

**DESARROLLO DE APLICACIONES WEB AVANZADO**

**LABORATORIO N° 02**

**Aplicaciones de los Módulos del Core y REPL**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Alumno(s):*** | ***Sosa Cruz Joshua Jesús*** | | | | | | ***Nota*** |  | |
| ***Grupo:*** |  | | ***Ciclo:V*** | | | | | | |
| ***Criterio de Evaluación*** | | ***Excelente (4pts)*** | | ***Bueno (3pts)*** | ***Requiere mejora (2pts)*** | ***No acept. (0pts)*** | | | ***Puntaje Logrado*** |
| Identifica la utilidad de acceder directamente a las respuestas de request de HTTP | |  | |  |  |  | | |  |
| Manipula los elementos del core de node.js | |  | |  |  |  | | |  |
| Exporta con éxito sus propios módulos | |  | |  |  |  | | |  |
| Logra entender lo propuesto en los formularios JavaScript | |  | |  |  |  | | |  |
| Es puntual y redacta el informe adecuadamente | |  | |  |  |  | | |  |

**Laboratorio 2: Aplicaciones de los**

**Módulos del Core y REPL**

**Objetivos:**

Al finalizar el laboratorio el estudiante será capaz de:

* Identificar las principales características del módulo HTTP de node.js
* Manipulación de variables y funciones comunes del core de node.js
* Exportar sus propios módulos.

**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**

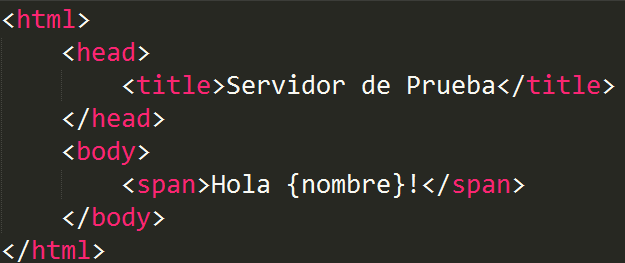
1. **Creación del equipo virtual**
   1. Encender el equipo
   2. Acceder empleando la cuenta de **usuario**: Tecsup, **contraseña**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
   3. Iniciar el Software VMWare.
   4. Abrir la plantilla ubicada en:

**E:\Equipos virtuales\Windows 7 Pro 64bits Español - Plantilla**

* 1. Crear un clon de la maquina anterior con los siguientes datos:
  2. Nombre del clon: **C15-DAWA**
  3. Ubicación del clon: **D:\C15-DAWA**
  4. Cerrar la plantilla
  5. Iniciar el equipo virtual **C15-DAWA**
  6. Identifíquese con la cuenta de usuario: **Redes**. Contraseña: **RCDTecsup2**
  7. Solicite el instalador de node.js al instructor e instálelo.

**Uso de la librería HTTP**

1. Pruebas de envío de respuesta al navegador
   1. Crear el archivo **form.html** con el siguiente contenido:



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

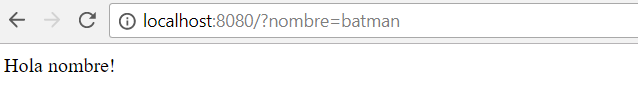


* 1. Al ejecutar el comando ***node variables.js*** y acceder a la URL ***http://localhost:8080*** deberíamos tener un error en la consola. Indique cuál es, soluciónelo y adjunte una explicación de que es lo que sucede.

