

HTTP with Node.js

MANAGING HTTP REQUESTS



Armen Avanesi
SOFTWARE DEVELOPER



The HTTP Module



Node comes with a built-in HTTP module

Provides a flexible, robust API

Allows us to:

- Create a server
- Work with streamed data
- Make API calls

No external dependencies



Additional Concepts

Readable and writable streams

EventEmitter interface

Underlying classes



Concepts to Know

Working knowledge of JavaScript

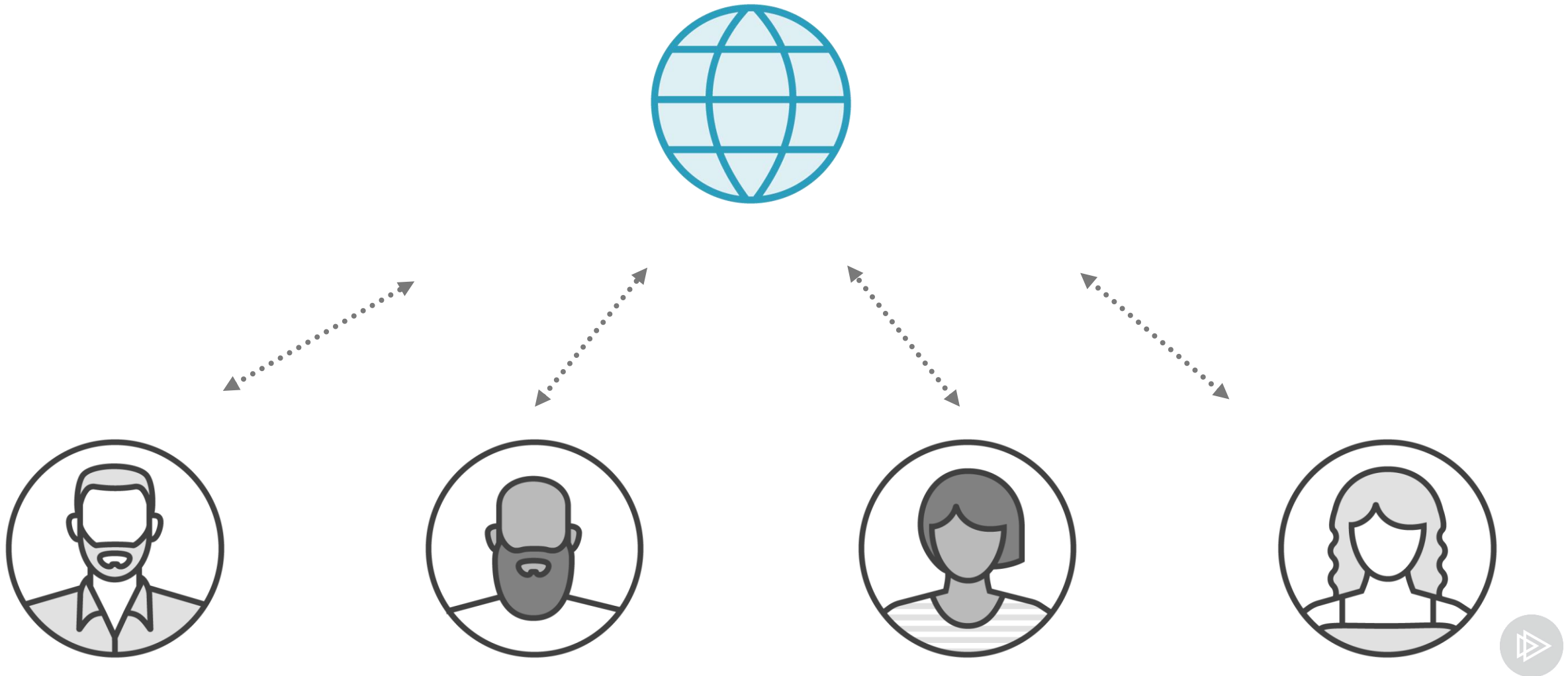
Requiring modules

Callbacks

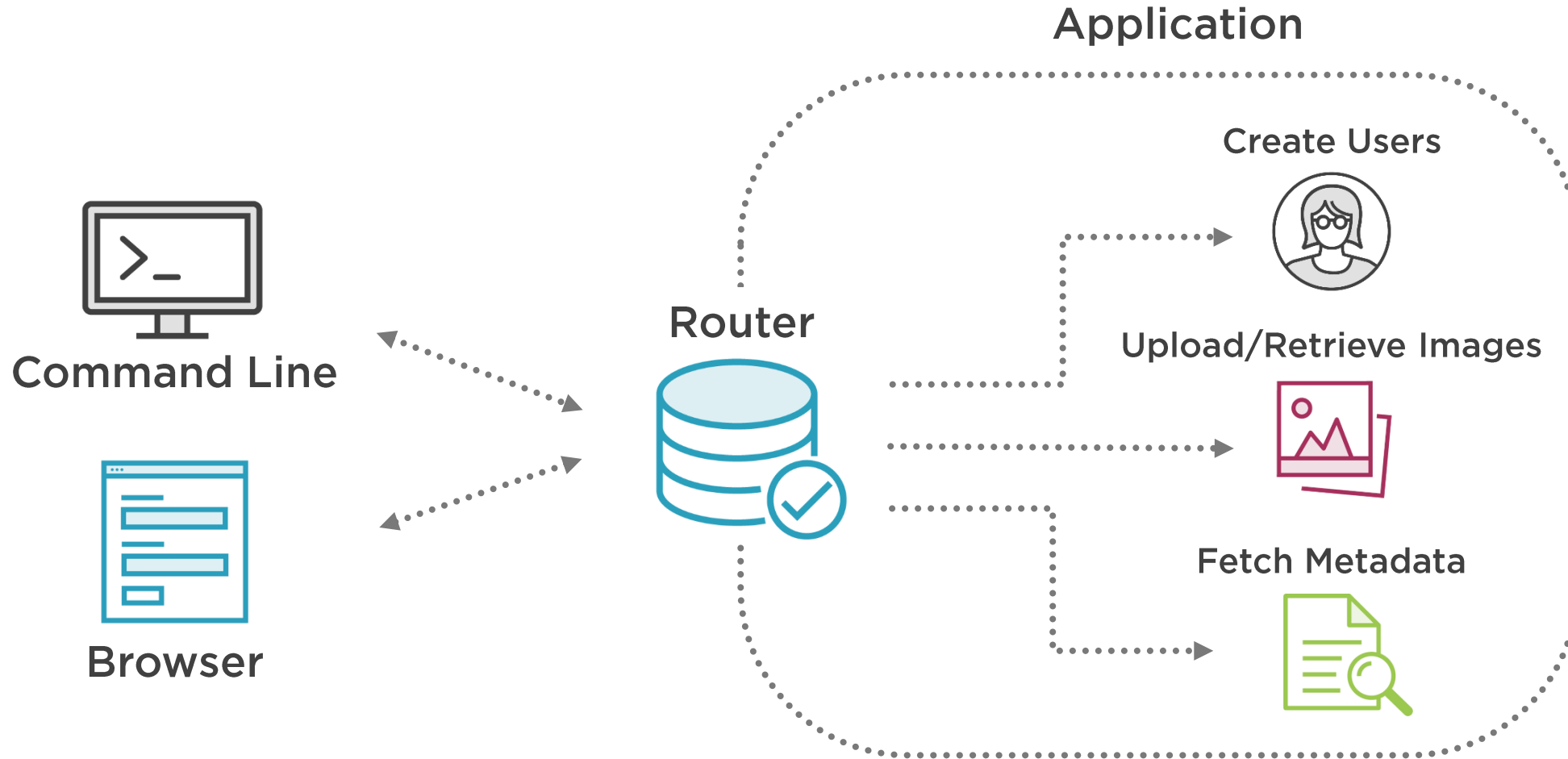
Web server concepts



Project Application



Application Architecture



GitHub Slide



Package Installation

// installing with yarn:

yarn install ←

// installing with npm

npm install ←



Node version 12.13.1



Example Mock Service



```
function saveUserToDatabase() {  
  console.log("saving..");  
  console.log("done");  
};
```



