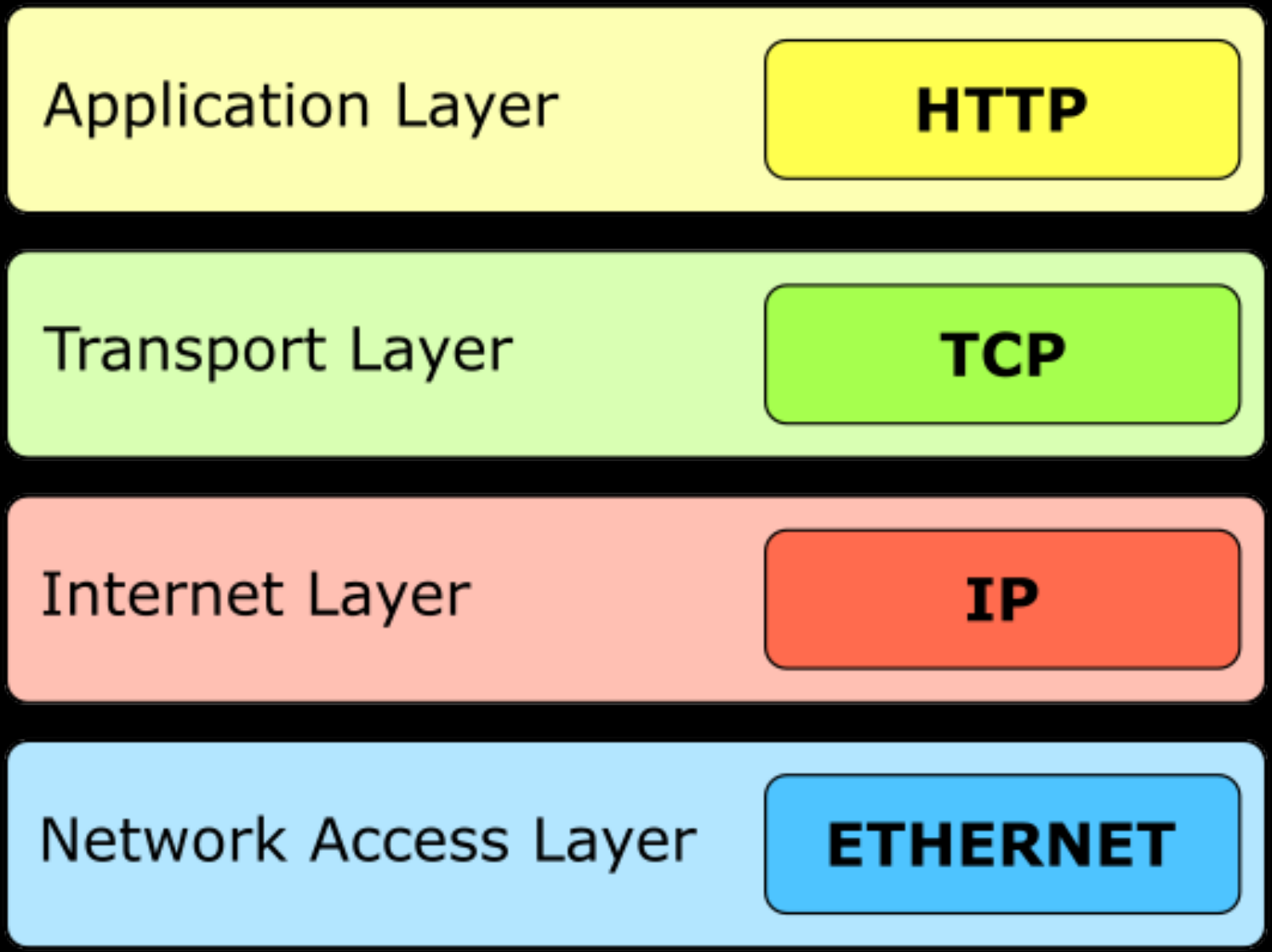

Application Layer
HTTP
Transport Layer
TCP
Internet Layer
IP
Network Access Layer
ETHERNET

HTTP

HTTP

Funciona como un **protocolo** de solicitud-respuesta (request-response) en el **modelo** cliente-servidor (client-server).