***La variable arreglo\_parametros no se encontraba declarada por lo que se procedió a declararla.***

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

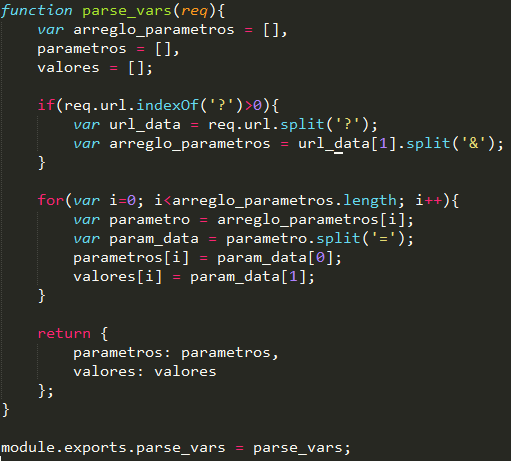




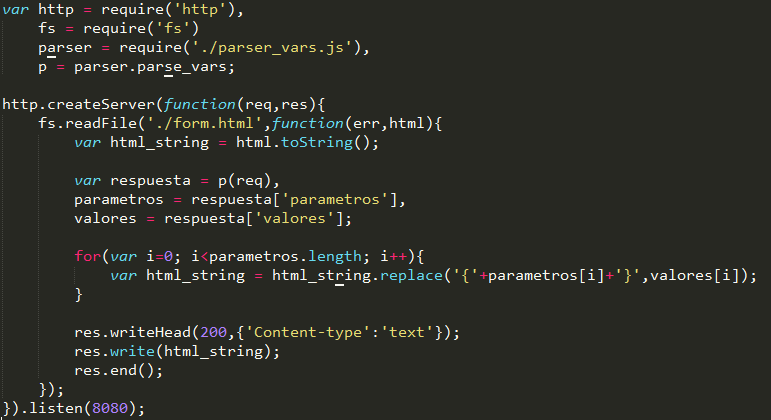
* 1. Incluya ahora los valores de las siguientes variables al finalizar la ejecución del archivo.
* arreglo\_parametros: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
* parametros: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
* valores: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
  1. 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**

***Se utilizó console.log para poder observar los cambios.***

1. **Creación de módulo básico:**
   1. Cree el archivo **parser\_var.js** con el siguiente contenido:



* 1. Cree el archivo importar.js con el siguiente contenido:



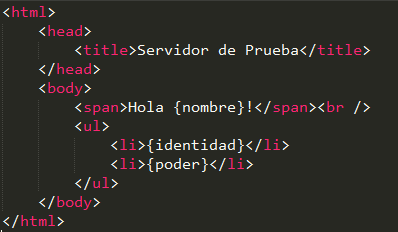
* 1. Observe lo que sucede al llamar a esta invocación con la URL utilizada en el punto **2.4**



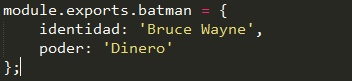
* 1. Todo código se puede optimizar. Indique como mejoraría esta función que permite retornar las variables de a URL.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

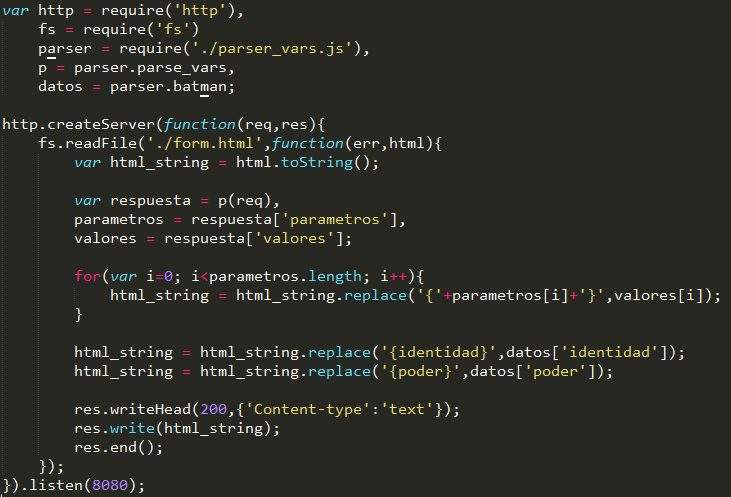
* 1. Modifiquemos form.html para que tenga la siguiente forma:



* 1. Agreguemos las siguientes líneas al final del archivo **parser\_vars.js**



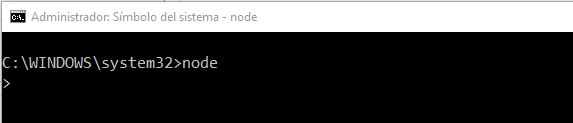
* 1. Finalmente modificaremos el archivo importar.js para que luzca así



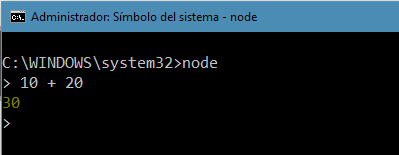
* 1. Como se puede observar, además de poder reciclar código de funciones, podemos reutilizar variables. Agregue sus observaciones de cuando esto puede ser útil:

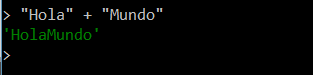
***Esto nos servirá para utilizar funciones para más de un caso.***

1. **Ejercicios de implementación de módulos**
   1. Adjunte el código de un módulo que muestre la hora en distintos formatos (investigue sobre la función ***date***).
   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 (investigue sobre ***timestamp*** para facilitar esta labor).
   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
2. Pruebas básicas con REPL
   1. Acceda al REPL con el comando node:

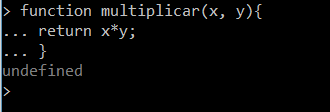


* 1. Realice pruebas de los comandos básicos de JavaScript para probar la eficiencia del REPL





* 1. Verifique el funcionamiento de declaración de funciones en el REPL, adjunte sus resultados y conclusiones.



