DESARROLLO DE APLICACIONES WEB AVANZADO

LABORATORIO N° 01

Configuración del Entorno de Node.js

Alumno(s):		***Nota***	
Grupo:		***Ciclo:V***	

Criterio de Evaluación	*Excelente (4pts)*	*Bueno (3pts)*	*Requiere mejora (2pts)*	*No acept. (0pts)*	*Puntaje Logrado*
Identifica las principales características de node.js					
Instala el servicio de node.js en Windows					
Ejecutar las páginas web propuestas					
Logra entender lo propuesto en los formularios JavaScript					
Es puntual y redacta el informe adecuadamente					

Laboratorio 1: Instalación de la plataforma node.js

Objetivos:

Al finalizar el laboratorio el estudiante será capaz de:
- Identificar las principales características del node.js
- Instalar node.js
- Identificar las características de utilizar Javascript en el lado del servidor.

Seguridad:

- Ubicar maletines y/o mochilas en el gabinete del aula de Laboratorio.
- No ingresar con líquidos, ni comida al aula de Laboratorio.
- Al culminar la sesión de laboratorio apagar correctamente la computadora y la pantalla, y ordenar las sillas utilizadas.

Equipos y Materiales:

- Una computadora con:
  - Windows 7 o superior
  - VMware Workstation 10+ o VMware Player 7+
  - Conexión a la red del laboratorio

- Máquinas virtuales:
  - Windows 7 Pro 64bits Español - Plantilla

- Instalador de node.js

Procedimiento:

Lab Setup

1. **Creación del equipo virtual**
   1.1. Encender el equipo
   1.2. Acceder empleando la cuenta de **usuario**: Tecsup, **contraseña**: _____
   1.3. Iniciar el Software VMWare.
   1.4. Abrir la plantilla ubicada en:
   E:\Equipos virtuales\Windows 7 Pro 64bits Español - Plantilla
   1.5. Crear un clon de la maquina anterior con los siguientes datos:
   1.6. Nombre del clon: **C15-BDAV**
   1.7. Ubicación del clon: **D:\C15-BDAV**
   1.8. Cerrar la plantilla
   1.9. Iniciar el equipo virtual **C15-BDAV**
   1.10. Identifíquese con la cuenta de usuario: **Redes**. Contraseña: **RCDTecsup2**

Instalación de node.js

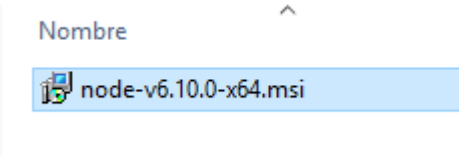
2. Obtención del instalador
   2.1. El instalador de node.js se puede obtener de la página oficial: https://nodejs.org/en/download/
   2.2. Veremos varias opciones, aparte de las plataformas de sistema operativo como son:

   - **LTS** (recomendada para la mayoría de usuarios): es la versión de Node.js con Long Term Support (LTS), es decir a la que se le da soporte a largo plazo. Esta versión puede no tener disponibles las últimas tecnologías que todavía no se consideran estables.
   - **Current**: esta es la versión más reciente de Node.js e incluye todas las funcionalidades, incluso aquellas más novedosas y que no se consideran estables.

