Basic Promise

A promise is simply an object that represents an async task. It can either resolve (success) or reject (failure).

```
let myPromise = new Promise((resolve, reject) => {
  let success = true;

  if (success) {
    resolve("✅ Task completed!");
  } else {
    reject("❌ Something went wrong!");
  }
});

myPromise
  .then(result => console.log(result))
  .catch(error => console.log(error));
```

JS

Promise.resolve()

This instantly creates a promise that is already resolved with a given value. I use it when I don't want to write the full new Promise() syntax.

```
Promise.resolve("Resolved instantly!")
  .then(msg => console.log(msg));
```

Promise.reject()

Same as resolve, but it directly creates a rejected promise. Helpful when I want to simulate or handle errors quickly.

```
Promise.reject("Rejected instantly!")
  .catch(err => console.log(err));
```

Promise.all()

It runs multiple promises together and only succeeds if all of them succeed. If even one fails, the whole thing fails.

```
let p1 = Promise.resolve("First done");
let p2 = Promise.resolve("Second done");

Promise.all([p1, p2])
  .then(result => console.log("All done:", result))
  .catch(err => console.log(" Error:", err));
```

Output: All done: ['First done', 'Second done']

Promise.allSettled()

Unlike all(), this one doesn’t fail if something rejects. It gives me the final status of every promise, whether it’s resolved or rejected.

```
let p1 = Promise.resolve("Done 1");
let p2 = Promise.reject("Failed 2");

Promise.allSettled([p1, p2])
  .then(results => console.log(results));
```

Output:
```
[
 { status: 'fulfilled', value: 'Done 1' },
 { status: 'rejected', reason: 'Failed 2' }
]
```

Promise.race()

This returns the result of whichever promise settles first. Doesn’t matter if it resolves or rejects — the first one wins.

```
let p1 = new Promise(resolve => setTimeout(resolve, 100, "Slow"));
let p2 = new Promise(resolve => setTimeout(resolve, 50, "Fast"));

Promise.race([p1, p2])
  .then(result => console.log("Winner:", result));
```

Output: Winner: Fast

Promise.any()

This is like a positive version of race(). It waits for the first successful promise. If all fail, only then it throws an error.

```
let p1 = Promise.reject("Fail 1");
let p2 = Promise.resolve("Success 2");
let p3 = Promise.reject("Fail 3");

Promise.any([p1, p2, p3])
  .then(result => console.log("First success:", result))
  .catch(err => console.log("All failed:", err));
```

Output: First success: Success 2