When receiving a POST or PUT request, the request body might be important to your application. Getting at the body data is a little more involved than accessing request headers. The request object that's passed in to a handler implements the [ReadableStream](https://nodejs.org/api/stream.html#stream_class_stream_readable) interface. This stream can be listened to or piped elsewhere just like any other stream. We can grab the data right out of the stream by listening to the stream's 'data' and 'end' events.

The chunk emitted in each 'data' event is a [Buffer](https://nodejs.org/api/buffer.html). If you know it's going to be string data, the best thing to do is collect the data in an array, then at the 'end', concatenate and stringify it.

let body = [];

request.on('data', (chunk) => {

body.push(chunk);

}).on('end', () => {

body = Buffer.concat(body).toString();

// at this point, `body` has the entire request body stored in it as a string

});

## Explicitly Sending Header Data

The methods of setting the headers and status code that we've already discussed assume that you're using "implicit headers". This means you're counting on node to send the headers for you at the correct time before you start sending body data.

If you want, you can explicitly write the headers to the response stream. To do this, there's a method call writeHead, which writes the status code and the headers to the stream.

response.writeHead(200, {

'Content-Type': 'application/json',

'X-Powered-By': 'bacon'

});

Once you've set the headers (either implicitly or explicitly), you're ready to start sending response data.

**Use Conditional Requests**

Conditional requests are HTTP requests which are executed differently depending on specific HTTP headers. We can think of these headers as preconditions: if they are met, the requests will be executed in a different way.

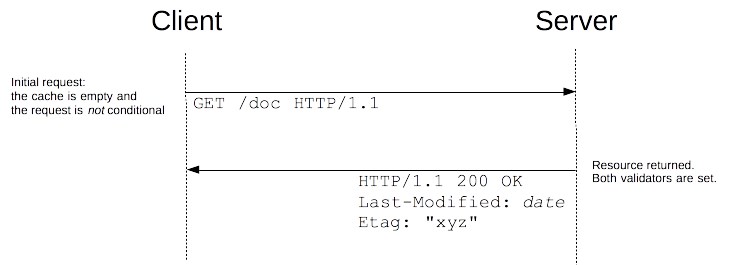
These headers try to check whether a version of a resource stored on the server matches a given version of the same resource. Because of this reason, these headers can be:

* the timestamp of the last modification,
* or an entity tag, which differs for each version.

These headers are:

* Last-Modified *(to indicate when the resource was last modified)*,
* Etag *(to indicate the entity tag)*,
* If-Modified-Since *(used with the Last-Modified header)*,
* If-None-Match *(used with the Etag header)*,

The client below did not have any previous versions of the doc resource, so neither the If-Modified-Since, nor the If-None-Match header was applied when the resource was sent. Then, the server responds with the Etag and Last-Modified headers properly set.

* 

The client can set the If-Modified-Since and If-None-Match headers once it tries to request the same resource – since it has a version now. If the response would be the same, the server simply responds with the 304 - Not Modified status and does not send the resource again.

