

BIENVENIDOS

» DIPLOMADO 2022-1 «

“Especialización en Tecnologías Móviles”

INICIO: 26 DE MARZO

Mobile Applications





Tema03

Node.js





¿Qué vamos a aprender hoy?

1. Introducción a Node.js
2. Http para servidor web en Node.js
3. Instalando Express
4. Creación de un Servidor Web con Express

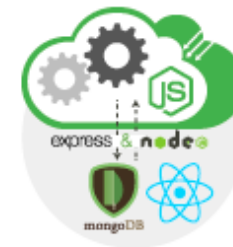




¿Qué es node.js?

- nodejs es una plataforma que se usa principalmente para desarrollar aplicaciones web utilizando javascript en el servidor.
- Es un entorno de desarrollo que sacó JavaScript del navegador.
- Node.js está basado en el motor V8 de Javascript de Google, el cual está diseñado para correr en un navegador y ejecutar el código de Javascript de una forma extremadamente rápida.
- Se han eliminado algunas funcionalidades que en el entorno de servidor no tenían sentido como por ejemplo el uso de Document Object Model.
- Node.js trabaja con un único hilo de ejecución que es el encargado de organizar todo el flujo de trabajo que se deba realizar.




node.js





Instalación Node.js

- <https://nodejs.org/es/download/>

LTS Recomendado para la mayoría	Actual Últimas características	
 Instalador Windows <small>node-v14.18.0-x64.msi</small>	 Instalador macOS <small>node-v14.18.0.pkg</small>	 Código Fuente <small>node-v14.18.0.tar.gz</small>

Instalador Windows (.msi)

Binario Windows (.zip)

Instalador macOS (.pkg)

Binario macOS (.tar.gz)

Binario Linux (x64)

Binario Linux (ARM)

Código Fuente

32-bit	64-bit
32-bit	64-bit
64-bit	
64-bit	
64-bit	
ARMv7	ARMv8
node-v14.18.0.tar.gz	





Express

- Es un framework para implementar sitios web.
- Este framework nos facilita y nos ordena el desarrollo de sitios web.
- El framework Express está constituido por un conjunto de módulos.

node JS[™]
express





Express: Instalación

- Para trabajar con Express lo primero es instalarlo en nuestro proyecto.

```
$ npm install express --save
```





Hola mundo con Express

- **Línea 1:** “require” es una instrucción que usaremos para determinar la necesidad de importar algo. Debemos crear este tipo de instancias en la parte superior del archivo.
- **Línea 2:** Estamos creando la aplicación rápida configurándola como variable de aplicación.
- **Línea 3:** “.get” está diciendo que cuando obtiene esa ruta debe dar la respuesta que se especifica en la función. Toma 2 argumentos: (1) la url (2) la función que le dice a express qué enviar a la persona que hace la solicitud.
- **Línea 6:** “.listen” vinculará la aplicación al puerto de escucha de nuestra máquina.

```
const express = require('express');
const app = express();
const port = 3000;
app.get('/', (req, res) => {
  res.send('Hola Mundo!')
});
app.listen(port, () => {
  console.log(`Listening on port ${port}`);
});
```

```
$ node app.js
```



Express - Definir rutas

- El ruteo hace referencia a la determinación de cómo responde una aplicación a una solicitud de cliente en un determinado punto final, que es un URI (o una vía de acceso) y un método de solicitud HTTP específico (GET, POST, etc.).
- Cada ruta puede tener una o varias funciones de manejador, que se excluyen cuando se correlaciona la ruta.
- La definición de ruta tiene la siguiente estructura:

```
app.METHOD(PATH, HANDLER)
```

- Donde:
 - **app** es una instancia de express.
 - **METHOD** es un método de solicitud HTTP.
 - **PATH** es una vía de acceso en el servidor.
 - **HANDLER** es la función que se ejecuta cuando se correlaciona la ruta.





