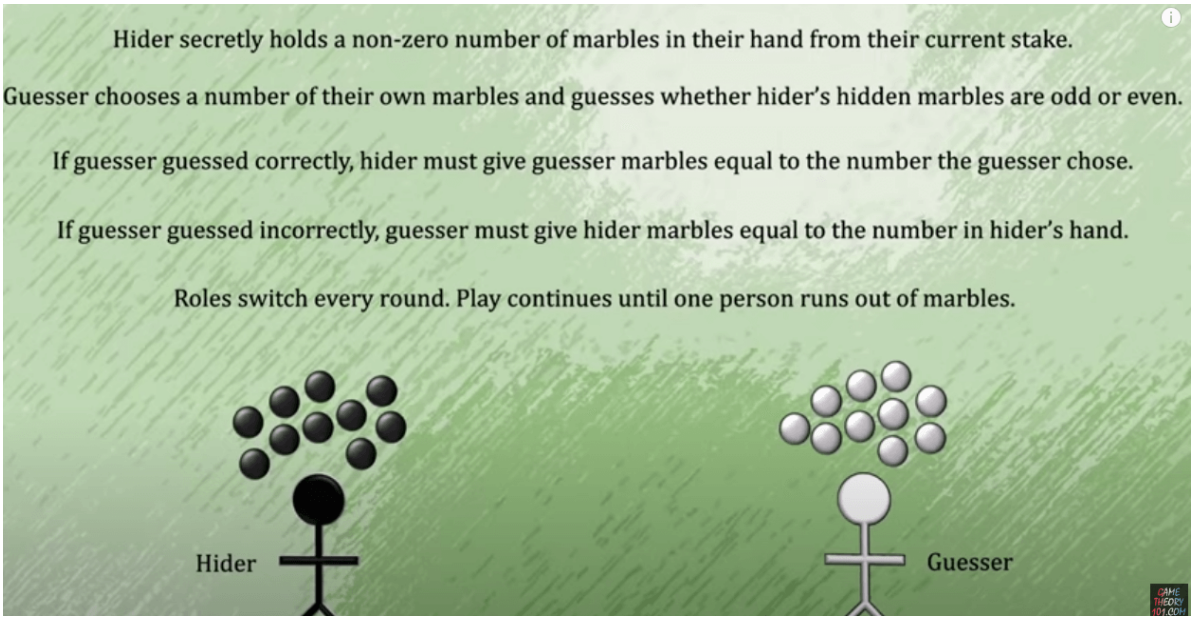
چالش بازی مرکب - تیله

بازی تیله سریال squid game رو که همه میشناسند. قراره یه custom hook بنویسیم که منطق بازی رو برامون اجرا کنه.



خلاصه بازی به این صورت است که در هر نوبت هر کدوم از بازیکن‌ها یکسری تیله از کیسه تیله‌هاشون که فقط در ابتدا شامل ۱۰ است انتخاب میکنند.
بازیکن دیگه guesser باید حدس بزنه که آیا بازیکن دیگه hider تعداد تیله‌هاش فرد هست یا زوج. اگه درست بگه تعداد تیله‌هایی که خودش gusesser توی مشتش قایم کرده بوده رو از طرف مقابل hider میگیره. ولی اگه اشتباه بگه باید به اندازه تیله‌هایی توی دست حریفش بوده hider به حریفش تیله بده.
توجه کنید که تعویض نوبت نخواهیم داشت و صرفا شما حدس میزنید. و این بازی تا زمانی ادامه داره که تیله‌های شما تموم بشه یا تعداد تیله‌هاتون به ۲۰ برسه.
آموزش بازی به زبان انگلیسی ۲ دقیقه اول این ویدیو را تماشا کنید.

ورودی‌های مسئله:

تابع validate: این تابع که به صورت async خواهد بود و وظیفه ارزیابی پاسخ شما را دارد. می‌توانید به آن به چشم request به سرور نگاه کنید.
سینکتس تابع validate به صورت زیر است.

- isOdd: آیا

```
async function validate(isOdd, betCount) {
    // اگر ورودی شما یک ارزش صحیح غلط نباشد. خطا بر میگرداند
    if (typeof isOdd !== 'boolean') throw new Error('MUST_BE_BOOLEAN')

    // در غیر این صورت یک عدد به شما بر میگرده.
    return number
}
```

در خروجی تابع validate

- اگر درست پیش بینی کرده باشید به اندازه‌ای که شرط بسته از حریفش تیله میگیره. به اندازه betCount.
- اگر اشتباه پیش بینی کرده باشید تعداد تیله‌هایی که حریفتون توی مشتش قایم کرده بوده به صورت منفی به شما برگردونده میشه تا از تیله‌هاتون کمش کنید. مثلا عددی مانند 3- بهتون

برگردونده میشه.