* 1. Cree un archivo llamado **mes.js** y desarrolle una función que devuelva el primer y último día de un mes ingresado. Solamente puede admitir valores válidos (es decir, 1, 2, 3, no puede admitir 15, enero o 0).
  2. Para cargar dicho archivo al REPL, basta con usar el comando:



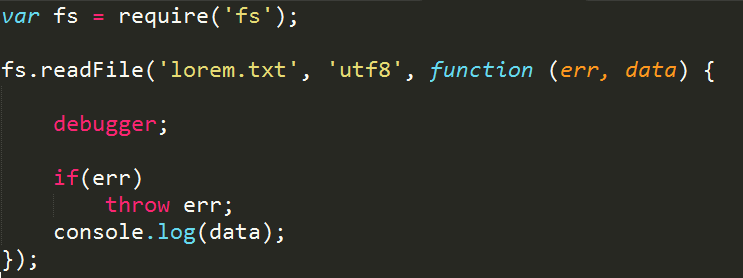
* 1. Para salir de la terminal del REPL, presione CTRL + C dos veces, o escriba .exit
  2. Entre nuevamente al REPL. Desarrolle una función **navidad()** que determine la cantidad de días faltantes hasta Navidad del presente año.
  3. Para guardar este archivo, utilice el comando **navidad.js**



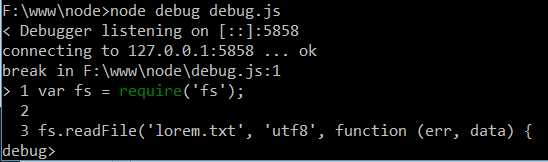
* 1. Salga de la terminal REPL e ingrese nuevamente. Una vez en ella, ingrese el siguiente comando y verifique que se puede cargar la función antes guardada y sobre todo, que está en funcionamiento.



1. Uso de la función debugger
   1. Cree el siguiente archivo con el nombre **debug.js**



* 1. Así mismo, cree el archivo **lorem.txt** con un texto de ejemplo.
  2. Ahora, ejecutemos el archivo con el siguiente comando. Anote sus observaciones.



* 1. Para pasar a la siguiente línea, ingrese el comando next



* 1. Para Finalizar en caso no haya otra parada, use cont



* 1. Utilizamos **watch(‘parametro’)** para poder observar alguna variable. Observemos el estado de la variable data.



|  |  |
| --- | --- |
| **Command** | **Description** |
| next | Stop at the next statement. |
| cont | Continue execute and stop at the debugger statement if any. |
| step | Step in function. |
| out | Step out of function. |
| watch | Add the expression or variable into watch. |
| watcher | See the value of all expressions and variables added into watch. |
| Pause | Pause running code. |

1. **Finalizar la sesión**
   1. Apagar el equipo virtual
   2. Apagar el equipo

**Tarea:**

1. Utilizando lo aprendido en el laboratorio, implemente las siguientes utilidades haciendo uso de módulos que reciclen las funciones solicitadas, incluyendo módulos que manejen las variables de los formularios y de procesamiento de URL.
   1. Una calculadora con las operaciones básicas.
   2. Un procesador de texto con funciones básicas como: dividir palabra, extraer cadena de texto, eliminar espacios, capitalizar, minúsculas, mayúsculas, contar caracteres, etc.
2. Haciendo uso de NPM (es el gestor de módulos de node.js, el cuál desarrollaremos en la siguiente sesión), instalemos **node-inspector**



***Node Inspector es una interfaz de depurador para aplicaciones Node.js que utiliza las herramientas de desarrollo de Blink (anteriormente WebKit Web Inspector).***

***Desde la versión 6.3, Node.js proporciona un depurador basado en DevTools incorporado que en su mayoría desaprueba el Inspector de nodos, consulte p. Ej. esta publicación del blog para comenzar. El depurador integrado es desarrollado directamente por el equipo V8 / Chromium y proporciona ciertas características avanzadas (por ejemplo, seguimientos de pila larga / asíncrona) que son demasiado difíciles de implementar en Node Inspector.***

1. Investigue in Internet la funcionalidad de ***node-inspector*** y adjunte un ejemplo de su utilización.

**Conclusiones:**

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

|  |
| --- |
| * *Se utilizó conocimientos aprendidos anteriormente para implementar el presente laboratorio.* * *Se pudieron crear distintos módulos.* * *Se utilizó una función para ser implementada en distintos procesos del laboratorio.* * *Se ejecutaron archivos del tipo “js.* * *Se creó una calculadora usando lo conocimientos aprendidos en el presente laboratorio.* |