In [15]:

```
import numpy as np
import json
import matplotlib.pyplot as plt
from tqdm import tqdm
import random
import subprocess
import time
import os
```

Read the data

In [5]:

```
# js_scripts.txt constains the paths to js files
with open("js_dataset/js_scripts.txt") as file:
    scripts = file.read().strip().split('\n')

# dirs_data.txt constains the names of directories in data directory of js 150 dataset
# we assume that different directories indicates different js apps
with open("js_dataset/dirs_data.txt") as file:
    dirs = file.read().strip().split('\n')
```

In [18]:
```
# group script paths by directories
scripts_by_dirs = []

for directory in tqdm(dirs):
    dir_scripts = []
    for script in scripts:
        if script.startswith("data/" + directory):
            dir_scripts.append(script)
    if len(dir_scripts):
        scripts_by_dirs.append(dir_scripts)
```

```
100%|██████████| 9620/9620 [06:50<00:00, 23.42it/s]
```

Perform compression

In [21]:
```
def get_seconds(time):
    min_ind = time.find('m')
    mins = int(time[:min_ind])
    second = float(time[min_ind + 1:-1])
    return mins * 60 + second

def log(file, msg):
    f = open(file, 'a+')
    f.write(msg + '\n')
    f.close()
```

In [25]:
```
rates_gzip = []
rates_brotli = []
times_gzip = []
times_brotli = []
speed_gzip = []
speed_brotli = []
init_sizes = []

for i in range(len(scripts_by_dirs)):

    #concatenate all scripts inside the directory to simulate web bundle
    script_concatenated = ""
    for url in scripts_by_dirs[i]:
        if url == "":
            continue
        if not os.path.exists("js_dataset/" + url):
            print("DOESN'T EXIST: ", url)
            continue
        try:
            with open("js_dataset/" + url) as file:
                script_concatenated += file.read()
        except:
            print("didn't read")
    rates_gzip_compressed = []
    rates_brotli_compressed = []
    times_gzip_compressed = []
    times_brotli_compressed = []
    speed_gzip_compressed = []
    speed_brotli_compressed = []

    with open("example2.txt", "w") as file:
        file.write(script_concatenated)
    size_non_compressed = os.stat("example2.txt").st_size
    init_sizes.append(size_non_compressed)

    # do the gzip compression with different levels
    for level in range(4, 10):
        result = subprocess.run(["bash", "gzip_compress.sh", str(le
vel), "time2.txt",
                                 "example_gzip2.txt.gz", "example2.
txt"])
        with open("time2.txt") as file:
            user_sys = file.read().strip().split('\n')[1:]
        time = get_seconds(user_sys[0].split('\t')[1]) + get_second
s(user_sys[1].split('\t')[1])
        size_gzip_compressed = os.stat("example_gzip2.txt.gz").st_s
ize
        rates_gzip_compressed.append(size_non_compressed / size_gzi
p_compressed)
        times_gzip_compressed.append(time)
        speed_gzip_compressed.append(size_non_compressed / time)

    # do the brotli compression with different levels
    for level in range(4, 12):
        result = subprocess.run(["bash", "brotli_compress.sh", str(
level), "time2.txt",
                                 "example_brotli2.txt.br", "example
2.txt"])
        with open("time2.txt") as file:
            user_sys = file.read().strip().split('\n')[1:]
        time = get_seconds(user_sys[0].split('\t')[1]) + get_second
s(user_sys[1].split('\t')[1])
        size_br_compressed = os.stat("example_brotli2.txt.br").st_s
ize
        rates_brotli_compressed.append(size_non_compressed / size_b
r_compressed)
        times_brotli_compressed.append(time)
        speed_brotli_compressed.append(size_non_compressed / time)

    rates_gzip.append(rates_gzip_compressed)
    rates_brotli.append(rates_brotli_compressed)
    times_gzip.append(times_gzip_compressed)
    times_brotli.append(times_brotli_compressed)
    speed_gzip.append(speed_gzip_compressed)
    speed_brotli.append(speed_brotli_compressed)
    if i != 0 and i % 500 == 0:
        log("logs4.txt", "rates_gzip: " + str(np.mean(rates_gzip, axis=0)))
        log("logs4.txt", "rates_brotli: " + str(np.mean(rates_brotli, axis=0)))
        log("logs4.txt", "times_gzip: " + str(np.mean(times_gzip, axis=0)))
        log("logs4.txt", "times_brotli: " + str(np.mean(times_brotli, axis=0)))
        log("logs4.txt", "speed_gzip: " + str(np.mean(speed_gzip, axis=0)))
        log("logs4.txt", "speed_brotli: " + str(np.mean(speed_brotli, axis=0)))
```

