

Curso de Node.js, Javascript y Git

Requerimientos para el curso

- Instalar SourceTree para el SO que corresponda (<https://www.sourcetreeapp.com/>)
- Crear sus usuarios para el repositorio de Bitbucket usando la invitación que recibieron por correo
- Descargar el instalador de Node para al SO que corresponda (<https://nodejs.org/>)
- Descargar el instalador de MongoDB para el SO que corresponda (<https://www.mongodb.org/downloads>)
- Instalar un editor de textos de preferencia:
 - o Sublime Text 2 (<http://www.sublimetext.com/2>)
 - o Notepad++ (<https://notepad-plus-plus.org/>)
 - o Brackets (<http://brackets.io/>)
 - o Bloc de notas (=P)

Temario

Día 1

1. ¿Qué es Node.js?
2. Instalación y crear mi primer programa “Hola mundo”
3. Introducción a Javascript Programación Orientada a Objetos

Día 2

4. Ejercicios con Javascript utilizando Node.js
5. Instalación de Express y crear un proyecto

Día 3

6. Manejar rutas y peticiones http
7. ¿Qué es MongoDB?
8. Instalación de MongoDB y ejercicio con Node.js

Día 4

9. ¿Qué es Socket.io y la librería NET?
10. Instalación de Socket.io y ejercicio con Node.js
11. Ejercicio con la librería NET

Día 5

12. Control de versiones con Git
13. Ejercicio con Javascript y Node.js utilizando Git

Lugar, Fecha y horario:

Del Lunes 31 de Agosto al Viernes 4 de Septiembre.

De las 8:00 a 10:00am

Sala de VideoConferencias (A2)

Contenido

Contenido	2
¿Qué es Node.js?	3
¿Qué se puede hacer con node.js?	3
¿Qué problemas resuelve node.js?	3
¿Por qué Javascript?	3
Características de node.js	3
¿En donde se utiliza node.js actualmente?	4
Notas interesantes para leer:	5
Tecnologías relacionadas con node.js	5
Instalación de node.js	6
Creando mi primer programa en node.js	7
¿Cómo ejecuto un programa en node.js?	7
Introducción a Javascript	8
Conceptos básicos sobre variables	9
Significado de undefined, null y NaN	10

¿Qué es Node.js?

Node.js es un software de código abierto que funciona como framework del lado del servidor, que utiliza para la ejecución del código interpretado de Javascript el motor V8 Javascript de Chrome. Node.js utiliza una arquitectura orientada a eventos.

Node.js permite que el servidor y las aplicaciones de escritorio se comuniquen por medio de Javascript.

En general podemos decir que Node.js es un entorno de ejecución y una biblioteca, que nos permite crear aplicaciones altamente escalables. Esto último es la meta número uno declara por node.js, “proporcionar una manera fácil para construir programas de red escalables”.

¿Qué se puede hacer con node.js?

Podemos generar desde pequeños módulos, hasta aplicaciones complejas, ya que node.js permite manejar cientos de miles de conexiones concurrentes; así mismo, por la manera en que node.js está implementado, es totalmente adecuado para lo que tiene que ver con intercomunicación a gran escala, como aplicaciones de mensajería, juegos multijugador y aplicaciones en tiempo real.

¿Qué problemas resuelve node.js?

Node.js modifica la forma en que se realiza una conexión con el servidor, ¿cómo?, hace que en lugar de generar un nuevo hilo de OS para cada conexión (y de asignarle la memoria acompañante), cada conexión dispara una ejecución de evento dentro del proceso del motor de node. Node afirma que un servidor que lo ejecute puede soportar decenas de miles de conexiones concurrentes.

¿Por qué Javascript?

JavaScript es un gran lenguaje para la programación asíncrona, ya que fue diseñado para ser usado en programación orientada a eventos en lugar de otros lenguajes orientados a objetos, como por ejemplo, Java. Es especialmente atractivo para realizar aplicaciones ‘no bloqueantes’ y de alta concurrencia y disponibilidad.

Características de node.js

- Se utiliza Javascript del lado del servidor
- Utiliza un ciclo de eventos en lugar de hilos nuevos por cada conexión
- Cada operación de I/O de node es asíncrona
- Puede manejar miles de conexiones simultáneas con una sobrecarga mínima en un solo proceso
- Amplia gama de herramientas a través de npm
- Ahorro en infraestructura. Especialmente en el número de servidores necesarios para las aplicaciones. Por ejemplo, cuando LinkedIn migró todo su backend desde Ruby on Rails hacia Node.js redujo desde 30 hasta 3 su número de servidores. (<http://www.nodehispano.com/2012/10/linkedin-migra-desde-rails-hacia-node-nodejs/>)

- Conjunción entre BackEnd y FrontEnd. Los equipos de BackEnd y FrontEnd móvil pueden ser combinados en un conjunto único.
- Flexibilidad.
- Fácil de usar. Sobre todo si tienes experiencia con JavaScript.