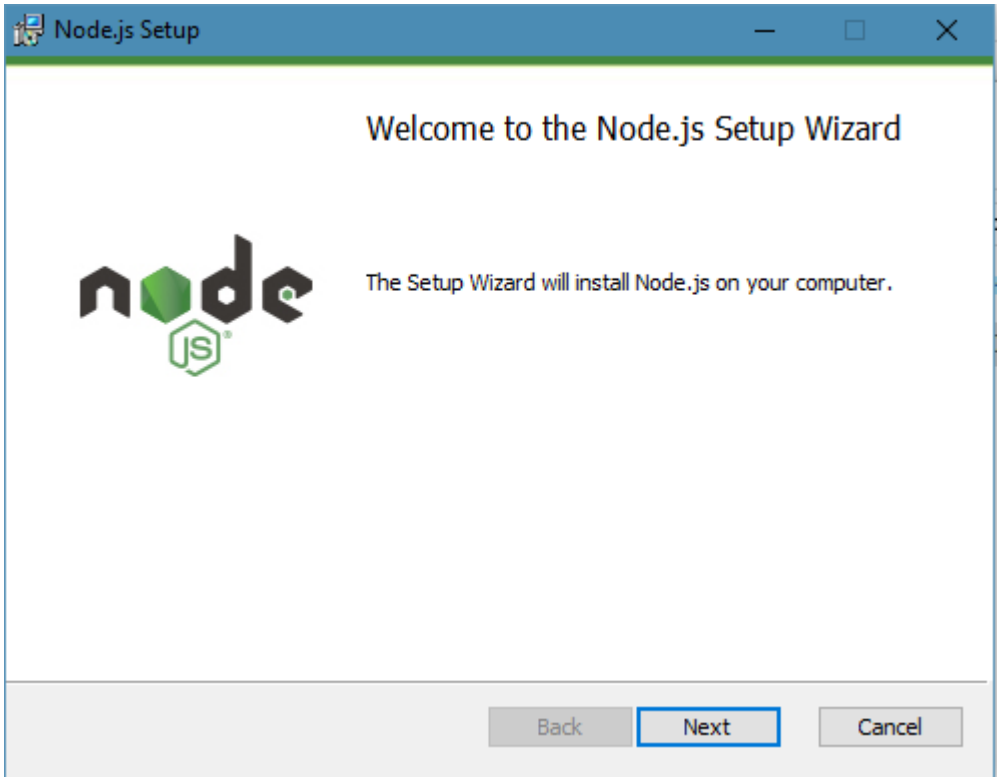
2.3. Por norma general, **seleccionar la versión LTS es la opción recomendada y estable.**
2.4. Esta sección es **informativa**, para ahorrar tiempo, el instructor le facilitará el archivo de instalación del node.js, aunque como podrá usted comprobar, el instalador no es tan pesado (aproximadamente unos 12MB)

3. **Proceso de instalación de node.js**

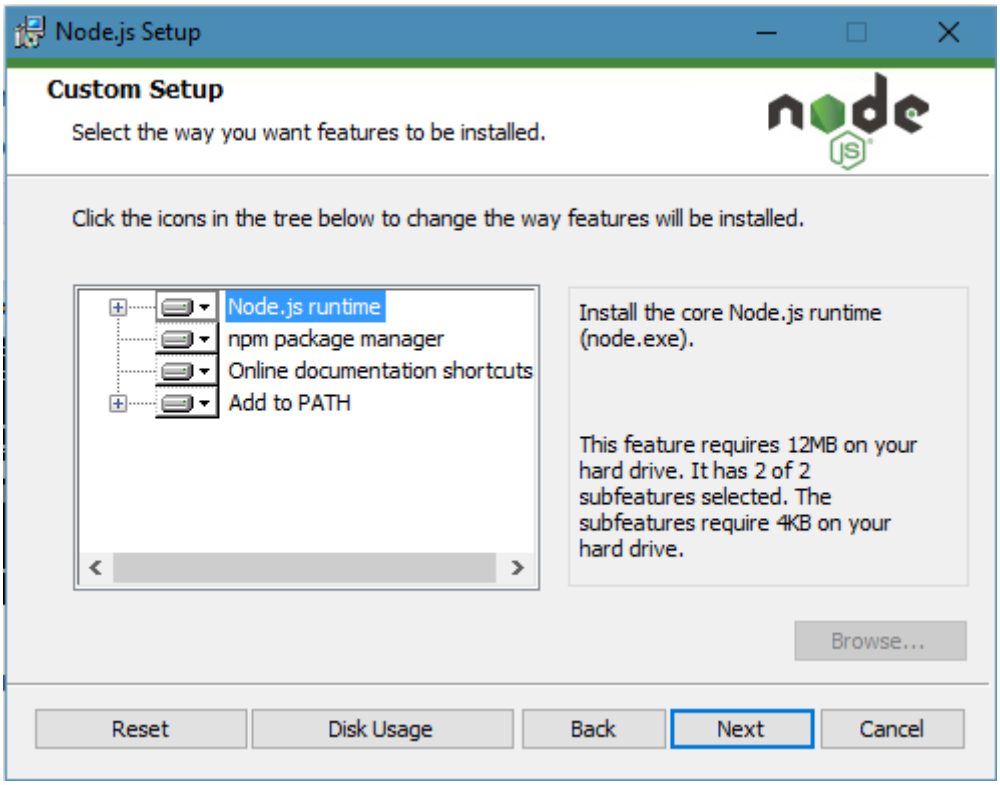
3.1. Solicite al instructor, el archivo de instalación de node.js



