominantColorBackground": "White", "dominantColors": ["White"],
"accentColor": "626369", "isBwImg": false, "isBWImg": false}, "description": {"tags": ["outdoor", "person", "m
an", "standing", "holding", "looking", "water", "riding", "ocean", "sitting", "wearing", "blue", "large", "you
ng", "surfing", "kite", "wave"], "captions": [{"text": "Chris Hemsworth looking at the camera", "confidence":
0.7414831937420129}]}, "requestId": "790b599f-f344-48aa-867c-f90dbc1e1858", "metadata": {"height": 1080, "widt
h": 1080, "format": "Jpeg"}}

Out[62]: <matplotlib.image.AxesImage at 0x2a65cadadf0>



Domain Specific(landmark identification)

In [59]:
```
import os
import sys
import requests
# If you are using a Jupyter notebook, uncomment the following line.
# %matplotlib inline
import matplotlib.pyplot as plt
from PIL import Image
from io import BytesIO

subscription_key = "16d670a0ebd7461db042805430872e6f"
endpoint = "https://sjreddy.cognitiveservices.azure.com/"

landmark_analyze_url = endpoint + "vision/v3.1/models/landmarks/analyze"

# Set image_url to the URL of an image that you want to analyze.
image_url = "https://upload.wikimedia.org/wikipedia/commons/f/f6/" + \
    "Bunker_Hill_Monument_2005.jpg"

headers = {'Ocp-Apim-Subscription-Key': subscription_key}
params = {'model': 'landmarks'}
data = {'url': image_url}
response = requests.post(
    landmark_analyze_url, headers=headers, params=params, json=data)
response.raise_for_status()

# The 'analysis' object contains various fields that describe the image. The
# most relevant landmark for the image is obtained from the 'result' property.
analysis = response.json()
assert analysis["result"]["landmarks"] is not []
print(analysis)
landmark_name = analysis["result"]["landmarks"][0]["name"].capitalize()

# Display the image and overlay it with the landmark name.
image = Image.open(BytesIO(requests.get(image_url).content))
plt.imshow(image)
plt.axis("off")
_ = plt.title(landmark_name, size="x-large", y=-0.1)
plt.show()
```

```
{'result': {'landmarks': [{'name': 'Bunker Hill Monument', 'confidence': 0.9788281321525574}]}, 'requestId':
'dc25f6e7-c4f4-4c2a-95c5-58c980150d11', 'metadata': {'height': 1600, 'width': 1200, 'format': 'Jpeg'}}
```

Bunker hill monument

Domain identification(Celebrity)

In [61]:
```
import requests
import matplotlib.pyplot as plt
from PIL import Image
from io import BytesIO
subscription_key = "16d670a0ebd7461db042805430872e6f"

endpoint = "https://sjreddy.cognitiveservices.azure.com/vision/v2.1/"

celebrity_analyze_url = endpoint + "models/celebrities/analyze"

# Set image_url to the URL of an image that you want to analyze.
image_url = "https://i.ytimg.com/vi/D0STjbmtMjY/maxresdefault.jpg"

headers = {'Ocp-Apim-Subscription-Key': subscription_key}
params = {'model': 'celebrities'}
data = {'url': image_url}
response = requests.post(
    celebrity_analyze_url, headers=headers, params=params, json=data)
response.raise_for_status()

# The 'analysis' object contains various fields that describe the image. The
# most relevant celebrity for the image is obtained from the 'result' property.
analysis = response.json()
assert analysis["result"]["celebrities"] is not []
print(analysis)
celebrity_name = analysis["result"]["celebrities"][0]["name"].capitalize()

# Display the image and overlay it with the celebrity name.
image = Image.open(BytesIO(requests.get(image_url).content))
plt.imshow(image)
plt.axis("off")
_ = plt.title(celebrity_name, size="x-large", y=-0.1)
plt.show()
```

```
{'result': {'celebrities': [{'name': 'Dwayne Johnson', 'confidence': 0.9999840259552002, 'faceRectangle': {'le
ft': 502, 'top': 76, 'width': 212, 'height': 212}}]}, 'requestId': '608a9b9f-4616-4c3b-b645-3262432c1fd9', 'me
tadata': {'height': 720, 'width': 1280, 'format': 'Jpeg'}}
```

Dwayne johnson

In [ ]: