ExpressJS Request & Response Request both are the callback function parameters and are used for Express.js and Node.js. You can get the request query, params, body, headers, and cookies. It can overwrite any value or anything there. However, overwriting headers or cookies will not affect the output back to the browser.

The request object represents the HTTP request and contains properties for the request query string, parameters, body, HTTP headers, etc.

S.No	Properties	Description
1	req.app	Used to hold a reference to the instance of the express application.
2	req.body	Contains key-value pairs of data submitted in the request body. By default, it is undefined and is populated when you use body-parsing middleware such as body-parser.
3	req.cookies	This property contains cookies sent by the request, used for the cookie-parser middleware.
4	req.ip	req.ip is remote IP address of the request.
5	req.path	req.path contains the path part of the request URL.
6	req.route	req.route is currently-matched route.

The response object specifies the HTTP response when an Express app gets an HTTP request. The response is sent back to the client browser and allows you to set new cookies value that will write to the client browser.

S.No	properties	Description
1	res.app	res.app is hold a reference to the instance of the express application that is using the middleware.
2	res.locals	Specify an object that contains response local variables scoped to the request.

res.redirect

Redirects to the URL derived from the specified path, with specified status, a positive integer that corresponds to an HTTP status code . If not specified, status defaults to “302 “Found”.

res.json([body])

Sends a JSON response. This method sends a response (with the correct content-type) that is the parameter converted to a JSON string using JSON.stringify().

res.send()

This method performs many useful tasks for simple non-streaming responses: For example, it automatically assigns the Content-Length HTTP response header field (unless previously defined) and provides automatic HEAD and HTTP cache freshness support.

Difference between res.json vs res.send in Express.js

To sum things up, res.json() allows for extra formatting of the JSON data - if this is not required res.send() can also be used to return a response object using Express. Both of these methods also end the response correctly, and there's no further action required.