

**HENRY**

A bright yellow beam of light originates from the left edge of the frame and points towards the letter 'R' in the word 'HENRY'. The beam is wider on the left and tapers as it moves to the right. The word 'HENRY' is written in a bold, black, sans-serif font.



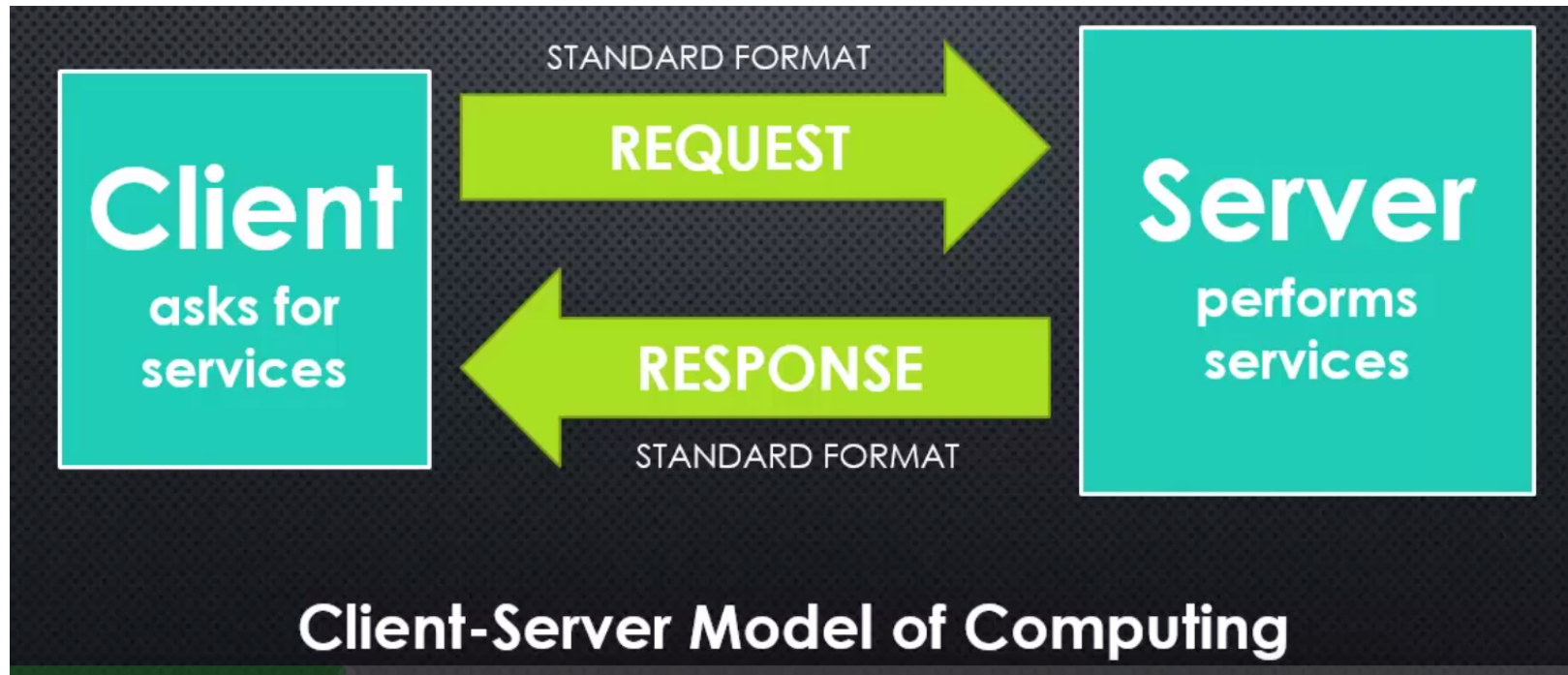
# Web Server



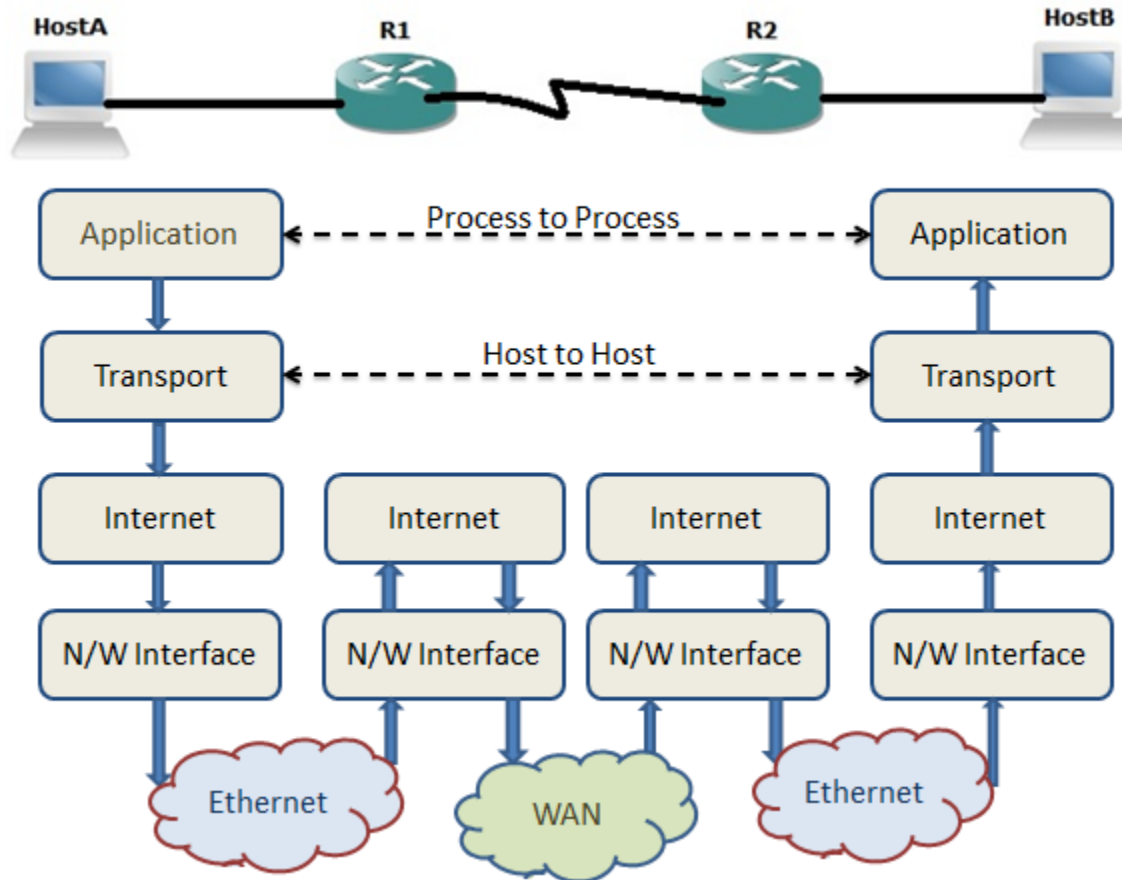
# Web Server

*“ Un servidor web es cualquier computadora o sistema que procese solicitudes (requests) y que devuelva una respuesta (response) a través de un protocolo de red.*

