

2721. Execute Asynchronous Functions in Parallel

Description

Given an array of asynchronous functions `functions`, return a new promise `promise`. Each function in the array accepts no arguments and returns a promise. All the promises should be executed in parallel.

`promise` resolves:

- When all the promises returned from `functions` were resolved successfully in parallel. The resolved value of `promise` should be an array of all the resolved values of promises in the same order as they were in the `functions`. The `promise` should resolve when all the asynchronous functions in the array have completed execution in parallel.

`promise` rejects:

- When any of the promises returned from `functions` were rejected. `promise` should also reject with the reason of the first rejection.

Please solve it without using the built-in `Promise.all` function.

Example 1:

```
Input: functions = [
  () => new Promise(resolve => setTimeout(() => resolve(5), 200))
]
Output: {"t": 200, "resolved": [5]}
Explanation:
promiseAll(functions).then(console.log); // [5]
```

The single function was resolved at 200ms with a value of 5.

Example 2:

```
Input: functions = [
  () => new Promise(resolve => setTimeout(() => resolve(1), 200)),
  () => new Promise((resolve, reject) => setTimeout(() => reject("Error"), 100))
]
Output: {"t": 100, "rejected": "Error"}
Explanation: Since one of the promises rejected, the returned promise also rejected with the same error at the same time.
```

Example 3:

```
Input: functions = [
  () => new Promise(resolve => setTimeout(() => resolve(4), 50)),
  () => new Promise(resolve => setTimeout(() => resolve(10), 150)),
  () => new Promise(resolve => setTimeout(() => resolve(16), 100))
]
Output: {"t": 150, "resolved": [4, 10, 16]}
Explanation: All the promises resolved with a value. The returned promise resolved when the last promise resolved.
```

Constraints:

- `functions` is an array of functions that returns promises
- `1 <= functions.length <= 10`