In [27]:

```
import pandas as pd
frame = pd.DataFrame()
frame["name"] = ["gzip 4", "gzip 5", "gzip 6", "gzip 7", "gzip 8", "gzip 9",
                 "brotli 4", "brotli 5", "brotli 6", "brotli 7", "brotli 8", "brotli 9", "brotli 10", "brotli 11"]

frame["rates"] = np.hstack((np.mean(rates_gzip, axis=0), np.mean(rates_brotli, axis=0)))
frame["savings"] = 1 - 1 / np.hstack((np.mean(rates_gzip, axis=0), np.mean(rates_brotli, axis=0)))
frame["speed(MB/s)"] = np.hstack((np.mean(speed_gzip, axis=0), np.mean(speed_brotli, axis=0))) / 1000000

frame
```

Out[27]:

	name	rates	savings	speed(MB/s)
0	gzip 4	3.825069	0.738567	15.719552
1	gzip 5	3.948932	0.746767	13.392738
2	gzip 6	4.003179	0.750199	10.956911
3	gzip 7	4.017695	0.751101	9.777660
4	gzip 8	4.029332	0.751820	7.136008
5	gzip 9	4.031706	0.751966	6.170267
6	brotli 4	4.135726	0.758204	12.866184
7	brotli 5	4.496571	0.777608	9.528445
8	brotli 6	4.543836	0.779922	8.582947
9	brotli 7	4.582319	0.781770	6.631221
10	brotli 8	4.599897	0.782604	5.447145
11	brotli 9	4.622002	0.783644	4.209170
12	brotli 10	4.930100	0.797164	1.157362
13	brotli 11	5.019602	0.800781	0.506957

In [46]:

```python
print("non compressed size range {}MB-{}MB".format(np.min(init_sizes) / 1000000, np.max(init_sizes)/ 1000000))
```

```
non compressed size range 0.0MB-519.170072MB
```

Group results by non compressed size ranges

In [49]:

```
splits = [0, 100000, 1000000, 519170072]
init_sizes = np.array(init_sizes)
group1 = np.where((init_sizes >= 0)*(init_sizes <= 100000))[0]
group2 = np.where((init_sizes > 100000)*(init_sizes <= 1000000))[0]
group3 = np.where((init_sizes > 1000000)*(init_sizes <= 519170072))[0]

print(0, "-", 100000, "bytes")
frame = pd.DataFrame()
frame["name"] = ["gzip 4", "gzip 5", "gzip 6", "gzip 7", "gzip 8", "gzip 9",
                 "brotli 4", "brotli 5", "brotli 6", "brotli 7", "brotli 8", "brotli 9", "brotli 10", "brotli 11"]

frame["rates"] = np.hstack((np.mean(np.array(rates_gzip)[group1], axis=0), np.mean(np.array(rates_brotli)[group1], axis=0)))
frame["savings"] = 1 - 1 / np.hstack((np.mean(np.array(rates_gzip)[group1], axis=0), np.mean(np.array(rates_brotli)[group1], axis=0)))
frame["speed(MB/s)"] = np.hstack((np.mean(np.array(speed_gzip)[group1], axis=0), np.mean(np.array(speed_brotli)[group1], axis=0))) / 1000000

frame
```

```
0 - 100000 bytes
```