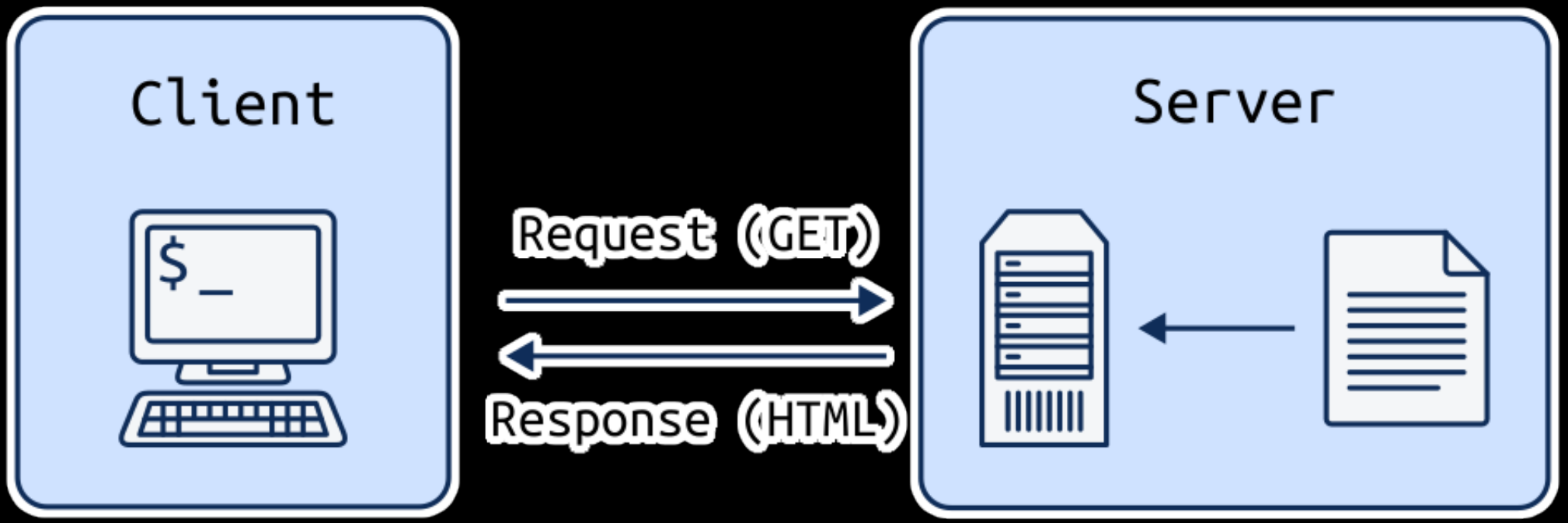
HTTP

Protocolo: Conjunto de reglas que describen cómo se debe ejecutar una actividad.

HTTP

Modelo Cliente-Servidor: Estructura de aplicaciónes distribuidas que divide las tareas entre los proveedores de un recurso o servicio (server) y los solicitantes de servicio (client).

