Dosya | C:/Users/Evren/Desktop/hw1/chap4.html

Data Analysis

Minimum correlation = 0.12

90 journal entries and 6 events
weekend: 0.1372 (weekend occurs in 30 entries)
brushed teeth: -0.3805 (brushed teeth occurs in 64 entries)
candy: 0.1296 (candy occurs in 6 entries)
work: -0.1372 (work occurs in 60 entries)
spaghetti: 0.2425 (spaghetti occurs in 9 entries)
peanuts: 0.5903 (peanuts occurs in 13 entries)

---

1. fonksiyonda sincaba dönüşmüş olduğu eventleri filtreledim. Sonrasında bu eventlerin item ve indexlerini map() ve forEach() kullanarak listeledim.

2. fonksiyonda bir array oluşturdum. forEach ile JOURNAL da gezip indexlerin 5 ile modunu aldığımda 0 elde ettiğim eventleri arrayin içine pushladım.

Sample code

```
function analyze(min = 0) {
  return [...EVENTS]
      .map(e => ({ evt: e, cor: phi(tableFor(e)) }))
      .filter(x => Math.abs(x.cor) > min)
      .map(x => {
          let table = tableFor(x.evt);
          return x.evt + ': ' + x.cor.toFixed(4) + ' (' + x.evt + ' occurs in ' + (table[1] + table[3]) + ' entries)';
      });
}
function journalEvents() {
  EVENTS.clear();
  for (let entry of JOURNAL)
    for (let e of entry.events)
      EVENTS.add(e);
}
```

Ref: Chap 4. Final analysis

Öğeler | Konsol | Kaynaklar | Ağ | Performans | Bellek | Uygulama

top | Filtre | Varsayılan seviyeler | Sorun Yok

```
> let a = JOURNAL.filter(x => x.squirrel === true)
      .map(x -> x.events.forEach((item, index) -> console.log(`item: ${item}, index:
  ${index}`)));
```

```
item: spaghetti, index: 0                                   VM31:2
item: peanuts, index: 1                                     VM31:2
item: computer, index: 2                                    VM31:2
item: weekend, index: 3                                     VM31:2
item: lasagna, index: 0                                     VM31:2
item: peanuts, index: 1                                     VM31:2
item: work, index: 2                                        VM31:2
item: pizza, index: 0                                       VM31:2
item: peanuts, index: 1                                     VM31:2
item: candy, index: 2                                       VM31:2
item: work, index: 3                                        VM31:2
item: spaghetti, index: 0                                   VM31:2
item: peanuts, index: 1                                     VM31:2
item: exercise, index: 2                                    VM31:2
item: weekend, index: 3                                     VM31:2
item: carrot, index: 0                                      VM31:2
item: peanuts, index: 1                                     VM31:2
item: reading, index: 2                                     VM31:2
item: weekend, index: 3                                     VM31:2
< undefined
```

```
> array = [];
  let b = JOURNAL.forEach((item, index) => {
      if (index % 5 === 0) {
          array.push(item.squirrel);
      }
  });
  console.log(array);
```

```
                                                            VM37:7
▶ (18) [false, false, false, false, false, true, false, false, false, false, false, false, false, false, false, false, false, false]
< undefined
>
```

Exercises

The sum of a range

The introduction of this book alluded to the following as a nice way to compute the sum of a range of numbers:

```
console.log(sum(range(1, 10)));
```

Write a `range` function that takes two arguments, `start` and `end`, and returns an array containing all the numbers from `start` up to (and including) `end`.

Next, write a `sum` function that takes an array of numbers and returns the sum of these numbers. Run the example program and see whether it does indeed return 55.

As a bonus assignment, modify your `range` function to take an optional third argument that indicates the "step" value used when building the array. If no step is given, the elements go up by increments of one, corresponding to the old behavior. The function call `range(1, 10, 2)` should return `[1, 3, 5, 7, 9]`. Make sure it also works with negative step values so that `range(5, 2, -1)` produces `[5, 4, 3, 2]`.

```
function range(start, end, step = 1) {
  const myArray = [];
  if (step > 0) {
    for (let i = start; i <= end; i += step) {
      myArray.push(i);
    }
  } else {
    for (let i = start; i >= end; i += step) {
      myArray.push(i);
    }
  }
  return myArray;
}
console.log(range(1, 10, 2));
console.log(range(5, 2, -1));

```

```
[1, 3, 5, 7, 9]
[5, 4, 3, 2]
```