Let's get started

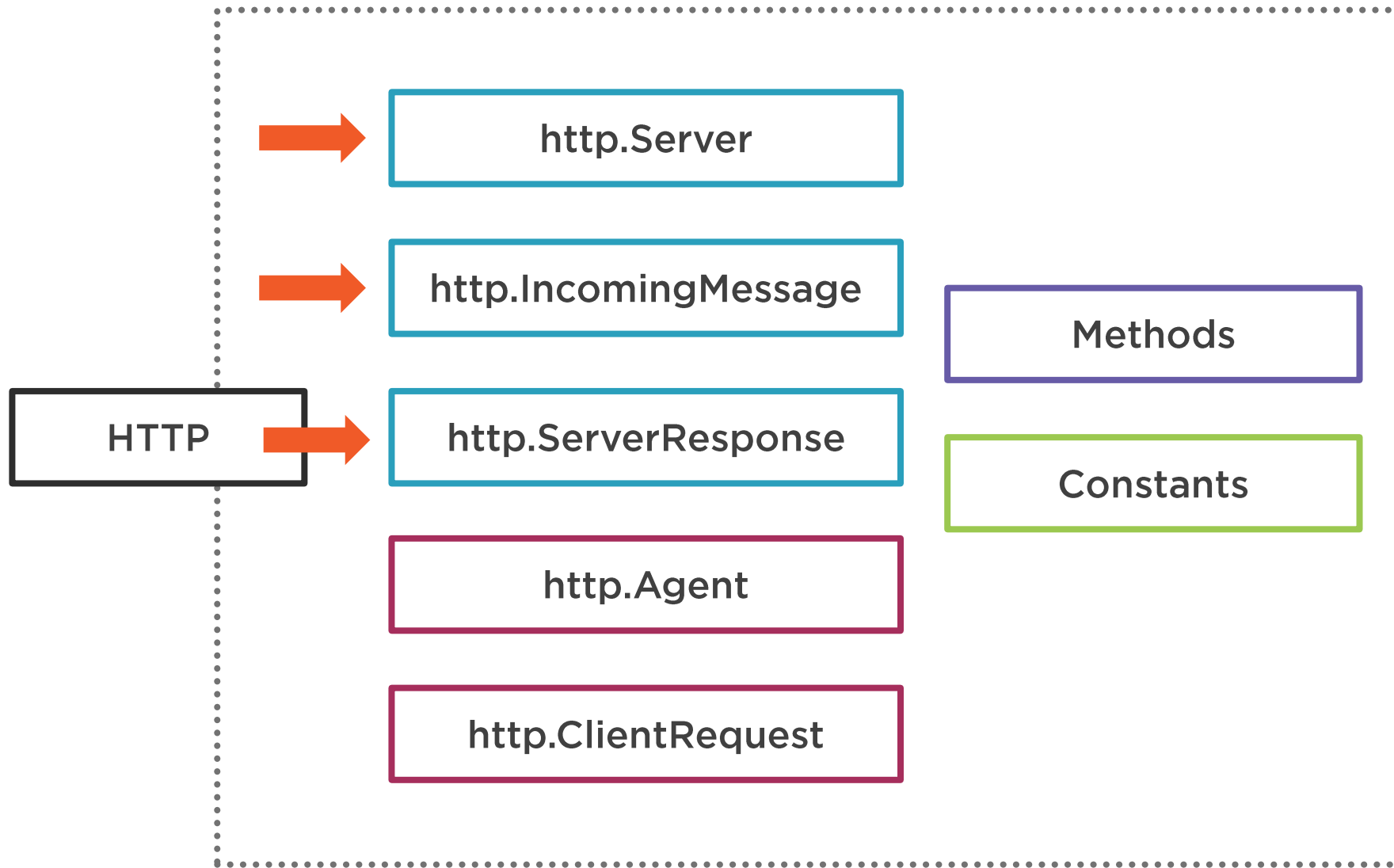


Creating Your First Server



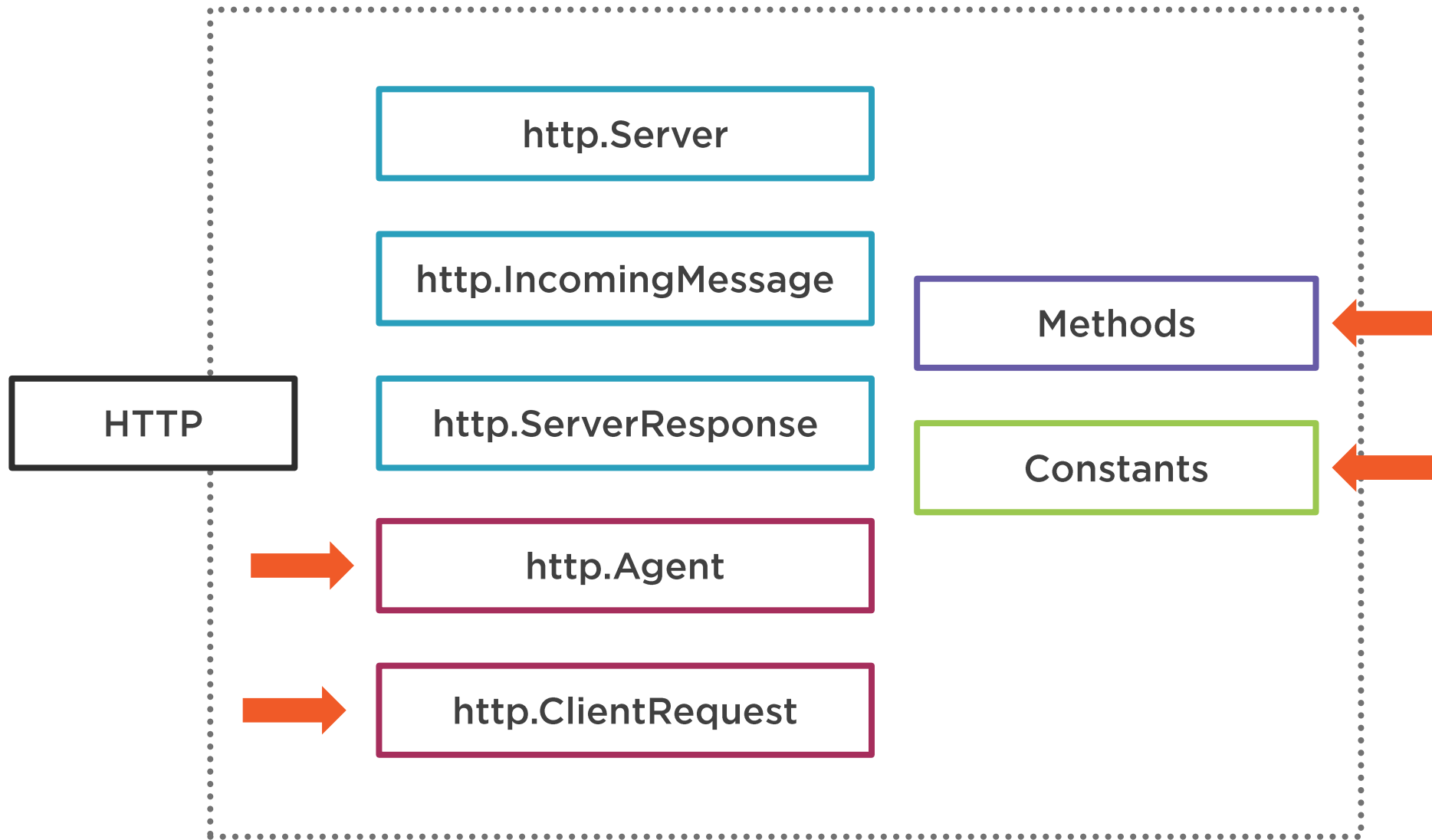
The Anatomy of the HTTP Module





server.js

```
const server = http.createServer();  
server.on('request', (request, response) => {  
       
});
```



server.js



```
console.log(http.STATUS_CODES[404]);
```

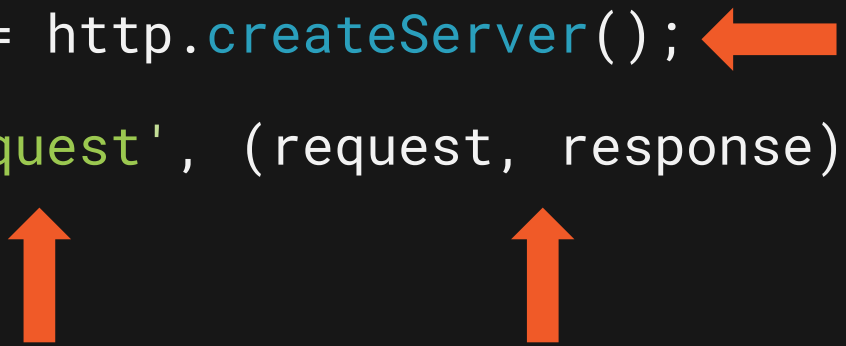
```
"Not Found"
```

What are streams and event emitters?