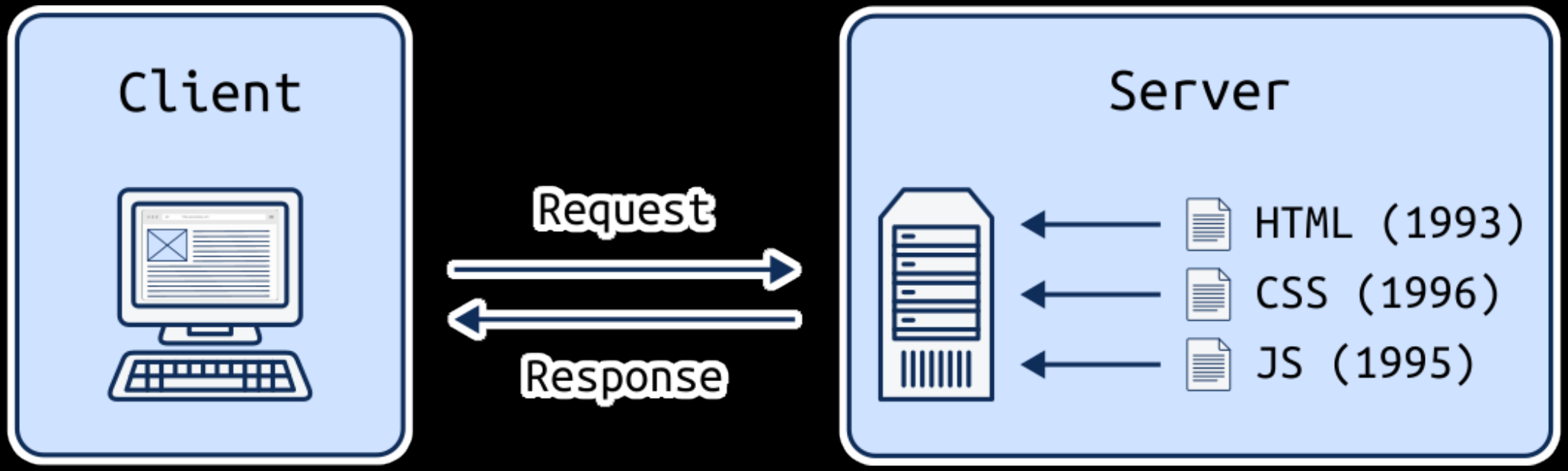
HTTP/0.9 (1991)



```
$ links commonmark.org
```

```
<!doctype html>
<html>
  <head>
    <title>Título de la página.</title>
  </head>
  <body>
    <h1>Título principal del documento</h1>
    <p>Lorem ipsum dolor sit amet, consectetur
adipiscing elit. Nullam placerat leo vel.</p>
    <p>
      <a href="https://www.wikipedia.org/">
        Wikipedia!
      </a>
    </p>
  </body>
</html>
```

HTTP/1.1 (1997)



```
<!doctype html>
<html>
  <head>
    <title>Título de la página.</title>
    <link rel="stylesheet" href="style_file.css">
    <style>h1 {color: red;}</style>
  </head>
  <body>
    <h1 class="big">Título principal del
documento</h1>
    <p style="color:blue;">Lorem ipsum dolor sit
amet, consectetur adipiscing elit.</p>
  </body>
  <script src="scripts_file.js"></script>
  <script>console.log("Hello, World!");</script>
</html>
```

Request Methods

(También llamados "verbos")

- GET
- HEAD
- POST
- PUT
- PATCH
- DELETE
- OPTIONS

URL

La URL (Uniform Resource Locator) es un tipo de URI (Uniform Resource Identifier).

```
URI = scheme ":"
      [authority]
      path
      ["?" query]
      ["#" fragment]
```

```
authority = "//"
            [userinfo "@"]
            Host
            [":" port]
```

Headers

Request Headers:

```
Host: en.wikipedia.org
Accept: text/html
Cookie: key1=value1;key2=value2
```

