Vamos ver uma lista de alguns erros comuns e como o Node.js nos avisa deles:

**ENOTFOUND**

*Error: not found* - Retorna quando o Node.js tenta estabelecer uma conexão com um servidor e a tentativa falha no *DNS lookup*; ou seja, ou o host não existe ou há algum erro no endereço fornecido, que não consegue ser resolvido pelo DNS.

Algumas das soluções possíveis: verifique se a URL fornecida realmente existe e se a string com a URL está sendo fornecida da forma correta para a função ou método que fará a conexão.

**ENOENT**

*Error: no entity* - Como vimos no curso, muitas vezes acontece quando um caminho especificado, seja um diretório ou um arquivo, não existe no sistema de arquivos. Neste caso, temos que verificar se o caminho fornecido está correto.

Importante: algumas operações com fs requerem o uso do **caminho absoluto** de um arquivo ou diretório. Para “montar” a string com estes caminhos, você pode utilizar uma das libs mais comuns do Node.js, a [path](https://nodejs.org/api/path.html).

**EISDIR**

*Error: is a directory* - O caminho fornecido é um diretório. Normalmente vemos este erro quando o programa esperava receber um caminho de um arquivo, mas ao invés disso recebeu um diretório. Novamente, devemos verificar a string recebida pela função ou método para conferir se está completa e montada da forma correta.

**ENOTDIR**

*Error: not a directory* - é o contrário do erro anterior, quando o programa espera receber o caminho de um diretório mas recebe o caminho de um arquivo.

**EADDRINUSE**

*Error: address already in use* - Muitas vezes este erro acontece quando estamos iniciando ou reiniciando um servidor web. Ele indica que o servidor está tentando se conectar a uma porta lógica que já está sendo ocupada por outro programa.

Você pode conferir mais sobre servidores web e portas de conexão em nosso curso de [fundamentos do HTTP](https://cursos.alura.com.br/course/http-fundamentos).

**ECONNREFUSED**

*Error: connection refused* - Houve uma tentativa de envio de requisição a um endpoint, porém a conexão foi recusada. Normalmente é causada por inatividade do serviço que está sendo requisitado.

**ECONNRESET**

*Error: connection reset* - Uma conexão em andamento foi fechada durante o processo de requisição-resposta, antes que a resposta fosse recebida. Pode ser causada por um timeout ou reinício do servidor.

A lista completa de erros pode ser conferida [aqui](https://nodejs.org/api/errors.html), direto na documentação do Node.js.