Functions and Higher-order Functions
Homework Assignment

1. Use arrow notation (=>) to define a function called `square` which takes a number as argument and returns the square of that number.
2. Use arrow notation (=>) to define a function called `cube` which takes a number as argument and returns the cube of that number.
3. Use arrow notation (=>) to define a function called `perfectsquares` which takes a number `N` as argument and returns the first `N` perfect squares.
   - Make use of the `square` function you implemented in (1), as well as the `each` and `range` functions provided on the next slide.
4. Use arrow notation (=>) to define a function called `perfectcubes` which takes a number `N` as argument and returns the first `N` perfect cubes.
   - Make use of the `cube` function you implemented in (2), as well as the `each` and `range` functions provided on the next slide.
5. Define a function called `perfectpowers` which takes two numbers `N` and `P` as arguments and returns the first `N` perfect powers of `P`.
   - You can make use of the `Math.pow` function or implement your own *power* function. Make use of the `each` and `range` functions provided on the next slide.

Functions and Higher-order Functions
Homework Assignment

- The function `each` below is a higher-order function (it takes a function as argument) that takes an array `A` and a function `func` as arguments. It applies the function `func` to each item in `A` and returns the resulting array.

```
function each(A, func) {
    for (var i = 0; i < A.length; i++) {
        A[i] = func(A[i]);
    }
    return A;
}
```

You can test the function by trying: `console.log(each([1,2,3], square))`

Which should print: `[1,4,9]`

- The function `sequence` below takes a number `N` as argument and it returns an array with the numbers `[1, ... , N]`

```
function sequence(N) {
    return Array(N).fill().map((_, idx) => idx+1)
}
```

You can test the function by trying : `console.log(sequence(5))`

Which should print: `[1, 2, 3, 4, 5]`