3.2. Copie el archivo instalador al escritorio del equipo virtual
3.3. Doble clic en el instalador para iniciar el proceso de instalación. Haga click en Next



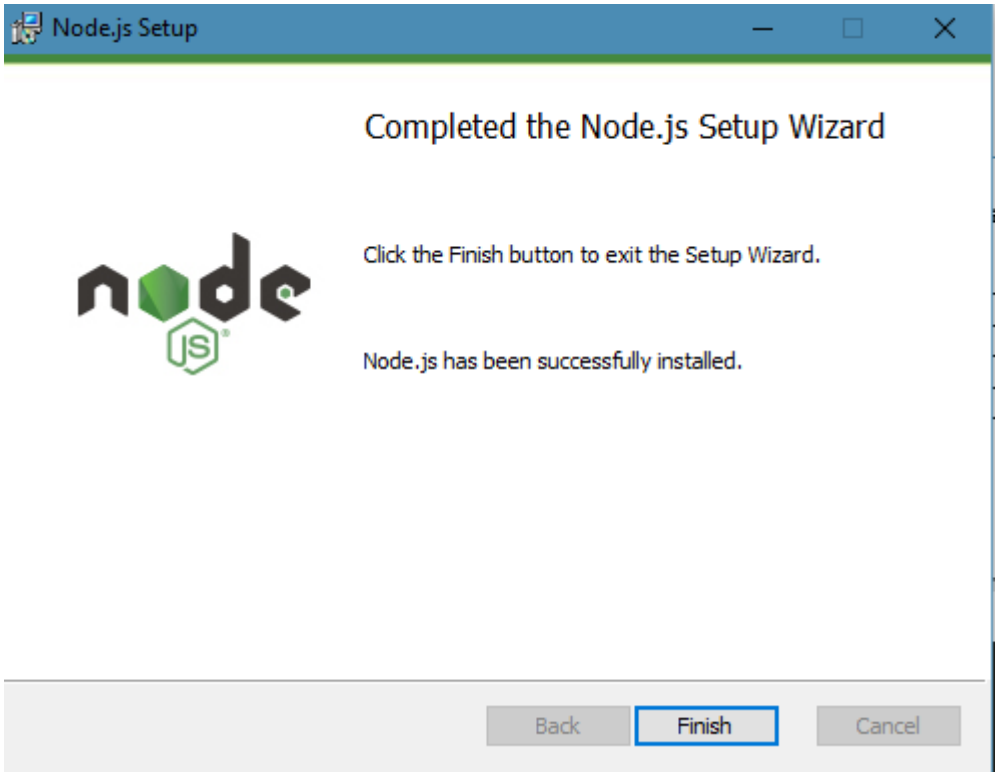
3.4. No deberá tener problemas en el instalador ya que se seleccionarán las opciones por defecto. Llegará usted a la siguiente pantalla:



Haga click en cada una de las características a instalar e indique que es lo que realizarán al ser instaladas:

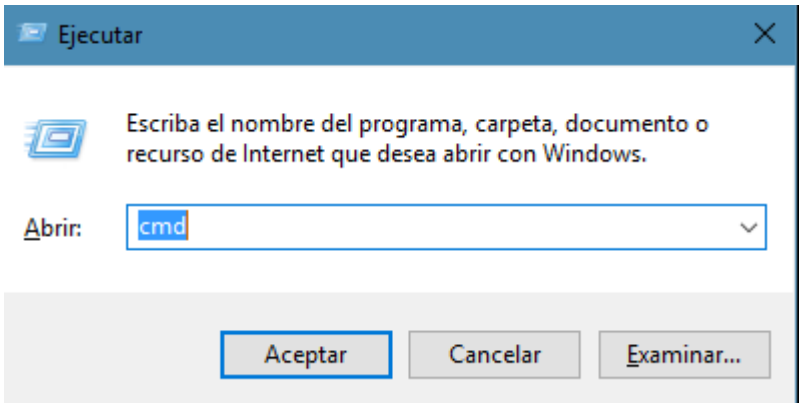
_____
_____
_____
_____
_____

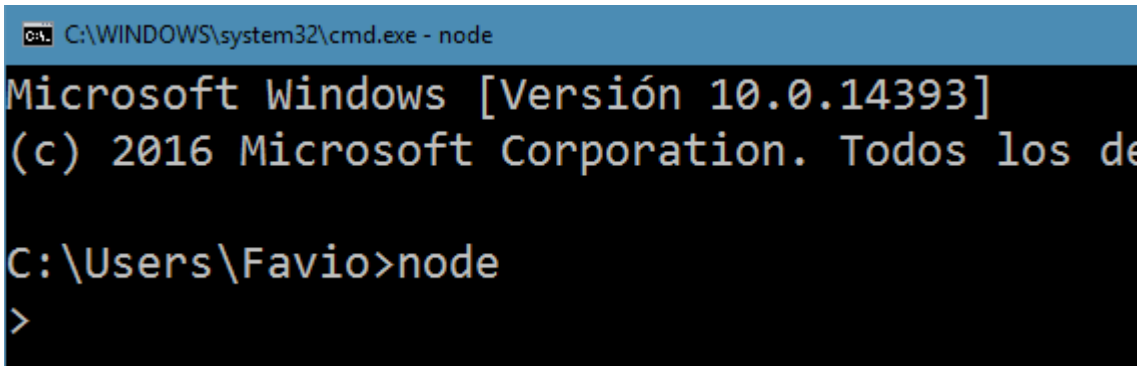
3.5. Siga con el proceso de instalación con las opciones por defecto.



4. Verificación de la instalación

4.1. Abra una consola de comandos y ejecutemos el comando **node**





4.2. Uso de JavaScript en la consola del Shell de MongoDB

4.2.1. Definir variables en mongoDB

```
> var a = 20, b = 30, c = 40;
```

4.2.2. Realizar operaciones aritméticas

```
> var suma = a + b;
> var resta = c - b;
> var producto = a * b;
> var cociente = a / b;
> var residuo = a % 7;
> 8 << 1;
> 8 >> 1;
> --a;
> ++b;
```

4.2.3. Visualizar valor de variables

```
> console.log(a, b, c, suma, resta, producto, cociente, residuo);
```

4.2.4. Visualizar el valor de una única variable en forma simple

```
> a
> b
> c
> residuo
```

4.2.5. Funciones de la librería javascript Math

```
> Math.min(0, 150, 30, 20, -8, -200);
> Math.max(0, 150, 30, 20, -8, -200);
> Math.random();
> Math.round(4.7);
> Math.round(4.4);
> Math.ceil(4.4);
> Math.floor(4.7);
```