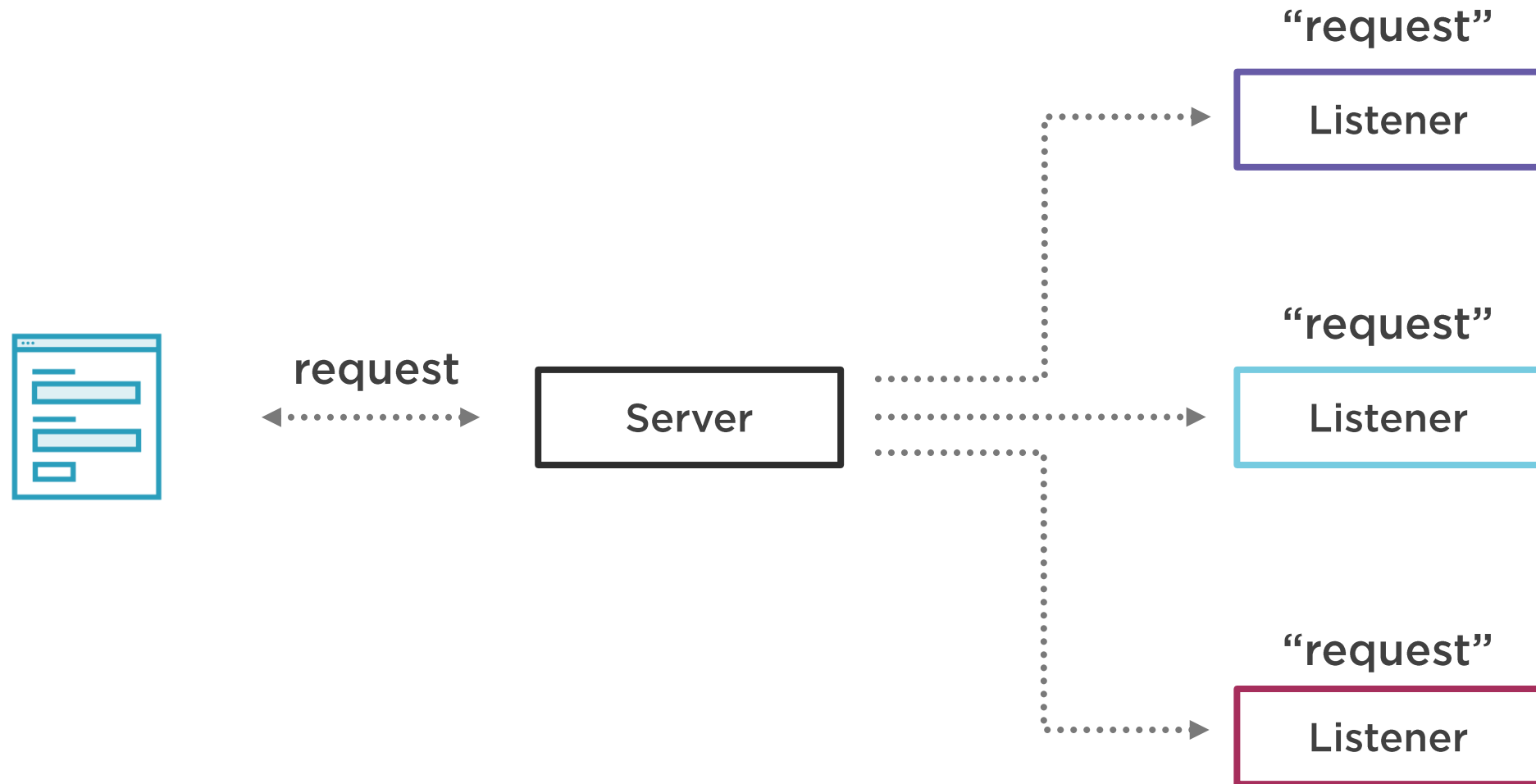
server.js

```
const server = http.createServer();  
server.on('request', (request, response) => {  
});
```



The diagram consists of three orange arrows. One arrow points from the right towards the `createServer()` function call in the first line of code. Two other arrows point upwards from below the `'request'` string and the `(request, response)` parameter list in the second line of code.

Event Emitters



Node.js

About these Docs

Usage & Example

Assertion Testing

Async Hooks

Buffer

C++ Addons

C/C++ Addons with N-API

Child Processes

Cluster

Command Line Options

Console

Crypto

Debugger

Deprecated APIs

DNS

Domain

HTTP | Node.js v13.5.0 Docum

nodejs.org/api/http.html#http_event_abort

Event: 'abort'

Added in: v1.4.1

Emitted when the request has been aborted by the client. This event is only emitted on the first call to `abort()`.

Event: 'connect'

Added in: v0.7.0

- `response` `<http.IncomingMessage>`
- `socket` `<stream.Duplex>`
- `head` `<Buffer>`

Emitted each time a server responds to a request with a `CONNECT` method. If this event is not being listened for, clients receiving a `CONNECT` method will have their connections closed.

This event is guaranteed to be passed an instance of the `<net.Socket>` class, a subclass of `<stream.Duplex>`, unless the user specifies a socket type other than `<net.Socket>`.