Out[49]:

	name	rates	savings	speed(MB/s)
0	gzip 4	3.580008	0.720671	7.447231
1	gzip 5	3.676672	0.728015	7.008153
2	gzip 6	3.712879	0.730667	6.428090
3	gzip 7	3.723238	0.731417	6.065006
4	gzip 8	3.730148	0.731914	5.120283
5	gzip 9	3.731493	0.732011	4.732681
6	brotli 4	3.694064	0.729295	5.004788
7	brotli 5	4.011637	0.750725	4.648579
8	brotli 6	4.033570	0.752081	4.471990
9	brotli 7	4.049876	0.753079	3.971136
10	brotli 8	4.056882	0.753505	3.708456
11	brotli 9	4.065070	0.754002	3.146465
12	brotli 10	4.318749	0.768451	1.005612
13	brotli 11	4.426846	0.774106	0.470691

In [50]:

```
print(100000, "-", 1000000, "bytes")
frame = pd.DataFrame()
frame["name"] = ["gzip 4", "gzip 5", "gzip 6", "gzip 7", "gzip 8", "gzip 9",
                 "brotli 4", "brotli 5", "brotli 6", "brotli 7", "brotli 8", "brotli 9", "brotli 10", "brotli 11"]

frame["rates"] = np.hstack((np.mean(np.array(rates_gzip)[group2], axis=0), np.mean(np.array(rates_brotli)[group2], axis=0)))
frame["savings"] = 1 - 1 / np.hstack((np.mean(np.array(rates_gzip)[group2], axis=0), np.mean(np.array(rates_brotli)[group2], axis=0)))
frame["speed(MB/s)"] = np.hstack((np.mean(np.array(speed_gzip)[group2], axis=0), np.mean(np.array(speed_brotli)[group2], axis=0))) / 1000000

frame
```

```
100000 - 1000000 bytes
```

Out[50]:

	name	rates	savings	speed(MB/s)
0	gzip 4	4.610515	0.783104	40.486917
1	gzip 5	4.821605	0.792600	32.909052
2	gzip 6	4.927023	0.797038	25.098103
3	gzip 7	4.953874	0.798138	21.498278
4	gzip 8	4.976779	0.799067	13.639378
5	gzip 9	4.981861	0.799272	10.864447
6	brotli 4	5.086900	0.803417	35.662622
7	brotli 5	5.540047	0.819496	24.545289
8	brotli 6	5.629082	0.822351	21.281608
9	brotli 7	5.707098	0.824780	14.285831
10	brotli 8	5.742195	0.825851	10.013604
11	brotli 9	5.777223	0.826906	6.719082
12	brotli 10	6.213230	0.839053	1.659122
13	brotli 11	6.359796	0.842762	0.617029

In [51]:

```python
print(1000000, "-", 519170072, "bytes")
frame = pd.DataFrame()
frame["name"] = ["gzip 4", "gzip 5", "gzip 6", "gzip 7", "gzip 8", "gzip 9",
                 "brotli 4", "brotli 5", "brotli 6", "brotli 7", "brotli 8", "brotli 9", "brotli 10", "brotli 11"]

frame["rates"] = np.hstack((np.mean(np.array(rates_gzip)[group3], axis=0), np.mean(np.array(rates_brotli)[group3], axis=0)))
frame["savings"] = 1 - 1 / np.hstack((np.mean(np.array(rates_gzip)[group3], axis=0), np.mean(np.array(rates_brotli)[group3], axis=0)))
frame["speed(MB/s)"] = np.hstack((np.mean(np.array(speed_gzip)[group3], axis=0), np.mean(np.array(speed_brotli)[group3], axis=0))) / 1000000

frame
```

```
1000000 - 519170072 bytes
```

Out[51]:

	name	rates	savings	speed(MB/s)
0	gzip 4	4.947584	0.797881	62.609464
1	gzip 5	5.195765	0.807536	47.516889
2	gzip 6	5.366891	0.813672	33.629151
3	gzip 7	5.405544	0.815005	27.704937
4	gzip 8	5.458839	0.816811	16.148692
5	gzip 9	5.468953	0.817150	12.309037
6	brotli 4	8.551782	0.883065	61.243214
7	brotli 5	9.349877	0.893047	35.094220
8	brotli 6	9.716333	0.897081	29.873129
9	brotli 7	10.018952	0.900189	23.305218
10	brotli 8	10.169293	0.901665	18.596231
11	brotli 9	10.418385	0.904016	13.694826
12	brotli 10	11.215429	0.910837	1.773174
13	brotli 11	10.618584	0.905825	0.704833