# TEMA 1 INTRODUCCIÓN A NODE.JS

## 3- Las librerías o módulos

# LIBRERÍAS O MÓDULOS

- NODE.JS es un framework muy modularizado. Y en cada proyecto añadiremos los que necesitemos.
- MÓDULO=LIBRERÍA=PAQUETE
- Ejemplos:
  - http y https (Servidor Web)
  - Fs (Sistema de ficheros)
  - Utils (Formato de cadenas de texto)
- Todos: https://nodejs.org/api/

# Tipos de Módulos

- Los que incluye el NODE.JS.
  - Estan ya incluidos
- Los realizados por terceros (mongoose, que nos permite acceder a Mongo DB o EXPRESS, un framework que nos facilitará la vida para NODE.js).
  - Para incluirlos necesitaremos el gestor npm
- Los que nosotros mismos realicemos.

# Incluir Módulos. REQUIRE

- Se usa para incluir cualquier módulo.
- Ejemplo: Listar ficheros

```
const ruta = ‘c:/escritorio/pablo';
const fs = require('fs');
fs.readdirSync(ruta).forEach(fichero =>
    {console.log(fichero);});
```

# EJERCICIO 1

- Crea una carpeta llamada "Tema1_SaludoUsuario" en tu espacio de trabajo, dentro de la carpeta de "Ejercicios".
- Añade un archivo llamado "saludo.js".
- Echa un vistazo en la API de Node al módulo "os", y en concreto al método userInfo.
- Utilízalo para hacer un programa que salude al usuario que ha accedido al sistema operativo.
- Por ejemplo, si el usuario es "nacho", debería decir "Hola nacho".
- AYUDA: el método userInfo devuelve un *objeto* con varias propiedades del usuario que ha accedido. Para obtener el nombre del usuario, deberemos acceder a la propiedad username.
- AYUDA: el método console.log admite un número indefinido de parámetros, y los concatena uno tras otro con un espacio. *console.log("Hola", nombre,…);*

# Incluir nuestros propios módulos I

- En grandes proyectos, será muy recomendable que separemos en varios ficheros.
- Crea carpeta “PruebasRequire” dentro de PRUEBAS, dentro haremos 2 archivos:
  - principal.js
  - utilidades.js

# Incluir nuestros propios módulos II

- Dentro de principal.js Se incluye con REQUIRE indicando la ruta relativa, en nuestro caso:

  ```
  const utilidades = require('./utilidades.js');
  ```

- Ó también sin la extensión si va a ser código javascript:

  ```
  const utilidades = require('./utilidades');
  ```

- Dentro de utilidades.js imprimimos una frase y ejecutamos. (No es habitual poner sentencias directas)

# Como llamar desde otro fichero

- No basta con crear las funciones en un fichero, y luego desde otro usando require, llamarlas normalmente.
- Debemos usar module.exports
- Sintaxis:
  - module.exports.<nombre función>=<cuerpos de la función>

# Ejemplo. 2 Formas de hacerlo

- Usandolo en cada método:

```
module.exports.sumar = function(num1, num2) {return num1 + num2;};

module.exports.restar = function(num1, num2) {return num1 - num2};
```

- O por un lado los métodos, y por otro indicando el module.exports:

```
function sumar(num1, num2) {
return num1 + num2;
}

function restar (num1, num2) {
return num1 - num2
}

module.exports = {
sumar: sumar,
restar: restar
};
```

# Llamar desde el fichero principal

- Ahora sí podemos llamarlo desde otro fichero:

```
const utilidades = require('./utilidades');
console.log(utilidades.sumar(3, 2));
```

- También podriamos haber accedido a variables:

```
module.exports = {
pi: 3.1416,
sumar: sumar,
restar: restar
};
...
console.log(utilidades.pi);
```

# Funciones Arrows

- Se prescinde de la palabra function
- No tenemos acceso a this ni a arguments, si los necesitamos debemos usar las funciones tradicionales o anónimas.
- Expresiones Lambda. Por un lado los paréntesis (si es uno solo se puede prescindir de ellos) y por otro lado el cuerpo de la función, separados por una =>.

```
var sumar = (num1, num2) => {
return num1 + num2;
};
var restar = (num1, num2) => {
return num1 - num2
```

- Incluso en este caso podrías reducirse a:

```
var sumar = (num1, num2) => num1 + num2;
var restar = (num1, num2) => num1 – num2;
```

# EJERCICIO 2

- Crea una carpeta llamada "Tema1_VerificarUsuario" en tu espacio de trabajo, en la carpeta de "Ejercicios". Añade un archivo llamado "utilidades_os.js" y otro llamado "principal.js". Dentro de "utilidades_os.js" haz lo siguiente:
- Carga el módulo "os" (con require)
- Exporta una propiedad llamada "loginUsuario" que almacene el login del usuario que accedió al sistema, de forma similar a como lo obtuviste en un ejercicio anterior.
- Exporta un método llamado "esUsuario" que reciba como parámetro una cadena y devuelva un booleano dependiendo de si la cadena coincide con la propiedad "loginUsuario" o no.
- En el archivo "principal.js", haz lo siguiente:
  - Incluye el archivo "utilidades_os.js" hecho anteriormente (con require)
  - Llama al método "esUsuario" pasándole como parámetro el nombre "pepe", y escribe por pantalla un texto indicando si es ése el usuario logueado o no (en función de lo que te devuelva la llamada al método).
  - Utiliza la propiedad "loginUsuario" para mostrar por pantalla quién es el auténtico usuario logueado.
- Al ejecutar el programa principal, deberá mostrarte estos mensajes (suponiendo que tu usuario sea "nacho"):
  - El usuario no es 'pepe'
  - El usuario correcto es 'nacho'

# Incluir carpetas enteras

- Cuando tengamos muchos módulos, es recomendable organizarlos en carpetas.
- En estos casos es más cómodo usar REQUIRE que incluyan carpetas enteras.
- Necesitaremos un fichero index.js en cada carpeta que incluya todos los ficheros de la misma.
- Desde el programa principal, se podrá incluir el nombre de la carpeta, que automáticamente buscará el index.js correspondiente.

# Ejemplo. Incluir carpetas enteras

- Crea una carpeta ”idiomas” dentro de “PruebasRequire” y dentro habrá 3 ficheros:

- **Archivo "es.js":**

```
module.exports = {
saludo : "Hola"
};
```

- **Archivo "en.js":**

```
module.exports = {
saludo : "Hello"
};
```

- **Archivo "index.js":**

```
const en = require('./en');
const es = require('./es');


module.exports = {
es : es,
en : en
};
```

- Y para llamar a la carpeta entera **desde fuera:**

```
const idiomas = require('./idiomas');
console.log("English:", idiomas.en.saludo);
console.log("Español:", idiomas.es.saludo);
```

# JSON

- JSON como ya veremos es muy práctico. Para que sea más mantenible, el ejemplo anterior:
  - Archivo "saludos.json“

```
{
"es" : "Hola",
"en" : "Hello"
}
```

- Archivo "es.js":

```
const textos = require('./saludos.json');

module.exports = {
    saludo : textos.es
};
```

- Archivo "en.js":

```
const textos = require('./saludos.json');

module.exports = {
    saludo : textos.en
};
```