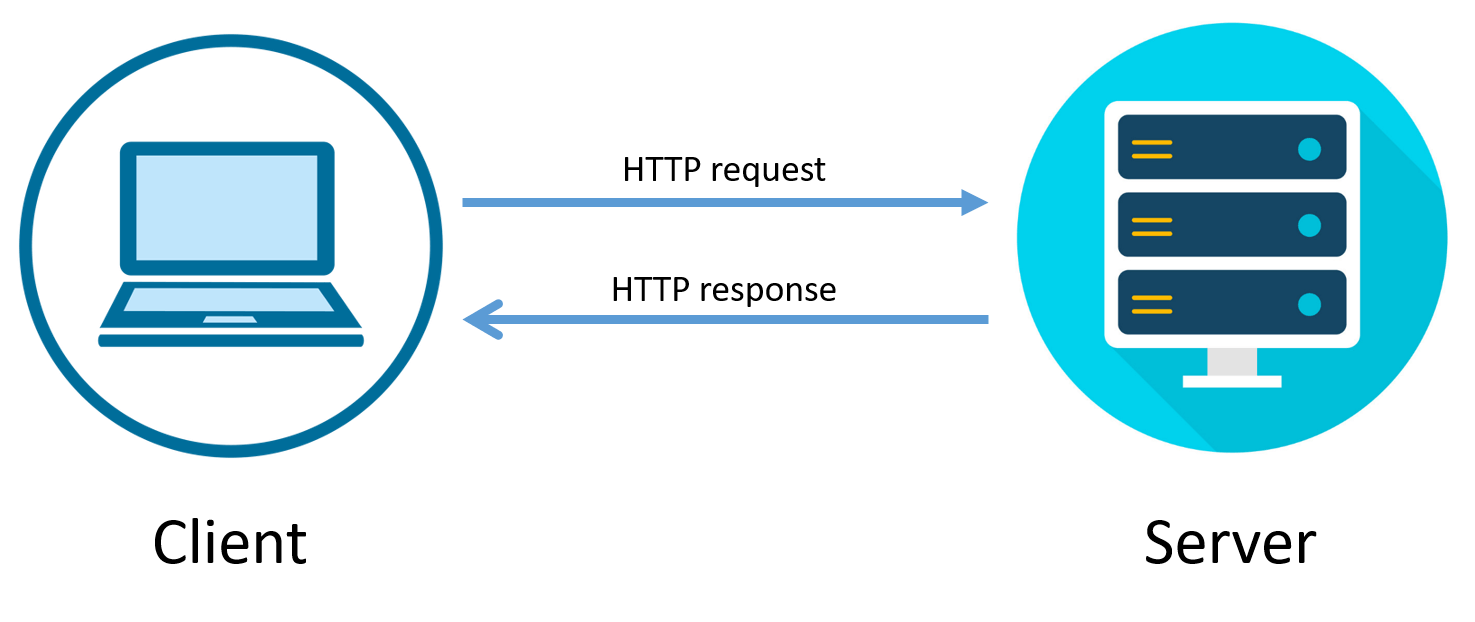
**Request - Response Best Practices**

Request and Response object 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.

****

**Topics**

* Request object
* Request object properties
* Request object methods
* Response object
* Response object properties
* Response object methods

### Request object

Express.js  is a  request & response objects parameters of the callback function and are used for the Express applications. The request object represents the HTTP request and contains properties for the request query string, parameters, body, HTTP headers, etc.

**Syntax :**  app.get(‘/’, function (req, res) { })

### Request Object Properties

These properties are represented below:

|  |  |  |
| --- | --- | --- |
| **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. |

### Request Object Methods

There are various types of request object method, these methods are represented below:

#### req.accepts (types)

It is used to the check content types are acceptable, based on the request accept HTTP header field.

Example :

req.accepts('html');

//=>?html?

req.accepts('text/html');

// => ?text/html?

#### req.get(field)

req.get is used to returns the specified HTTP request header field.

Example :

req.get('Content-Type');

// => "text/plain"

req.get('content-type');

// => "text/plain"

req.get('Something');

// => undefined

#### req.is(type)

If the incoming request is “CONTENT-TYPE”, this method returns true. HTTP header field matches the MIME type by the type parameter.

Example :

// With Content-Type: text/html; charset=utf-8

req.is('html');

req.is('text/html');

req.is('text/\*');

// => true

#### req.param(name [, defaultValue])

 req.param method is used to fetch the value of param name when present.