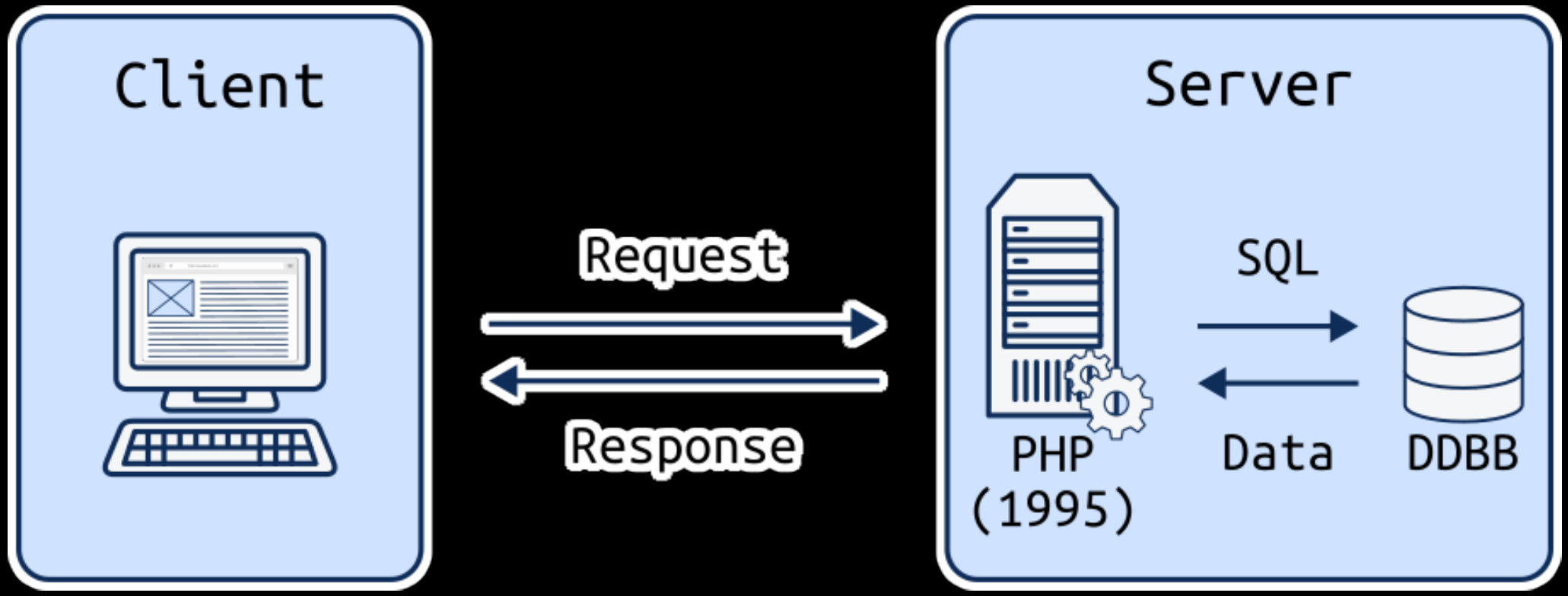
Response Headers:

```
Allow: GET, HEAD
Cache-Control: max-age=3600
Content-Encoding: gzip
Content-Length: 348
Date: Tue, 15 Nov 1994 08:12:31 GMT
Location: /path/to/resource.html
```

Status Codes

- 1XX (informational)
- 2XX (successful)
- 3XX (redirection)
- 4XX (client error)
- 5XX (server error)

Dynamic Content



REST

(**RE**presentational **S**tate **T**ransfer)

- Estilo de arquitectura de software para la World Wide Web.
- Promueve aplicaciones web sin estado y confiables.
- Las aplicaciones que siguen los lineamientos REST son descritas como RESTful.

REST

Estado Representacional:

Se refiere al estado o condición actual de un recurso.

Cada recurso (documento, imagen, etc.) tiene un estado que puede ser representado y manipulado a través de interfaces estándar.

REST

1. Arquitectura basada en recursos: Los recursos (como documentos o datos) son identificados por URLs y manipulados a través de métodos HTTP estándar (GET, POST, PUT, DELETE).

REST

2. Operaciones CRUD: REST facilita las operaciones Create, Read, Update, Delete (CRUD) sobre recursos usando los métodos HTTP adecuados.

REST

3. Sin estado (stateless): Cada solicitud HTTP contiene toda la información necesaria para procesarla, lo que simplifica la escalabilidad y la fiabilidad del sistema.