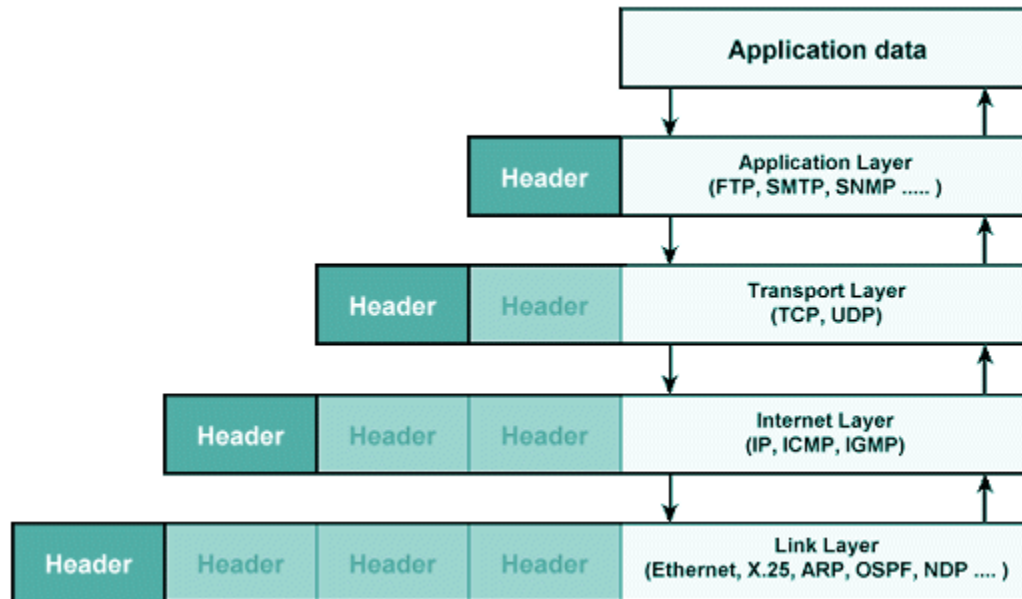
# Modelo Cliente-Servidor



# Networks



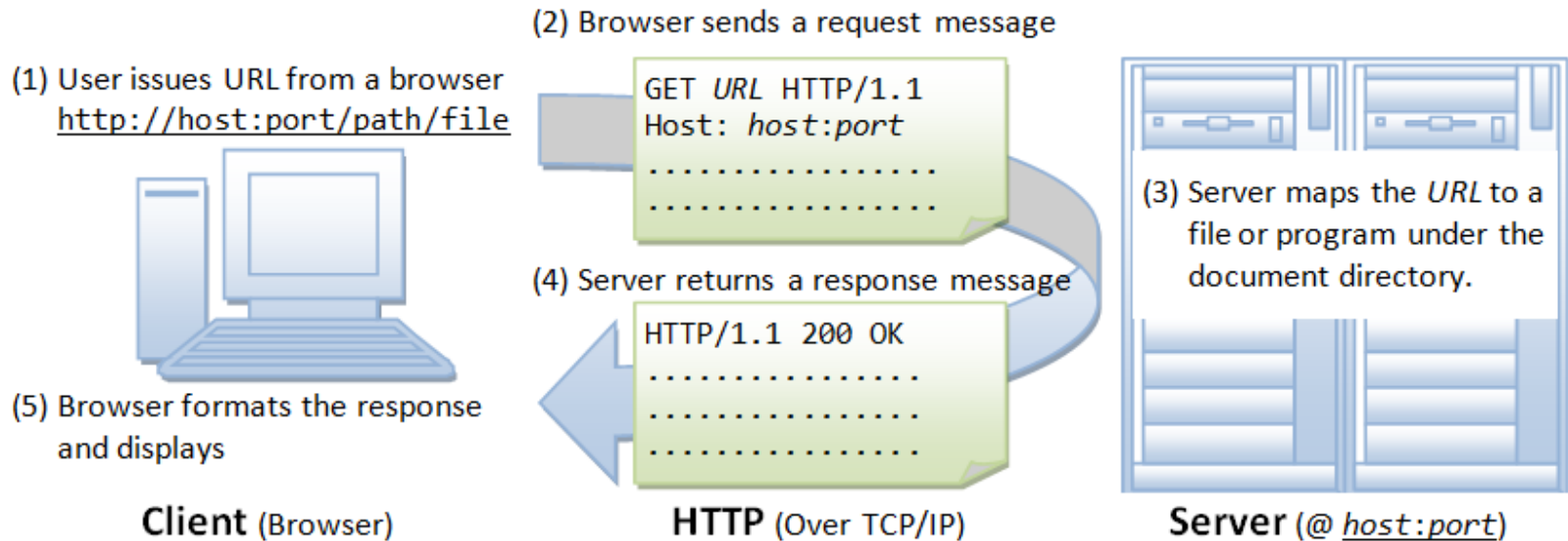
# Encapsulación



# Protocols

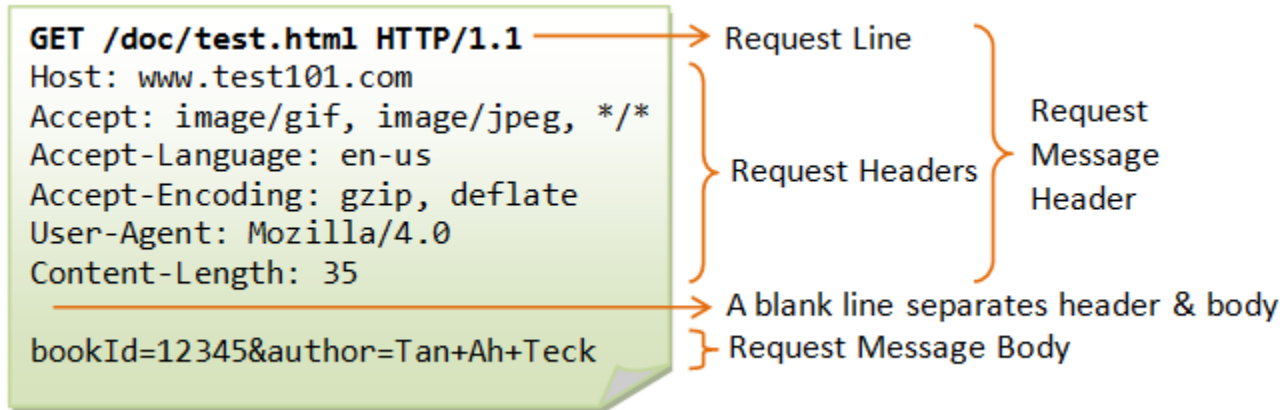
TCP/IP model	Protocols and services	OSI model
Application	HTTP, FTP, Telnet, NTP, DHCP, PING	Application
		Presentation
		Session
Transport	TCP, UDP	Transport
Network	IP, ARP, ICMP, IGMP	Network
Network Interface	Ethernet	Data Link
		Physical

# HTTP





# HTTP



A *socket* is one endpoint of a two-way communication link between two programs running on the network.

A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent to.

# Server Básico

```
1
2 var http = require('http'); // importamos el módulo http para poder trabajar con el protocolo
3
4
5 // Creamos una serie de events listener,
6 // que van a escuchar por requests que ocurren en este socket
7
8 http.createServer( function(req, res){
9
10     //Para crear un response empezamos escribiendo el header
11     res.writeHead(200, { 'Content-Type':'text/plain' })
12     //Le ponemos el status code y algunos pair-values en el header
13     res.end('Hola, Mundo!\n');
14
15 }).listen(1337, '127.0.0.1');
16 // Por último tenemos que especificar en que puerto y
17 // en qué dirección va a estar escuchando nuestro servidor
```

< DEMO />

# Enviando HTML

```
1 var http = require('http');
