Lab 10.1 - Synchronous Error Handling

The native **`URL`** constructor can be used to parse URLs, it's been wrapped into a function called **`parseURL`**:

```
function parseURL (str) {
  const parsed = new URL(str)
  return parsed
}
```

If **`URL`** is passed a unparsable URL string it will throw, so calling **`parseURL('foo')`** will result in an exception:


```
2. bash
$ cat example.js
function parseUrl (str) {
  const parsed = new URL(str)
  return parsed
}

parseUrl('foo')
$ node example.js
internal/url.js:243
  throw error;
  ^

TypeError [ERR_INVALID_URL]: Invalid URL: foo
    at onParseError (internal/url.js:241:17)
    at new URL (internal/url.js:319:5)
    at parseUrl (/training/ch-10/example.js:2:18)
    at Object.<anonymous> (/training/ch-10/example.js:6:1)
    at Module._compile (internal/modules/cjs/loader.js:778:30)
    at Object.Module._extensions..js (internal/modules/cjs/loader.js:789:10)
    at Module.load (internal/modules/cjs/loader.js:653:32)
    at tryModuleLoad (internal/modules/cjs/loader.js:593:12)
    at Function.Module._load (internal/modules/cjs/loader.js:585:3)
    at Function.Module.runMain (internal/modules/cjs/loader.js:831:12)
$
```

The labs-1 folder contains an **`index.js`** file with the following content:

```
'use strict'
const assert = require('assert')

function parseUrl (str) {
  const parsed = new URL(str)
  return parsed
}

assert.doesNotThrow(() => { parseUrl('invalid-url') })
assert.equal(parseUrl('invalid-url'), null)
assert.deepEqual(
  parseUrl('http://example.com'),
  new URL('http://example.com')
)
console.log('passed!')
```


Modify the **`parseURL`** function body such that instead of throwing an error, it returns **`null`** when URL is invalid. Use the fact that **`URL`** will throw when given invalid input to determine whether or not to return **`null`** or a parsed object.

Once implemented, execute the **`index.js`** file with node, if the output says `passed!` then the exercise was completed successfully:

```
2. bash
$ node index.js
passed!
$
```