Example :

// ?name=sonia

req.param('name')

// => "sonia"

// POST name=sonia

req.param('name')

// => "sonia"

// /user/soniafor /user/:name

req.param('name')

// => "sonia"

### Response Object

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.

### Response Object Properties

|  |  |  |
| --- | --- | --- |
| **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. |

### Response Object Method

There are various types of response object method, these methods are represented below :

#### Response Append Method

**Syntax :**  res.append(field, [value])

Response append  method appends the specified value to the HTTP response header field. That means if the specified value is not appropriate so this method redress that.

Example :

res.append('Link', ['<http://localhost/>', '<http://localhost:3000/>']);

res.append('Warning', '299 Miscellaneous warning');

#### Response Attachment Method

**Syntax :**  res.attachment(‘path/to/js\_pic.png’);

Response attachment method allows you to send a file as an attachment in the HTTP response.

Example :

res.attachment('path/to/js\_pic.png'); .

#### Response Cookie Method

**Syntax:**  res.cookie(name, value [, options])

It is used to set a cookie name to value. The value can be a string or object converted to JSON.

Example :

res.cookie('name', 'alish', { domain: '.google.com', path: '/admin', secure: true });

res.cookie('Section', { Names: [sonica,riya,ronak] });

res.cookie('Cart', { items: [1,2,3] }, { maxAge: 900000 });

#### Response Download Method

**Syntax:**  res.download(path [, filename] [, fn])

Example:

res.download(‘/report-12345.pdf’);

res.download method is transfer file at path as an “attachment” and  the browser to prompt user for download.

#### Response End Method

**Syntax:**  res.end([data] [, encoding])

Response end  method is used to end the response process.

Example :

res.end();

res.status(404).end();

#### Response Get Method

**Syntax :**  res.get(field)

res.get method provides HTTP response header specified by field.

Example :

res.get('Content-Type');

#### Response JSON Method

**Syntax :**  res.json([body])

Response JSON method returns the response in JSON format.

Example :

res.json(null)

res.json({ name: 'alish' })

#### Response Render Method

**Syntax :**  res.render(view [, locals] [, callback])

Response render method renders a view and sends the rendered HTML string to the client.

Example :

// send the rendered view to the client

res.render('index');

// pass a local variable to the view

res.render('user', { name: 'monika' }, function(err, html) {

// ...

});

#### Response Status Method

**Syntax :**  res.status(code)

res.status method sets an HTTP status for the response.

Example :

res.status(403).end();

res.status(400).send('Bad Request');

#### Response Type Method

**Syntax :**  res.type(type)

res.type method sets the content-type HTTP header to the MIME type.

Example :

res.type('.html'); // => 'text/html'

res.type('html'); // => 'text/html'

res.type('json'); // => 'application/json'

res.type('application/json'); // => 'application/json'

res.type('png'); // => image/png:

**HTTP Response status messages**

|  |  |  |
| --- | --- | --- |
| **Code** | Meaning | **Description** |
| 200 | OK | The request is OK (this is the standard response for successful HTTP requests) |
| 201 | **Created** | **The request has been fulfilled, and a new resource is created** |
| 202 | Accepted | The request has been accepted for processing, but the processing has not been completed |
| 203 | Non-Authoritative Information | The request has been successfully processed, but is returning information that may be from another source |
| 204 | No Content | The request has been successfully processed, but is not returning any content |
| 205 | Reset Content | **The request has been successfully processed, but is not returning any content, and requires that the requester reset the document view** |
| 206 | Partial Content | The server is delivering only part of the resource due to a range header sent by the client |
| 400 | Bad Request | The request cannot be fulfilled due to bad syntax |
| 401 | Unauthorized | The request was a legal request, but the server is refusing to respond to it. |
| 403 | Forbidden | The request was a legal request, but the server is refusing to respond to it |
| 404 | Not Found | The requested page could not be found but may be available again in the future |
| 405 | Method Not Allowed | A request was made of a page using a request method not supported by that page |
| 408 | Request Timeout | Request Timeout |
| 500 | Internal Server Error | A generic error message, given when no more specific message is suitable |
| 502 | Bad Gateway | The server was acting as a gateway or proxy and received an invalid response from the upstream server |
| 503 | Service Unavailable | The server is currently unavailable (overloaded or down) |