# Node.js HTTP Module Cheat Sheet

|  |  |  |  |
| --- | --- | --- | --- |
| Function | **Syntax** | Example | Explanation |
| http.createServer | **http.createServer([options][, requestListener])** | const http = require('http'); const server = http.createServer((req, res) => {  res.statusCode = 200;  res.setHeader('Content-Type', 'text/plain');  res.end('Hello, World!\n'); }); server.listen(3000, '127.0.0.1', () => {  console.log('Server running at http://127.0.0.1:3000/'); }); | Creates an HTTP server that listens for requests and defines a response handler. |
| http.request | **http.request(options[, callback])** | const http = require('http'); const options = {  hostname: 'example.com',  port: 80,  path: '/path',  method: 'GET' }; const req = http.request(options, (res) => {  let data = '';  res.on('data', (chunk) => {  data += chunk;  });  res.on('end', () => {  console.log(data);  }); }); req.on('error', (error) => {  console.error(error); }); req.end(); | Makes an HTTP request to a specified server and handles the response. |
| http.get | **http.get(options[, callback])** | const http = require('http'); http.get('http://example.com', (res) => {  let data = '';  res.on('data', (chunk) => {  data += chunk;  });  res.on('end', () => {  console.log(data);  }); }).on('error', (error) => {  console.error(error); }); | A convenience method for making HTTP GET requests. |
| http.Server.listen | **server.listen(port[, hostname][, backlog][, callback])** | server.listen(3000, '127.0.0.1', () => {  console.log('Server running at http://127.0.0.1:3000/'); }); | Starts the HTTP server and begins listening for incoming connections on a specified port and hostname. |
| http.Server.close | **server.close([callback])** | server.close(() => {  console.log('Server closed'); }); | Stops the server from accepting new connections and executes the callback once all active connections are closed. |
| http.Server.setTimeout | **server.setTimeout(msecs[, callback])** | server.setTimeout(5000, () => {  console.log('Request timeout'); }); | Sets the timeout value for sockets created by the server. |
| http.Server.on | **server.on(event, listener)** | server.on('request', (req, res) => {  console.log('New request received'); }); | Attaches an event listener to the server for specific events like 'request', 'connection', or 'error'. |
| http.IncomingMessage | **http.IncomingMessage** | server.on('request', (req, res) => {  console.log(req.method);  console.log(req.url); }); | Represents the incoming request to the server, including properties like method, URL, and headers. |
| http.ServerResponse | **http.ServerResponse** | server.on('request', (req, res) => {  res.statusCode = 200;  res.setHeader('Content-Type', 'text/plain');  res.end('Response sent'); }); | Represents the outgoing response sent from the server, including methods for setting headers and status codes. |
| http.Agent | **http.Agent** | const http = require('http'); const agent = new http.Agent({ keepAlive: true }); const options = { agent }; http.get('http://example.com', options, (res) => {  console.log('Response received'); }); | Manages connection persistence and reuse for HTTP clients. |