New Promise methods

There have been new additions to the promise arsenal that I think warrant a deeper look so we're ready to use them when they are stable enough to do so.

What we started with

promise.all returns a single Promise that resolves when all of the promises passed resolve. It rejects with the reason of the first promise that rejects.

Promise.race returns a promise that fulfills or rejects as soon as one of the promises in an iterable fulfills or rejects, with the value or reason from that promise.

What we got recently

Promise.finally

Promise.finally ensures that code will run once, regardless of the promess status (settled or rejected). This will make sure that any cleanup code will happen and that developers don't need to remember to put the code in mutliple places.

In the examples below, rather than having to put the `hideLoadingSpinner` method in the then and catch blocks like first example.

```
const fetchAndDisplay = ({ url, element }) => {
  showLoadingSpinner();
  fetch(url)
    .then((response) => response.text())
    .then((text) => {
      element.textContent = text;
      hideLoadingSpinner();
    })
    .catch((error) => {
      element.textContent = error.message;
      hideLoadingSpinner();
    });
  };

fetchAndDisplay({
  url: someUrl,
  element: document.querySelector('#output')
});
```

We can leverage the `finally` method and place it there, knowing that it will run regardless of how the promise settles and hide the spinner.

```
const fetchAndDisplay = ({ url, element }) => {
  showLoadingSpinner();
  fetch(url)
    .then((response) => response.text())
    .then((text) => {
      element.textContent = text;
    })
    .catch((error) => {
      element.textContent = error.message;
    })
    .finally(() => {
      hideLoadingSpinner();
    });
};
```

We can also use the async/await to do the same thing with the full try/catch/finally blocks; taking into account that we still want to use hideLoadingSpinner only once.

```
const fetchAndDisplay = async (url) => {
  showLoadingSpinner();
  try {
    const response = await fetch(url);
    const text = await response.text();
    element.textContent = text;
  } catch (error) {
    element.textContent = error.message;
  } finally {
    hideLoadingSpinner();
  }
};
```

The new and shiny

There are two new methods of the promise object that are making their way through TC39 process. `promise.allSettled` is at stage 3 and `promise.any` is at stage 1 of the TC39 process.

`promise.allSettled`

Promise.allSettled returns a promise that is fulfilled with an array of promise state snapshots, but only after all the original promises have settled, i.e. become either fulfilled or rejected.

A common use case for this combinator is wanting to take an action after multiple requests have completed, regardless of their success or failure. Other Promise combinators (`promise.all` and `promise.race`) can short-circuit, discarding the results of input values that lose the race to reach a certain state.

Promise.allSettled will always waiting for all of its input values.

Here we are only interested in the promises which failed, and thus collect the reasons. allSettled allows us to do this.

```
const promises = [
  fetch('index.html'),
  fetch('https://does-not-exist/')
];

const results = await Promise.allSettled(promises);
const errors = results
  .filter(p => p.status === 'rejected')
  .map(p => p.reason);
```

Promise.any

Promise.any accepts an iterable of promises and returns a promise that is fulfilled by the first given promise to be fulfilled, or rejected with an array of rejection reasons if all of the given promises are rejected.

This is different than `promise.race` and `promise.all` in that only one promise has to succeed for the promise to fulfill (unlike promise.all) but they all must fail for the promise to reject.

```
Promise.any([
  fetch('https://example.org')
    .then(() => 'home'),
  fetch('https://web.dev')
    .th'home'=> 'web dev'),
  fetch('https://mdn.neet')
    .then(() => 'docs')
]).then((first) => {
  'web dev'// Any of the promises was fulfilled.e.log(first);
  // → 'home'
}).catch'home'r) => {
  // All of the promises were rejected.
  console.log(// All of the promises were rejected.
  console.log(error);
});
```