A client and server pair demonstrating how to listen for the `'connect'` event:

```
const http = require('http');
const net = require('net');
const { URL } = require('url');

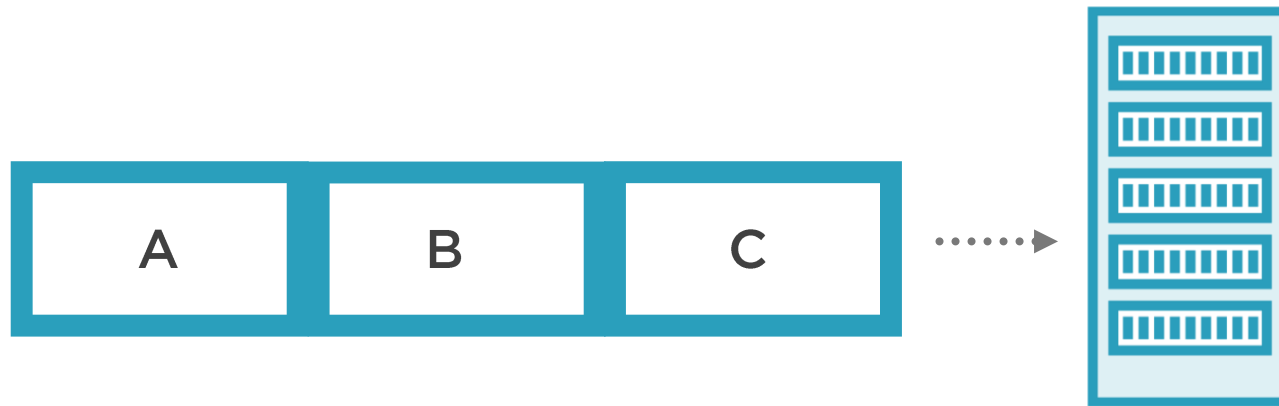
// Create an HTTP tunneling proxy
const proxy = http.createServer((req, res) => {
```