Express - Métodos de ruta

- Un método de ruta se deriva de uno de los métodos HTTP y se adjunta a una instancia de la clase express.
- El siguiente código es un ejemplo de las rutas que se definen para los métodos GET y POST a la raíz de la aplicación.

```
// GET method route
app.get('/', function (req, res) {
  res.send('GET request to the homepage');
});
// POST method route
app.post('/', function (req, res) {
  res.send('POST request to the homepage');
});
```



Express: Enviar y obtener parámetros por GET

- El objeto **req** representa la solicitud HTTP y tiene propiedades para la cadena de consulta de la solicitud, parámetros, cuerpo, encabezados HTTP, etc.
- Para el siguiente ejemplo, el objeto se denomina **req** y la respuesta HTTP es **res**.

```
app.get('/usuario/:id', function (req, res) {  
    res.send('usuario con id ' + req.params.id);  
})
```

- En el código anterior, pasamos un parámetro por la url denominado "id", para recuperarlo, accedemos con el siguiente código:

```
req.params.id
```





Express: Obtener parámetros por POST

- Para poder obtener los datos enviados por el método POST, primero debemos definir que tipo de datos vamos a analizar:

```
var express = require('express');  
var app = express();  
  
// Para analizar application/json  
app.use(express.json());  
  
// Para analizar application/x-www-form-urlencoded  
app.use(express.urlencoded({ extended: true }));
```





Express: Obtener parámetros por POST

- Para poder obtener los datos enviados por el método POST, debemos acceder al cuerpo de la solicitud definiendo post como método de solicitud HTTP:

```
app.post('/perfil', function (req, res, next) {  
  res.json(req.body.nombres);  
  res.json(req.body.apellidos);  
})
```

- En el código anterior, se esta enviando las variables “nombres” y “apellidos”, para recuperarlos, accedemos a los datos con el siguiente código:

```
req.body.nombres;  
req.body.apellidos;
```

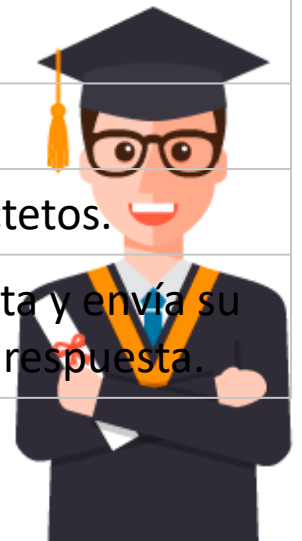




Métodos de respuesta

- Los métodos en el objeto de respuesta (res) de la tabla siguiente pueden enviar una respuesta al cliente y terminar el ciclo de solicitud/respuestas.
- Si ninguno de estos métodos se invoca desde un manejador de rutas, la solicitud de cliente se dejará colgada.

Método	Descripción
<u>res.download()</u>	Solicita un archivo para descargarlo.
<u>res.end()</u>	Finaliza el proceso de respuesta.
<u>res.json()</u>	Envía una respuesta JSON.
<u>res.jsonp()</u>	Envía una respuesta JSON con soporte JSONP.
<u>res.redirect()</u>	Redirecciona una solicitud.
<u>res.render()</u>	Representa una plantilla de vista.
<u>res.send()</u>	Envía una respuesta de varios tipos.
<u>res.sendFile()</u>	Envía un archivo como una secuencia de octetos.
<u>res.sendStatus()</u>	Establece el código de estado de la respuesta y envía su representación de serie como el cuerpo de respuesta.





express.Router

- Utilice la clase **express.Router** para crear manejadores de rutas montables y modulares.
- A continuación se crea un archivo de ruteo birds.js en el directorio de la aplicación, con el siguiente contenido:

```
var express = require('express');
var router = express.Router();

// define the home page route
router.get('/', function(req, res) {
  res.send('Birds home page');
});
// define the about route
router.get('/about', function(req, res) {
  res.send('About birds');
});
module.exports = router;
```

- Luego, se debe cargar el módulo de ruteador en la aplicación:

```
var birds = require('./birds');
...
app.use('/birds', birds);
```

- La aplicación ahora podrá manejar solicitudes a /birds y /birds/about.





¿Preguntas?