REST

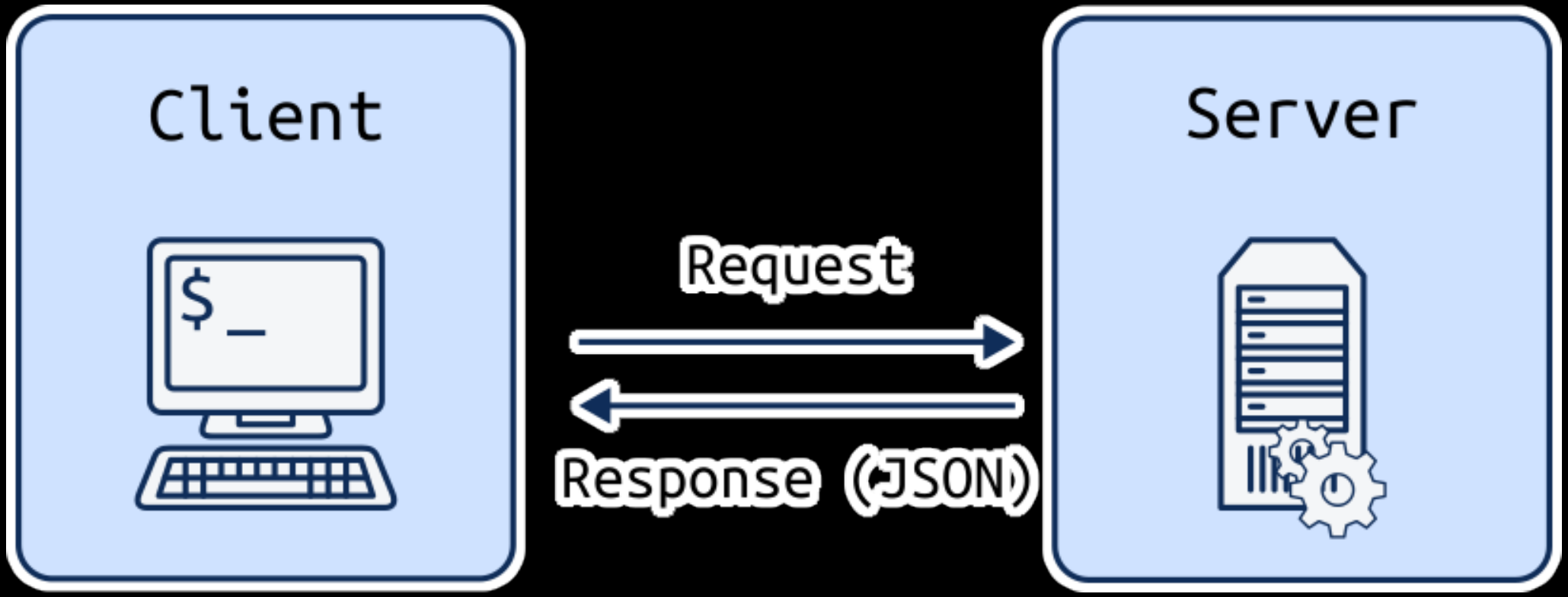
4. Interfaz uniforme:

- Cada recurso se identifica de manera única mediante una URI.
- Los recursos pueden manipularse a través de diferentes representaciones (JSON, XML, HTML, etc).
- Se proporcionan enlaces (hipermedia) que los clientes pueden seguir para descubrir y acceder a otros recursos relacionados.

REST

5. Independencia de la plataforma: Permite la interoperabilidad entre sistemas distribuidos y diversos clientes al utilizar protocolos y formatos estándar como HTTP, JSON o XML.