2 var fs   = require('fs');
3
4 //Importamos el módulo fs que nos permite leer y escribir archivos del file system
5
6 http.createServer( function(req, res){
7
8     res.writeHead(200, { 'Content-Type':'text/html' })
9     var html = fs.readFileSync(__dirname + '/html/index.html');
10    res.end(html);
11
12
13 }).listen(1337, '127.0.0.1');
```

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <title>Prueba!</title>
5 </head>
6 <body>
7     <h1>Hola, Mundo!</h1>
8     <p>Bienvenidos!</p>
9 </body>
10 </html>
```

< DEMO />

# Enviando HTML (Templates)

```
1 var http = require('http');
2 var fs   = require('fs');
3 //Importamos el módulo fs que nos permite leer y escribir archivos del file system
4
5 http.createServer( function(req, res){
6
7     res.writeHead(200, { 'Content-Type':'text/html' })
8     var html = fs.readFileSync(__dirname + '/html/template.html', 'utf8');
9     //Codificamos el buffer para que sea una String
10    var nombre = 'Soy Henry';
11    //Esta es la variable con la que vamos a reemplazar el template
12    html = html.replace('{nombre}', nombre);
13    // Usamos el método replace es del objeto
14    res.end(html);
15
16
17 }).listen(1337, '127.0.0.1');
```

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <title>Prueba!</title>
5 </head>
6 <body>
7     <h1>Hola, {nombre}!</h1>
8     <p>Bienvenidos!</p>
9 </body>
10 </html>
```