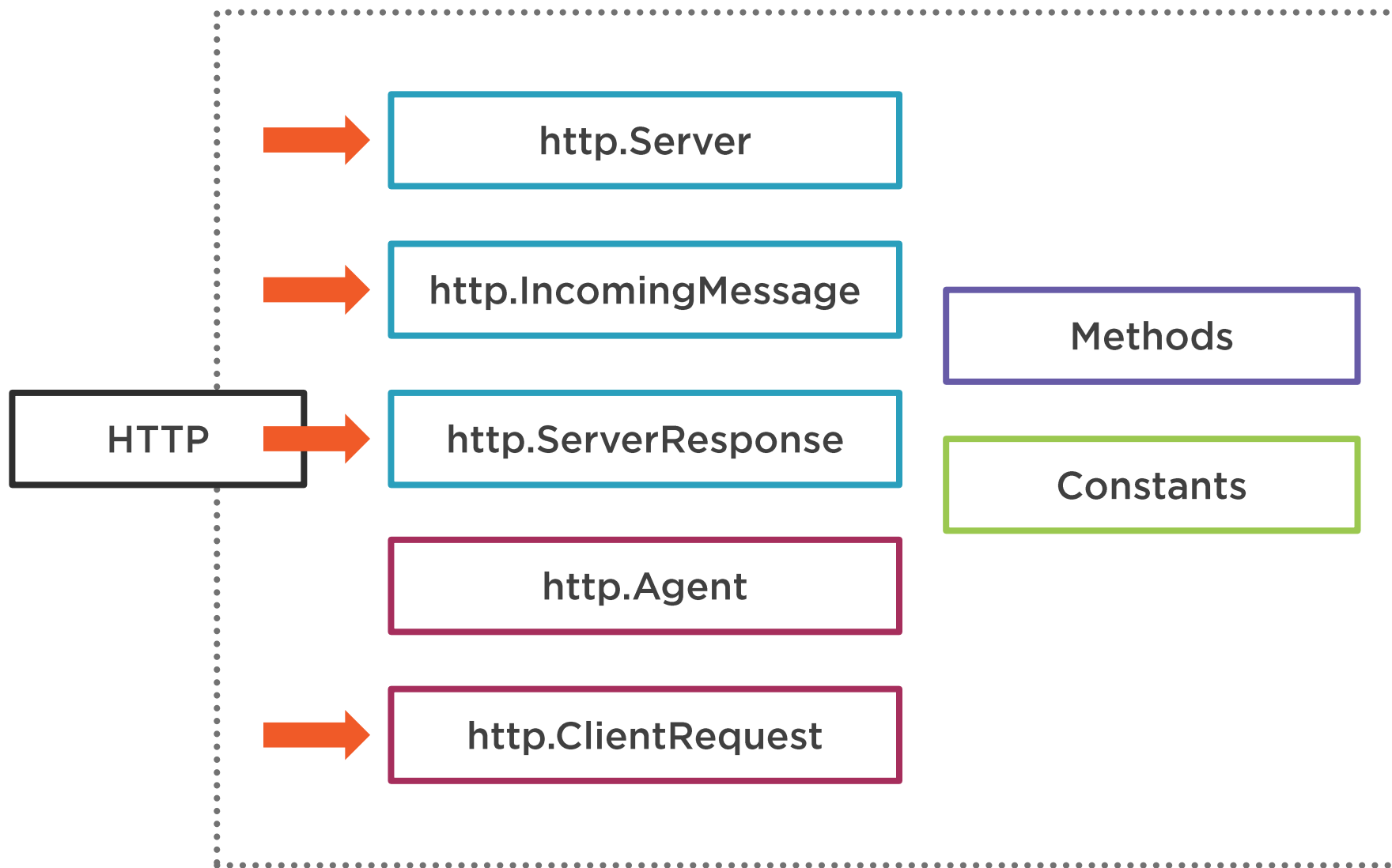
Streams

Buffer

Buffer

Buffer





Working with the HTTP Method, Headers, and URL





Router



Create Users



Upload/Retrieve Images



Fetch Metadata



Working with the Request Body



Request Object Properties

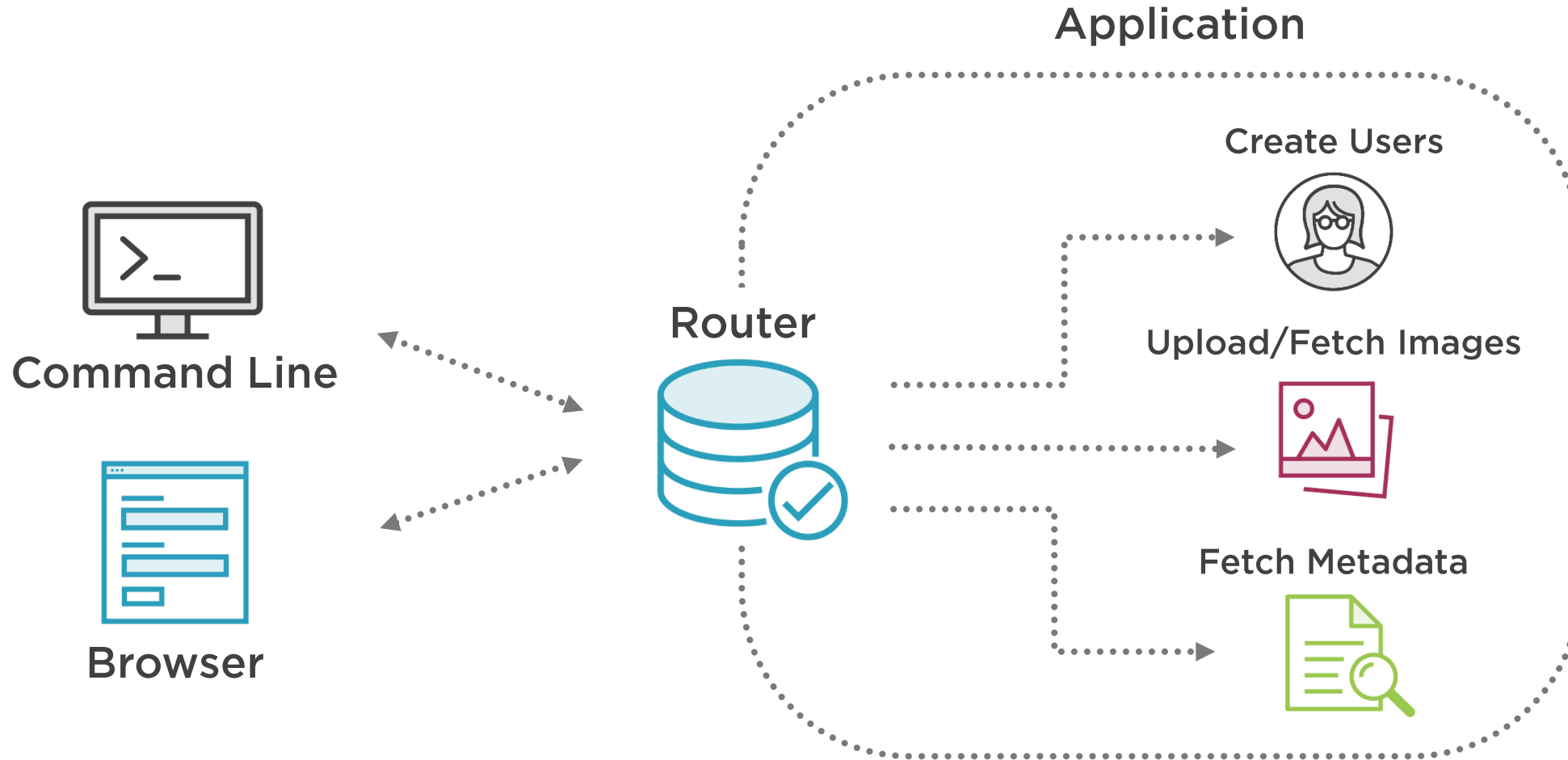
`request.url`

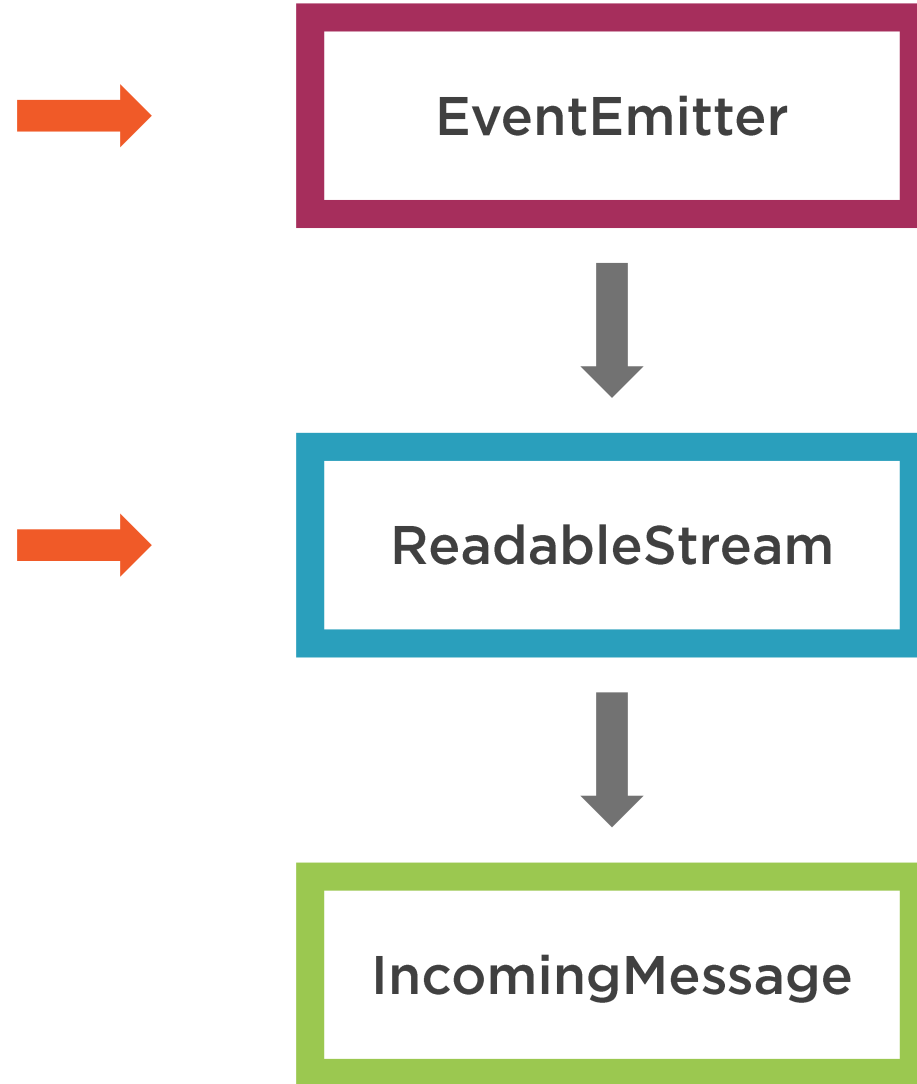
`request.method`

`request.headers`



Application Architecture





Recap

Our request headers, URL, and method are available to us immediately

We need to fully receive a request body

Buffers are like small packages of information

The “request” object makes use of two interfaces

- ReadableStream
- EventEmitter



Challenge

Modify the payload to include:

phoneNumber

firstName



The “body” Package



Benefits

Recommended in
Node.js
Documentation

Easy to Install

Simple to Use



How About HTTPS?



Summary



Creating a server

HTTP module components

Routing

Parsing request bodies

The "body" package

How to use the HTTPS module