هوک شما باید به صورت زیر تعریف شده باشد و اعمال خواسته شده در زیر را انجام دهد.

```
function useSquidGame(valitate) {

  return {
      register: (inputType) => ({
        type: inputType,
        ref,
        onChange,
        disabled,
      }),
      submit: {
          onClick,
          disabled
      },
      gameStatus,
      history,
  }
}
```

نحوه‌ی استفاده از تابع useSquidGame را میتوانید در مثال زیر ببنید

```

function Game(validate) {
  const {
    register,
    submit,
    gameStatus,
    isLoading,
    history
  } = useSquidGame(validate);

  if (isLoading) return <span>loading ... </span>;
  if (gameStatus !== "IN_PROGRESS") return <span>{gameStatus}</span>;

  return (
    <div>
      <label>
        Marble bet count:
        <input {...register("number")} />
      </label>

      <label>
        is Odd:
        <input {...register("checkbox")} />
      </label>

      <button {...submit}>submit</button>
      <pre>
        <code>{JSON.stringify(history, null, 2)}</code>
      </pre>
```

```
    </div>
  );
}
```

هوک شما باید توابع خواسته شده را تعریف کند و با ساختار نمایش داده شده برگرداند.

- register: تابع register باید برای دو تایپ ورودی number و checkbox دو مجموعه پراپس جدا گانه برگرداند که به عنوان handler های input ها استفاده میشوند. توجه کنید که ref و onChange همان توابع و prop های معمول هستند.
- onChange: تابع تغییر برای دستیابی به تعداد تیله‌های شرطبندی شده و تیک (چک باکس) که نشان‌دهنده فرد بودن انتخاب حریف شماست.
اگر تیک زده باشد به منظور پیش بینی اینکه حریف شما تعدادی فرد در دست دارد، خواهد بود.
- submit: با زدن دکمه ارسال چندین کار صورت میگیرد
  - تابع validate با ورودی‌های مناسب فراخوانی شده و وضعیت بعدی برنامه را به صورت async دریافت خواهید کرد.
  - در زمانی که ارزیابی صورت میگیرد دکمه submit باید disabled باشد.
  - تا وقتی که در نوبت خود عددی انتخاب نکرده باشید نیز دکمه submit باید disabled باشد.
  - پس از بازگشت مقدار validate وضعیت history بروز رسانی شود به این صورت که تعداد تیله‌های هر فرد در یک ارایه دو تایی که اولی نشان دهنده تیله‌های باقی مانده شما و دومی نشان دهنده تعداد تیله‌های باقی مانده حریف شماست خواهد بود.
  برای مثال اگر در حرکت اول شما ۵ تیله از دست بدهید ارایه زیر به عنوان اولین عنصر به history اضافه میشود. `[5 ,15]`
  - توجه کنید که پس از ارزیابی validate باید input متنی تایپ number را خالی کنید. نیازی به ریست کردن تایپ checkbox نیست.

توجه کنید که رفرنس تابع onChange و onClick نباید بیهوده عوض شده و تنها در مواقع نیاز رفرنس تابع را عوض کنید.

- gameStatus: یکی از سه ارزش زیر که در حین بازی طبق طبق تصمیم validate تغییر میکند. `'LOOSE' | 'WIN' | 'IN_PROGRESS'`. در ابتدای بازی باید مقدار 'IN_PROGRESS' را داشته باشد.
- history: تاریخچه ورودی‌های کاربر که در حین بازی بروز می‌شود.

توجه کنید: که اگر بیشتر از تیله‌های طرف مقابل درخواست شود تمام تیله‌هایی که داشته دا از دست میدهد. مثلا اگر شما قرار باشد که از حریف خود ۵ تیله بگیرید اما او فقط ۳ تیله داشته باشد. تمام آن ۳ تیله را به شما داده و بازی با برد شما پایان می‌یابد.

راهنمایی انتظار میرود که عوض شدن ورودی هوک را در نظر گرفته و تمامی قسمت‌هایی که نیاز به بروز رسانی دارند را بروز رسانی کنید.

کد خود را میتوانید در این لینک اجرا و ارزیابی کنید.