4.2.6. Implemente algunos ejemplos donde haga uso de las siguientes funciones o constantes:
Math Object Methods

Method	Description
abs(x)	Returns the absolute value of x
acos(x)	Returns the arccosine of x, in radians
asin(x)	Returns the arcsine of x, in radians
atan(x)	Returns the arctangent of x as a numeric value between -PI/2 and PI/2 radians
atan2(y,x)	Returns the arctangent of the quotient of its arguments
ceil(x)	Returns x, rounded upwards to the nearest integer
cos(x)	Returns the cosine of x (x is in radians)
exp(x)	Returns the value of Ex
floor(x)	Returns x, rounded downwards to the nearest integer
log(x)	Returns the natural logarithm (base E) of x
max(x,y,z,...,n)	Returns the number with the highest value
min(x,y,z,...,n)	Returns the number with the lowest value
pow(x,y)	Returns the value of x to the power of y
random()	Returns a random number between 0 and 1
round(x)	Rounds x to the nearest integer
sin(x)	Returns the sine of x (x is in radians)
sqrt(x)	Returns the square root of x
tan(x)	Returns the tangent of an angle

Math Constants

```
Math.E          // returns Euler's number
Math.PI         // returns PI
Math.SQRT2      // returns the square root of 2
Math.SQRT1_2    // returns the square root of 1/2
Math.LN2        // returns the natural logarithm of 2
Math.LN10       // returns the natural logarithm of 10
Math.LOG2E      // returns base 2 logarithm of E
Math.LOG10E     // returns base 10 logarithm of E
```

4.2.7. Funciones de cadena JavaScript en MongoDB

String Methods

Method	Description
charAt()	Returns the character at the specified index (position)
charCodeAt()	Returns the Unicode of the character at the specified index
endsWith()	Checks whether a string ends with specified string/characters
fromCharCode()	Converts Unicode values to characters
includes()	Checks whether a string contains the specified string/characters
indexOf()	Returns the position of the first found occurrence of a specified value in a string
lastIndexOf()	Returns the position of the last found occurrence of a specified value in a string
match()	Searches a string for a match against a regular expression, and returns the matches
repeat()	Returns a new string with a specified number of copies of an existing string
replace()	Searches a string for a specified value, or a regular expression, and returns a new string where the specified values are replaced
search()	Searches a string for a specified value, or regular expression, and returns the position of the match
slice()	Extracts a part of a string and returns a new string
split()	Splits a string into an array of substrings
startsWith()	Checks whether a string begins with specified characters
substr()	Extracts the characters from a string, beginning at a specified start position, and through the specified number of character
substring()	Extracts the characters from a string, between two specified indices
toLowerCase()	Converts a string to lowercase letters
toUpperCase()	Converts a string to uppercase letters
trim()	Removes whitespace from both ends of a string

Ejemplos de uso:

> var cad = 'Tecsup Arequipa';

```
> cad.length
> cad.indexOf( 'qui' );
> cad.charAt(5);
> cad.includes( 'qui' );
> cad.substr( 3, 4 );
> String.fromCharCode( 234 );
```

4.2.8. Creación de funciones JavaScript en MongoDB

```
> fsuma = function(a,b) {
... return a+b;
... }

> fsuma(34,26);

> fresta = function (x,y) {
... return x-y;
... }

> fresta(20,5);
```

4.2.9. Para salir de la consola de node, utilizamos el comando:

```
> .exit
```

5. Hola mundo con node.js

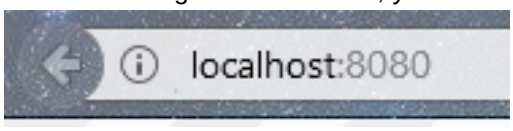
5.1. Cree el archivo **hola.js** en la unidad **C:** con el siguiente contenido:

```
var http = require('http');

var manejador = function(solicitud,respuesta){
    console.log('Hola mundo!');
};

var servidor = http.createServer(manejador);

servidor.listen(8080);
```

5.2. En una ventana de comandos sitúese en la unidad **C:** y escriba el siguiente comando:

```
>node hola.js
```

5.3. Abra un navegador de Internet, y escriba la siguiente URL:



5.4. Verificamos en la consola el resultado.

5.5. Anote sus conclusiones de lo que se acaba de realizar. Para finalizar la actividad del servidor, debe presionar **Ctrl + C**

