HTTP Methods

The most common types of request methods are **GET and POST** but there are many others, including HEAD, PUT, DELETE, CONNECT, and OPTIONS. GET and POST are widely supported while support for other methods is sometimes limited but expanding.

**GET**

The GET method is used to retrieve information from the given server using a given URI. Requests using GET should only retrieve data and should have no other effect on the data.

**HEAD**

Same as GET, but transfers the status line and header section only.

**POST**

A POST request is used to send data to the server, for example, customer information, file upload, etc. using HTML forms.

**PUT**

Replaces all current representations of the target resource with the uploaded content.

**DELETE**

Removes all current representations of the target resource given by a URI.

**CONNECT**

Establishes a tunnel to the server identified by a given URI.

**OPTIONS**

Describes the communication options for the target resource.

**TRACE**

Performs a message loop-back test along the path to the target resource.

**How the Server looks at the URL**

You type a URL in your browser and press Enter. **Browser looks up IP address for the domain**. Browser initiates TCP connection with the server. Browser sends the HTTP request to the server.

Server URLs (or Uniform Resource Locators) are **the names that we typically think of when we think about a server** (www.ehow.com, for example). These URLs are actually translated into server IP addresses when we navigate to a web page, because each URL is assigned to an IP address.

**Request-Response Cycle -**

1. Express app receives a request when someone accesses a server.
2. For which it creates a request and response object.
3. The data is then used to generate and send back a meaningful response.
4. Now, to process that data, in Express we use MIDDLEWARES, which can manipulate the request/response object or execute any other code.

It is k/a middleware because it is executed in between i.e in the middle of receiving a request and sending back a response.

All the middlewares that we use in our app are known as Middleware Stack, and the order in which they are executed is decided by the order they are defined in the code. So a middleware that appears first in the code is executed first.

So, what happens is

1. The request and response object is created through Express.js
2. Both of them go through each middleware where they are processed or some other code is executed.
3. Then at the end of each middleware, a **next()** function is called. **next()** is a function, that we have access to in each middleware just like the request or response object.
4. And when the next() function is called, the next middleware in the middleware stack is executed with the exact req and res object.
5. And like this, we complete the entire Request-Response Cycle.
6. *Middleware functions can perform the following tasks:*
7. *- Execute any code.*  
   *- Make changes to the request and the response objects.*  
   *- End the request-response cycle.*  
   *- Call the next middleware function in the stack.*
8. Mobile Responsive Testing and Debugging on Real Devices
9. Mobile Responsive Test across range of Mobile Devices and platforms is extremely important for superior User Experience. However, Mobile Responsive Test and debugging can get knotty, especially when there are issues like obfuscated HTML/CSS/javascript code, improperly implemented HTTPS, browser inconsistencies, page load performance and so on. So how do you overcome these issues? Using a Web Developer Tools or debugging extensions make debugging your web application easier. Here we’ll talk about the most popular one — [Google Chrome Developer Tools](https://www.pcloudy.com/mobile-web-debugging-on-real-devices-using-chrome-developer-tools/) Google Chrome Developer Tools.
10. The Google Chrome Developer Tools is a consolidated environment built into Google Chrome Browser. It is used for web authoring, debugging, monitoring, optimizing, and understanding web applications or websites. (Sourced from [Google Chrome DevTools](https://developers.google.com/web/tools/chrome-devtools/?hl=en))

JavaScript is a simple programming language that runs in any browser JavaScript Engine. Whereas Node JS is an interpreter or running environment for a JavaScript programming language that holds many excesses, it requires libraries that can easily be accessed from JavaScript programming for better use.

**JavaScript types**

* Boolean type.
* Null type.
* Undefined type.
* Number type.
* BigInt type.
* String type.
* Symbol type.