




Node.js

Ingeniería Web
Semestre 2021-2



¿Qué es?

- Es un entorno de programación de Javascript.
 - Permite ejecutar Javascript del lado del servidor (Backend).
 - Permite construir aplicaciones en tiempo real.
 - Plataforma basado en el motor V8 de Google Chrome.
 - Permite desarrollar aplicaciones de red escalables
 - Modelo de eventos y asincrono
 - Modelo no bloqueante
- 

Entorno de programación

JavaScript

Motor V8

Mecanismos para interactuar con el exterior

Librerías estándar

Utilidades – NPM

Framework, ejemplo Express – middleware y rutas

Modelo asíncrono y no bloqueante

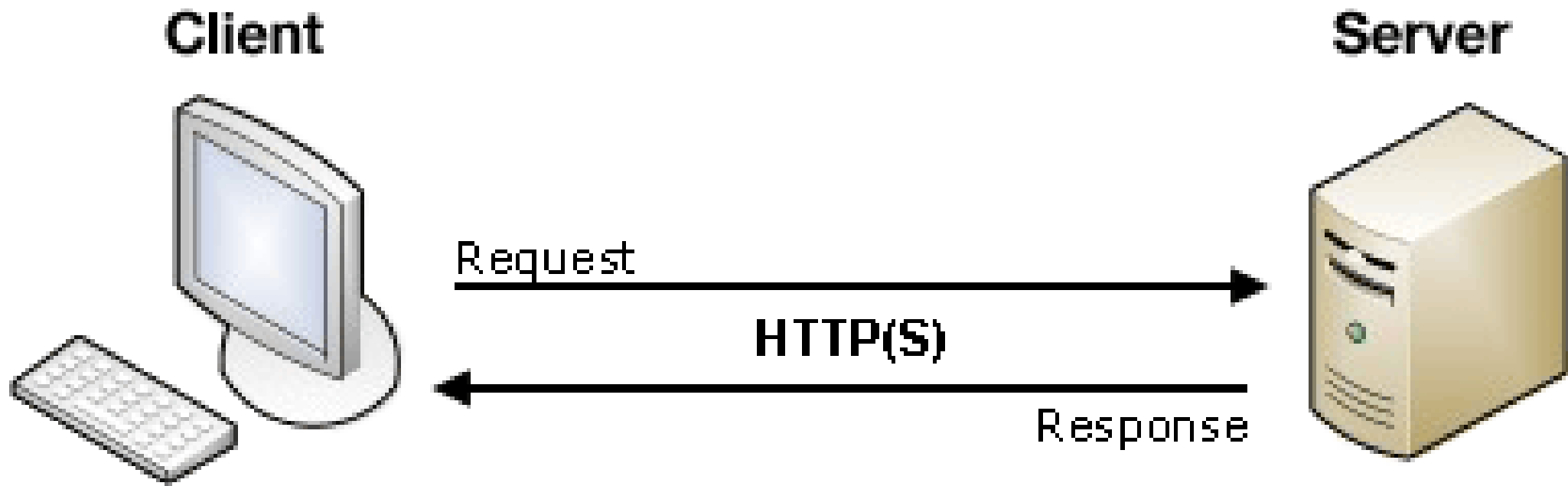
- **Paralelismo y concurrencia**

- Paralelismo - varios objetos realizando una misma tarea de forma simultánea.
- Concurrencia – un solo objeto con varias tareas activas, entre las que se va alternando (asíncrono).
- Node.js trabaja con un patrón concurrente Reactor (patrón de diseño)

- **Bloqueante y no bloqueante**

- Una tarea no bloquea a otra, se ejecuta cada tarea a su ritmo.

Envío de datos al servidor



Objeto request

- Una URL que identifica el servidor de destino y un recurso que puede ser un archivo HTML, un servicio Web o una herramienta a ejecutar.
- Un método que define la acción requerida, por ejemplo: obtener un archivo o guardar algunos datos.
- Verbos HTTP: GET, POST, HEAD, PUT, DELETE
- Las peticiones GET codifican los datos en la URL enviada al servidor añadiendo al final pares nombre/valor — por ejemplo,
`http://127.0.0.1:5000/index.html?name=Luis&edad=19.`