5.6. A continuación, cree el archivo **hola.js**

```
var http = require('http');

var manejador = function(solicitud,respuesta){
    console.log('Conexion entrante');
    respuesta.end('Hola mundo!');
};

var servidor = http.createServer(manejador);

servidor.listen(8080);
```

5.7. Una vez más, ejecute el comando:

```
>node hola.js
```

5.8. Anote sus observaciones y mencione en que se diferencia de la versión anterior del archivo.

5.9. Ahora creemos el archivo **bucle.js** con el siguiente contenido:

```
var http = require('http');

var manejador = function(solicitud,respuesta){
    var i = 0;
    while(true){
        respuesta.write(i+'-->');
        i++
    }
};

var servidor = http.createServer(manejador);

servidor.listen(8080);
```

5.10. Ejecute node.js apuntando al nuevo archivo. Anote sus observaciones de acuerdo a lo que sucede en este caso.

6. Leer archivo de servidor

6.1. Creamos el archivo **index.html** con el siguiente contenido:

```
<html>
<head>
    <title>Pagina de ejemplo</title>
</head>
<body>
    <h1>Hola mundo con HTML!</h1>
</body>
</html>
```

6.2. Creamos el archivo **web.js** con el siguiente contenido

```
var http = require('http'),
fs = require('fs');

var html = fs.readFileSync('./index.html');

http.createServer(function(solicitud,respuesta){
    respuesta.write(html);
    respuesta.end();
}).listen(8080);
```

6.3. Ejecútelo y anote sus observaciones.

6.4. A continuación, modificaremos e archivo para que luzca de la siguiente manera.

```
var http = require('http'),
fs = require('fs');

http.createServer(function(solicitud,respuesta){
    fs.readFile('./index.html',function(error,html){
        respuesta.write(html);
        respuesta.end();
    });
}).listen(8080);
```

6.5. Busque en Internet la diferencia entre los comandos **readFileSync** y **readFile** y explique cuál es la diferencia entre ambos.

Uso de la librería HTTP

7. Pruebas de envío de respuesta al navegador

7.1. Crear el archivo **form.html** con el siguiente contenido:

```
<html>
    <head>
        <title>Servidor de Prueba</title>
    </head>
    <body>
        <span>Hola {nombre}!</span>
    </body>
</html>
```

7.2. Cree el archivo **variables.js** con el siguiente contenido:

```
var http = require('http'),
    fs = require('fs');

var parametros = [],
valores = [];

http.createServer(function(req,res){
    fs.readFile('./form.html',function(err,html){
        var html_string = html.toString();

        if(req.url.indexOf('?')>0){
            var url_data = req.url.split('?');
            arreglo_parametros = url_data[1].split('&');
        }

        for(var i=0; i<arreglo_parametros.length; i++){
            var parametro = arreglo_parametros[i];
            var param_data = parametro.split('=');
            parametros[i] = param_data[0];
            valores[i] = param_data[1];
        }

        for(var i=0; i<parametros.length; i++){
            html_string = html_string.replace('{'+parametros[i]+'}',valores[i]);
        }

        res.writeHead(200,{'Content-type':'text/html'});
        res.write(html_string);
        res.end();
    });
}).listen(8080);
```

7.3. Al acceder a la URL deberíamos tener un error en la consola. Indique cuál es, soluciónelo y adjunte una explicación de que es lo que sucede.