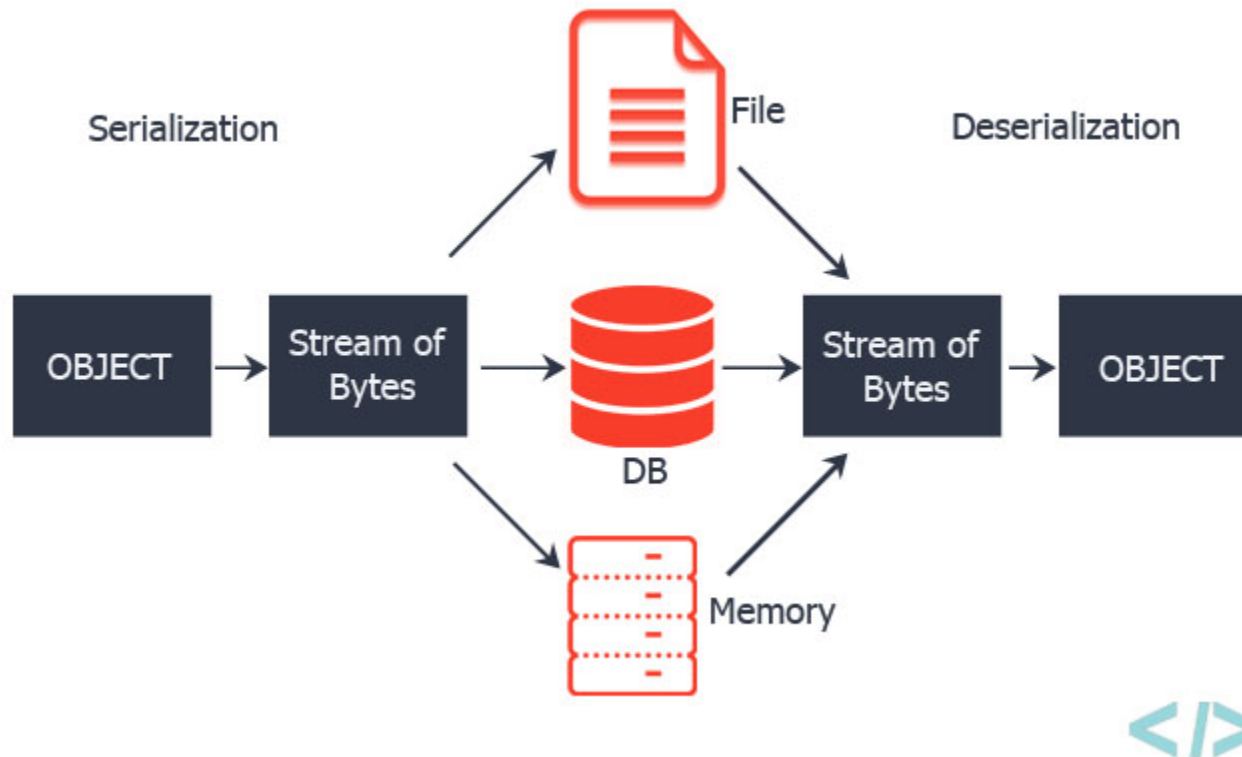
< DEMO />

# JSON

```
1 var http = require('http');
2 var fs   = require('fs');
3
4 http.createServer( function(req, res){
5
6     res.writeHead(200, { 'Content-Type':'application/json' })
7     //Vamos a devolver texto en formato JSON
8     var obj = {
9         nombre: 'Juan',
10        apellido: 'Perez'
11    }; //Creamos un objeto de ejemplo para enviar como response
12
13    res.end( JSON.stringify(obj) );
14    //Antes de enviar el objeto, debemos parsearlo y transformarlo a un string JSON
15
16 }).listen(1337, '127.0.0.1');
```

< DEMO />

# Serialize / DeSerialize



# Rutas

```
1 var http = require('http');
2 var fs   = require('fs');
3
4 http.createServer( function(req, res){
5     if( req.url === '/' ){
6         res.writeHead(200, { 'Content-Type': 'text/html' })
7         var html = fs.readFileSync(__dirname + '/html/index.html');
8         res.end(html);
9     } else if( req.url === '/api' ){
10        res.writeHead(200, { 'Content-Type': 'application/json' })
11        var obj = {
12            nombre: 'Juan',
13            apellido: 'Perez'
14        };
15        res.end( JSON.stringify(obj) );
16    } else{
17        res.writeHead(404); //Ponemos el status del response a 404: Not Found
18        res.end(); //No devolvemos nada más que el estado.
19    }
20
21 }).listen(1337, '127.0.0.1');
```

< DEMO />



# RESTful API





# RESTful API

*“ Rest es una arquitectura o forma de diseñar el backend de una aplicación que viva en internet. REST viene de "REpresentational State Transfer" y está basado fuertemente en cómo trabaja HTTP, que es el protocolo que usamos comunmente en la web.*

# RESTful API

Task	Method	Path
Create a new task	POST	/tasks
Delete an existing task	DELETE	/tasks/{id}
Get a specific task	GET	/tasks/{id}
Search for tasks	GET	/tasks
Update an existing task	PUT	/tasks/{id}



# RESTful API

- Cliente - Servidor
- Stateless
- Cacheable
- Sistema de capas