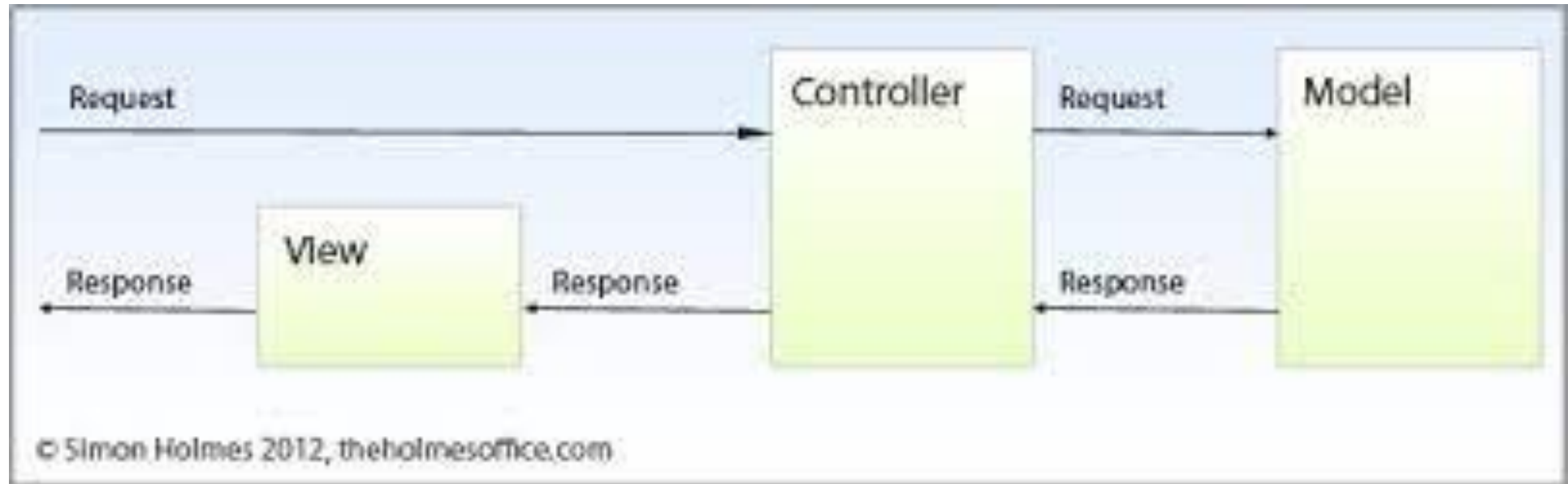
Objeto Respond

- La respuesta contiene un código de estado de respuesta HTTP que indica si la petición ha tenido éxito o no. Algunos estados son: el código 200 indica que la petición tuvo éxito, el código 400 indica que el recurso solicitado no se encontró y el código 403 indica que el usuario no está autorizado para acceder al recurso solicitado.
- El cuerpo de la respuesta de éxito a una petición GET contendría el recurso solicitado.
- Cuando se devuelve una página HTML, esta es renderizada por el navegador web. Si el navegador descubre que el documento hace referencia a otros recursos enviará peticiones HTTP separadas para descargar estos archivos, ejemplo (CSS y JS).



Framework Express

Arquitectura MVC



¿Qué es?

- Framework Web base de otros frameworks.
- Proporciona mecanismos para gestionar peticiones con verbos HTTP en diferentes rutas (URL).
- Permite integrar motores de renderización de "vistas" para generar respuestas mediante la introducción de datos en plantillas.
- Permite establecer ajustes de aplicaciones web como qué puerto usar para conectar, y la localización de las plantillas que se utilizan para renderizar la respuesta.
- Añadir procesamiento de peticiones "middleware" adicional para el manejo de la petición.

Otras características

- Express posee métodos para especificar que función ha de ser llamada dependiendo del verbo HTTP usado en la petición y la ruta especificada.
- Incluye métodos para especificar que plantilla ("view") o gestor de visualización utilizar, donde están guardadas las plantillas de HTML que han de usarse y como generar la visualización adecuada para cada caso.
- El middleware de Express, puede usarse también para añadir funcionalidades para la gestión de cookies, sesiones y usuarios, mediante el uso de parámetros, en los métodos POST/GET.

Instalación y configuración



Creando un proyecto para uso de express.js

Estructura de directorios

- public: contiene los archivos CSS, recursos multimedia, HTML.
- src: contiene los archivos JS de los códigos para la gestión de peticiones y respuestas de la aplicación Web

Inicializar el proyecto: creación del archivo package.json

- npm init o npm init –yes

Creación del archivo de inicio de la aplicación

- app.js o index.js generalmente

Asociar express al proyecto

Instalando express con npm y guardarlo en la lista de dependencias del proyecto

- `npm install express --save`
- En el archivo `app.js` o `index.js` se debe incluir el módulo Express y crear una aplicación de Express. El módulo posee métodos para el enrutamiento de las peticiones HTTP, configuración del 'middleware', y visualización de las vistas de HTML, uso de motores de 'templates'.

Código inicial del proyecto

```
var express = require('express');
var app = express();

app.get('/', function(req, res) {
  res.send('Hola Mundo!');
});

app.listen(3000, function() {
  console.log('Aplicación ejemplo, escuchando el puerto 3000!');
});
```