_____
_____
_____
_____
_____

_____
_____

7.4. Ahora acceda a la URL de nuestro servidor pero agregando lo siguiente a la dirección:

7.5. Incluya ahora los valores de las siguientes variables al finalizar la ejecución del archivo.

- arreglo_parametros: _____
- parametros: _____
- valores: _____

7.6. Explique lo que se ha realizado y lo que se puede realizar al manipular el objeto **request** del **createServer**, que es una función del módulo **HTTP** de **node.js**

_____
_____
_____
_____

8. Creación de módulo básico:

8.1. Cree el archivo **parser_var.js** con el siguiente contenido:

```
function parse_vars(req){
    var arreglo_parametros = [],
    parametros = [],
    valores = [];

    if(req.url.indexOf('?')>0){
        var url_data = req.url.split('?');
        var arreglo_parametros = url_data[1].split('&');
    }

    for(var i=0; i<arreglo_parametros.length; i++){
        var parametro = arreglo_parametros[i];
        var param_data = parametro.split('=');
        parametros[i] = param_data[0];
        valores[i] = param_data[1];
    }

    return {
        parametros: parametros,
        valores: valores
    };
}

module.exports.parse_vars = parse_vars;
```

8.2. Cree el archivo importar.js con el siguiente contenido:

```
var http = require('http'),
    fs = require('fs')
    parser = require('./parser_vars.js'),
    p = parser.parse_vars;

http.createServer(function(req,res){
    fs.readFile('./form.html',function(err,html){
        var html_string = html.toString();

        var respuesta = p(req),
        parametros = respuesta['parametros'],
        valores = respuesta['valores'];

        for(var i=0; i<parametros.length; i++){
            var html_string = html_string.replace('{'+parametros[i]+'}',valores[i]);
        }

        res.writeHead(200,{'Content-type':'text'});
        res.write(html_string);
        res.end();
    });
}).listen(8080);
```

8.3. Observe lo que sucede al llamar a esta invocación con la URL utilizada en el punto **2.4**

8.4. Todo código se puede optimizar. Indique como mejoraría esta función que permite retornar las variables de a URL.

_____
_____
_____

8.5. Modifiquemos form.html para que tenga la siguiente forma:

```
<html>
    <head>
        <title>Servidor de Prueba</title>
    </head>
    <body>
        <span>Hola {nombre}!</span><br />
        <ul>
            <li>{identidad}</li>
            <li>{poder}</li>
        </ul>
    </body>
</html>
```

8.6. Agreguemos las siguientes líneas al final del archivo **parser_vars.js**

```
module.exports.batman = {
    identidad: 'Bruce Wayne',
    poder: 'Dinero'
};
```

8.7. Finalmente modificaremos el archivo importar.js para que luzca así

```
var http = require('http'),
    fs = require('fs')
    parser = require('./parser_vars.js'),
    p = parser.parse_vars,
    datos = parser.batman;

http.createServer(function(req,res){
    fs.readFile('./form.html',function(err,html){
        var html_string = html.toString();

        var respuesta = p(req),
        parametros = respuesta['parametros'],
        valores = respuesta['valores'];

        for(var i=0; i<parametros.length; i++){
            html_string = html_string.replace('{'+parametros[i]+'}',valores[i]);
        }

        html_string = html_string.replace('{identidad}',datos['identidad']);
        html_string = html_string.replace('{poder}',datos['poder']);

        res.writeHead(200,{'Content-type':'text'});
        res.write(html_string);
        res.end();
    });
}).listen(8080);
```

8.8. Como se puede observar, además de poder reciclar código de funciones, podemos reutilizar variables. Agregue sus observaciones de cuando esto puede ser útil:

_____
_____
_____

9. Ejercicios de implementación de módulos

9.1. Adjunte el código de un módulo que muestre la hora en distintos formatos.

9.2. Adjunte el código de un módulo que calcule la cantidad de días faltantes para una fecha. Deberemos ingresar una fecha en nuestra URL y a partir de ella calcular los días faltantes.

9.3. Adjunte el código de un enrutador de URL. La idea es poder darle una dirección y que nos muestre una página HTML, por ejemplo:

- **/inicio** mostrará inicio.html
- **/galería** mostrará fotos.html

10. Finalizar la sesión

10.1. Apagar el equipo virtual

10.2. Apagar el equipo

Conclusiones:

Indicar las conclusiones que llegó después de los temas tratados de manera práctica en este laboratorio.