the shortcut

Fullstack Web Development Tutorial Lesson 17

Today's lesson will cover

- Promises chaining
- Promise static methods
- Async/await

JavaScript fundamentals

Promises chaining

- The idea is that the result is passed through the chain of `.then` handlers. The whole thing works, because a call to `promise.then` returns a promise, so that we can call the next `.then` on it.
- When a handler returns a value, it becomes the result of that promise, so the next `.then` is called with it.
- **A classic newbie error: technically we can also add many `.then` to a single promise. This is not chaining.**
- In practice we rarely need multiple handlers for one promise. Chaining is used much more often.

Promise static methods

- There are 5 static methods of `Promise` class:
  - `Promise.all(promises)` – waits for all promises to resolve and returns an array of their results. If any of the given promises rejects, it becomes the error of `Promise.all`, and all other results are ignored.
  - `Promise.allSettled(promises)` (recently added method) – waits for all promises to settle and returns their results as an array of objects with:
    - `status`: `"fulfilled"` or `"rejected"`
    - `value` (if fulfilled) or `reason` (if rejected).
  - `Promise.race(promises)` – waits for the first promise to settle, and its result/error becomes the outcome.
  - `Promise.resolve(value)` – makes a resolved promise with the given value.
  - `Promise.reject(error)` – makes a rejected promise with the given error.
- Of these five, `Promise.all` is probably the most common in practice.

Async/Await

- Async functions always return a promise. If the return value of an async function is not explicitly a promise, it will be implicitly wrapped in a promise.
- Syntax
  - ```
    async function name([param[, param[, ...param]]]) {
       Statements // await mechanism may be used
    }
    ```
- Example
  - ```
    async function asyncFunction() {
          const result1 = await new Promise((resolve) => setTimeout(()resolve('1')))
          const result2 = await new Promise((resolve) => setTimeout(()resolve('2')))
    }
    asyncFunction()
    ```
- The keyword `await` makes JavaScript wait until that promise settles and returns its result. It's just a more elegant syntax of getting the promise result than `promise.then`, easier to read and write.
- When we use `async/await`, we rarely need `.then`, because `await` handles the waiting for us. And we can use a regular `try..catch` instead of `.catch`. That's usually (but not always) more convenient.

the shortcut

Self Study Assignments

To Dos

- Create a game of Rock, Paper and Scissors using JS which works on console, or with interactive UI using HTML, CSS and JS however you prefer ***(If you are working on your own project where you are using JS already, feel free to ignore this task but please share the project update with Lena.)***
- Continue freecodecamp (FCC) Javascript. Ideally finish before we resume after summer.
- Continue with FCC HTML, CSS lessons. Ideally finish all the lessons by end of this month.
- If you believe FCC exercises aren't the best for you if you are quite advanced already, please start working on your own project and reach out to mentors for help if needed.