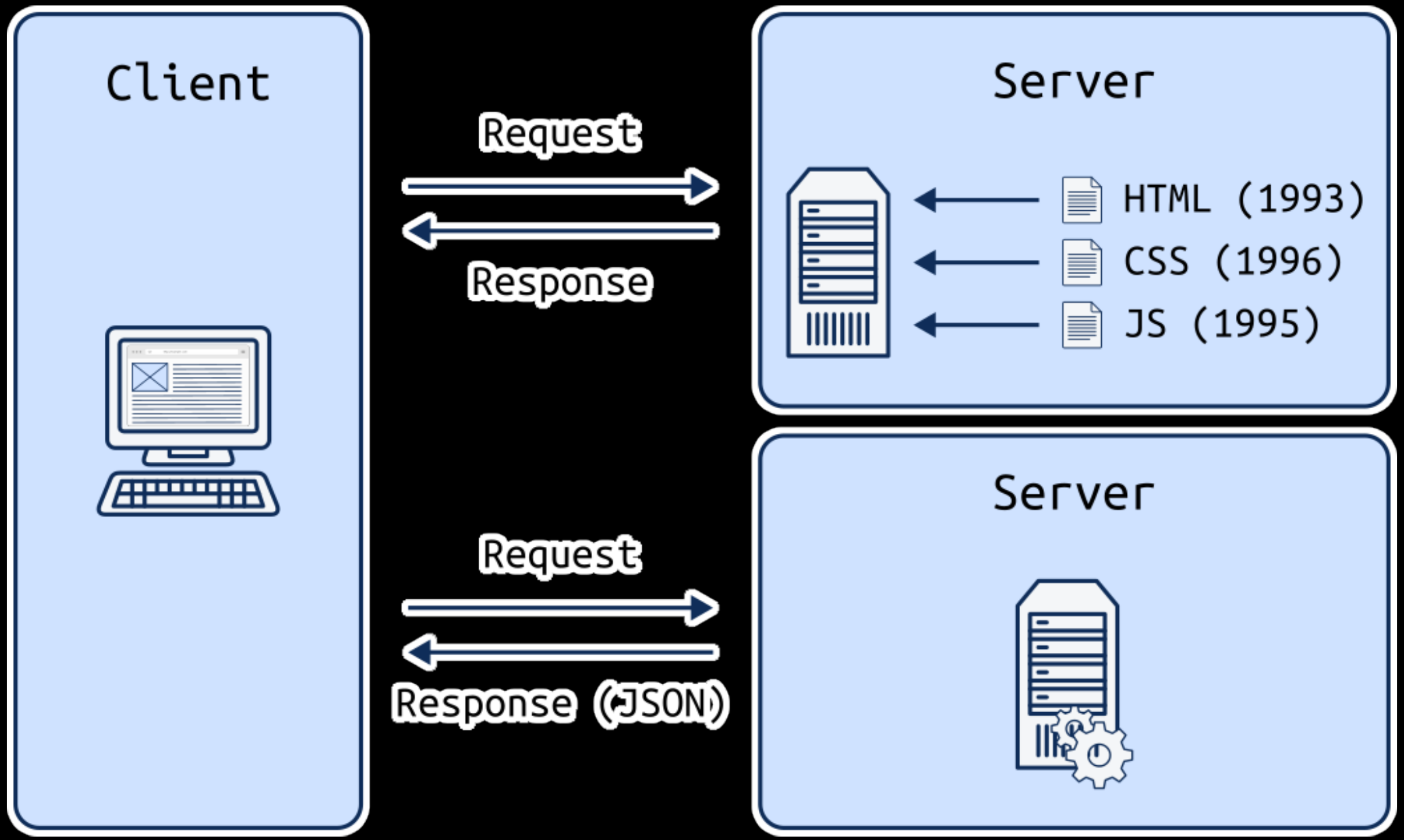
RESTful API



https://httpbin.org/

A simple HTTP Request & Response Service.

Web Application



HTML

CSS

JavaScript

JSON

AJAX

PHP

SQL

REST API SERVER

```
POST   /candidates {"name":string} -> {"id":number,"name":string}
DELETE /candidates/id
GET    /candidates -> [{"id":number,"name":string}]
DELETE /candidates
POST   /vote {"id":number}
GET    /report -> [{"id":number,"name":string,"votes":number}]
```

https://repo.or.cz/asis23-votoe-server.git

REST API CLIENT

https://repo.or.cz/asis23-votoe-client.git