24. Promise.all 、 allSettled 、race 、any 用法对比

1. 核心概念 (Core Concept)

`Promise.all`，`Promise.allSettled`，`Promise.race`，`Promise.any` 是 JavaScript 中 `Promise` 原型上的静态方法，它们都接收一个 Promise 可迭代对象（如数组）作为输入，并返回一个新的 Promise，用于处理多个 Promise 并发执行的结果集合。它们的主要区别在于如何处理输入 Promise 的解决(fulfilled)或拒绝(rejected)状态。

2. 为什么需要它？ (The "Why")

在异步编程中，我们经常需要并发执行多个任务（每个任务可能是一个 Promise），并根据这些任务的执行结果进行后续处理。这些静态方法提供了标准化的、声明式的方式来处理这些并发场景，避免了手动管理多个 Promise 状态的复杂性，提高了代码的可读性和可维护性。

3. API 与用法 (API & Usage)

这四个方法都接收一个可迭代对象（通常是 Promise 数组）作为参数，并返回一个新的 Promise。

- **`Promise.all(iterableOfPromises)`**
  - **描述:** 等待所有输入的 Promise 都解决（fulfill）或其中任何一个 Promise 拒绝（reject）。
  - **返回值:**
    - 如果所有 Promise 都解决，则返回一个解决状态的 Promise，其值是一个包含所有解决值的数组，顺序与输入的 Promise 顺序一致。
    - 如果任何一个 Promise 拒绝，则返回一个拒绝状态的 Promise，其值为第一个拒绝的 Promise 的拒绝原因。
  - **经典示例 (摘自 MDN):**

```
const promise1 = Promise.resolve(3);
const promise2 = 42; // 非 Promise 值会被解释为 Promise.resolve(42)
const promise3 = new Promise((resolve, reject) => {
  setTimeout(resolve, 100, 'foo');
});

Promise.all([promise1, promise2, promise3]).then((values) => {
  console.log(values);
  // Expected output: Array [3, 42, "foo"]
});

const promise4 = Promise.resolve(10);
```

```
const promise5 = Promise.reject('Error occurred');

Promise.all([promise4, promise5]).catch((error) => {
    console.error(error);
    // Expected output: Error occurred
});
```

- `Promise.allSettled(iterableOfPromises)`
  - **描述:** 等待所有输入的 Promise 都达到 settled 状态（即解决或拒绝）。
  - **返回值:** 返回一个解决状态的 Promise，其值为一个包含所有输入 Promise 结果对象的数组。每个结果对象具有 `status` (值可以是 `"fulfilled"` 或 `"rejected"`) 和 `value` (如果状态是 `"fulfilled"`) 或 `reason` (如果状态是 `"rejected"`) 属性。
  - **经典示例 (摘自 MDN):**

```
const promise1 = Promise.resolve(3);
const promise2 = new Promise((resolve, reject) => setTimeout(reject,
100, 'foo'));
const promises = [promise1, promise2];

Promise.allSettled(promises).then((results) =>
results.forEach((result) => console.log(result.status)));
// Expected output:
// "fulfilled"
// "rejected"
```

- `Promise.race(iterableOfPromises)`
  - **描述:** 等待第一个输入的 Promise 改变状态（解决或拒绝）。
  - **返回值:** 返回一个 Promise，其状态和值（或原因）与第一个改变状态的输入 Promise 相同。
  - **经典示例 (摘自 MDN):**

```
const promise1 = new Promise((resolve, reject) => {
  setTimeout(resolve, 500, 'one');
});
const promise2 = new Promise((resolve, reject) => {
  setTimeout(resolve, 100, 'two');
});

Promise.race([promise1, promise2]).then((value) => {
  console.log(value);
  // Both resolve, but promise2 is faster
});
// Expected output: "two"
```

- `Promise.any(iterableOfPromises)`

- **描述:** 等待第一个输入的 Promise 解决（fulfill）。如果所有 Promise 都拒绝，则返回一个聚合错误（AggregateError）。
- **返回值:**
  - 如果任何一个 Promise 解决，则返回一个解决状态的 Promise，其值为第一个解决的 Promise 的值。
  - 如果所有 Promise 都拒绝，则返回一个拒绝状态的 Promise，其原因为一个 `AggregateError` 实例，该实例包含所有拒绝 Promise 的原因。
- **经典示例 (摘自 MDN):**

```
const promise1 = Promise.reject(0);
const promise2 = new Promise((resolve) => setTimeout(resolve, 100,
'quick'));
const promise3 = new Promise((resolve) => setTimeout(resolve, 500,
'slow'));

Promise.any([promise1, promise2, promise3]).then((value) =>
console.log(value));
// Expected output: "quick"

const promise4 = Promise.reject('Error A');
const promise5 = Promise.reject('Error B');

 Promise.any([promise4, promise5]).catch((err) => {
   console.log(err);
   console.log(err.errors);
 });
 // Expected output:
 // AggregateError: All promises were rejected
 // Array [ "Error A", "Error B" ]
```

4. 关键注意事项 (Key Considerations)

- **输入非 Promise 值:** 所有这些方法在接收到非 Promise 值时，会将其立即包裹成一个已解决的 Promise（例如 `Promise.resolve(value)`）处理。
- **`Promise.all` 的失败快速:** `Promise.all` 的一个主要特点是“失败快速”（fail fast）。只要有一个 Promise 拒绝，整个 `Promise.all` 返回的 Promise 就会立即拒绝，不会等待其他 Promise 完成。
- **`Promise.allSettled` 的完整结果:** `Promise.allSettled` 总是会等待所有 Promise 结束，无论成功或失败，并提供每个 Promise 的详细结果信息。这使得它非常适合并行执行多个独立任务，即使部分失败也需要获取所有结果的场景。
- **`Promise.any` 的成功优先:** `Promise.any` 专注于找到第一个成功的 Promise。只有在所有 Promise 都失败的情况下才会拒绝，并且提供一个聚合了所有失败原因的错误。它适用于需要从多个可能的来源中获取第一个可用结果的场景。

- **`Promise.race` 的竞速:** `Promise.race` 只关心第一个改变状态的 Promise，无论是成功还是失败。它常用于设置任务超时，通过让一个会超时拒绝的 Promise 与目标 Promise 竞争，来判断目标是否在规定时间内完成。

5. 参考资料 (References)

- MDN Web Docs - Promise
- MDN Web Docs - Promise.all()
- MDN Web Docs - Promise.allSettled()
- MDN Web Docs - Promise.race()
- MDN Web Docs - Promise.any()