Promise.all, Promise.race, Promise.allSettled

Here's a **detailed explanation** of `Promise.all`, `Promise.race`, and `Promise.allSettled` in JavaScript:

🔗 Purpose

These are **combinator methods** used to handle multiple promises concurrently. They allow you to run multiple asynchronous operations and control how their results are aggregated.

🔷 1. `Promise.all()`

📌 Description:

Waits for **all promises** in the iterable to be **fulfilled**.

If **any promise rejects**, it immediately rejects.

✅ Syntax:

```
Promise.all([promise1, promise2, promise3])
```

✅ Example:

```
const p1 = Promise.resolve("One");
const p2 = Promise.resolve("Two");
const p3 = Promise.resolve("Three");

Promise.all([p1, p2, p3])
  .then((values) => {
    console.log(values); // ["One", "Two", "Three"]
```

```
})
.catch((err) => {
  console.error(err); // Won't trigger here
});
```

❌ Rejection Example:

```
const p1 = Promise.resolve("Success");
const p2 = Promise.reject("Failed");
const p3 = Promise.resolve("Still works");

Promise.all([p1, p2, p3])
  .then(console.log)
  .catch(console.error); // Output: "Failed"
```

🔷 2. Promise.race()

📌 Description:

Returns the **first settled promise** (fulfilled or rejected) out of a group.

✅ Syntax:

```
Promise.race([promise1, promise2, promise3])
```

✅ Example:

```
const slow = new Promise((res) => setTimeout(res, 2000, "Slow"));
const fast = new Promise((res) => setTimeout(res, 1000, "Fast"));

Promise.race([slow, fast])
  .then(console.log); // Output: "Fast"
```

❌ Rejection Wins Too:

```
const p1 = new Promise((_, rej) => setTimeout(rej, 500, "Error"));
const p2 = new Promise((res) => setTimeout(res, 1000, "Done"));

Promise.race([p1, p2])
  .then(console.log)
  .catch(console.error); // Output: "Error"
```

🔷 3. Promise.allSettled()

📌 Description:

Waits for **all promises** to settle (either resolve or reject), and returns an array of **results with status**.

✅ Syntax:

```
Promise.allSettled([promise1, promise2, promise3])
```

✅ Example:

```
const p1 = Promise.resolve("Success");
const p2 = Promise.reject("Failed");
const p3 = Promise.resolve("Done");

Promise.allSettled([p1, p2, p3])
  .then((results) => {
    console.log(results);
  });
```

✅ Output:

```
[
  { status: "fulfilled", value: "Success" },
  { status: "rejected", reason: "Failed" },
```

```
  { status: "fulfilled", value: "Done" }
]
```

📊 Comparison Table

Feature	`Promise.all()`	`Promise.race()`	`Promise.allSettled()`
Waits for all?	✅ Yes	❌ No (only the first settles)	✅ Yes
Rejects if one fails	✅ Immediately	✅ If first is reject	❌ Never (returns status per promise)
Use case	Wait for all to succeed	Compete (e.g., timeout vs task)	Collect results regardless of success

🧠 Real-world Use Cases

- `Promise.all()`:
  - Load multiple APIs in parallel and process together.
- `Promise.race()`:
  - Timeout fallback: Race actual request with a timeout.
- `Promise.allSettled()`:
  - Log the result of multiple uploads, even if some fail.

Let me know if you'd like to see real-world API examples for these!