¿En donde se utiliza node.js actualmente?



Figura 1 Ejemplos de empresas que utilizan node.js actualmente

En el siguiente link se puede ver un listado más extenso de las empresas que utilizan node.js

<https://github.com/joyent/node/wiki/projects,-applications,-and-companies-using-node>

Notas interesantes para leer:

<https://www.talentbuddy.co/blog/building-with-node-js-at-netflix/>

<http://venturebeat.com/2012/01/24/why-walmart-is-using-node-js/>

<https://www.paypal-engineering.com/2013/11/22/node-js-at-paypal/>

<http://venturebeat.com/2011/08/16/linkedin-node/>

<http://venturebeat.com/2012/01/24/node-at-google-mozilla-yahoo/>

<http://highscalability.com/blog/2012/10/4/linkedin-moved-from-rails-to-node-27-servers-cut-and-up-to-2.html>

Tecnologías relacionadas con node.js



Figura 2 Tecnologías relacionadas con node.js

En conclusión node.js es una plataforma basada en el motor de JavaScript V8 de Google que es utilizado en el navegador Chrome. Está pensada para facilitar el desarrollo de aplicaciones basadas en red, rápidas y fiables. Node.js utiliza un modelo I/O (entrada/salida) orientado a eventos y basado en el 'no-bloqueo', que lo hace ligero y eficiente, ideal para aplicaciones en tiempo real que hagan uso de datos intensivos y que se ejecuten a través de dispositivos distribuidos.

Instalación de node.js

El proceso de instalación para los diferentes SO es sencilla, como ejemplo se indican algunos de los pasos para el caso de Windows.

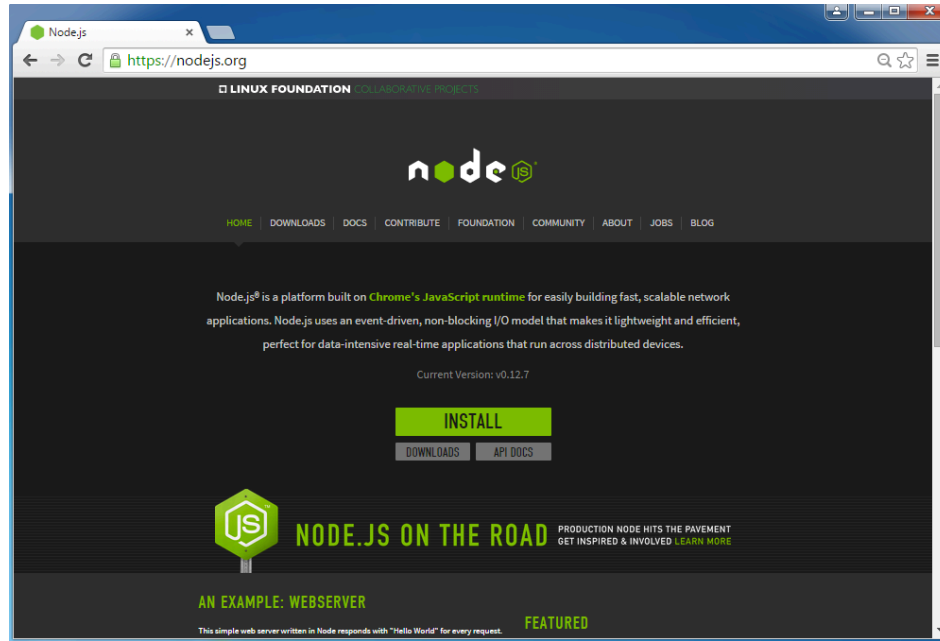
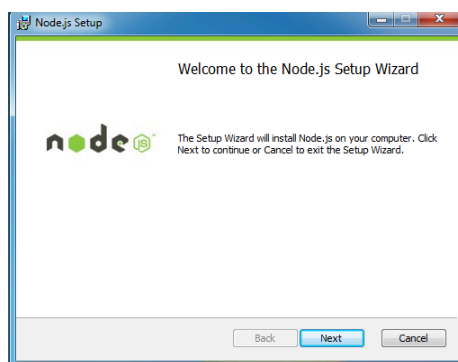


Figura 3 Página de descarga de node.js

Una vez que se ha descargado del instalador de node.js, simplemente damos inicio al proceso, lanzando el ejecutable y avanzamos con el típico next, next, finish.



next → next → finish

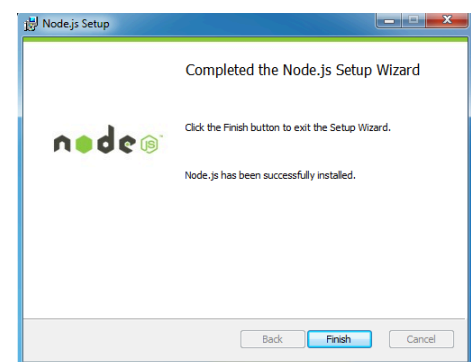
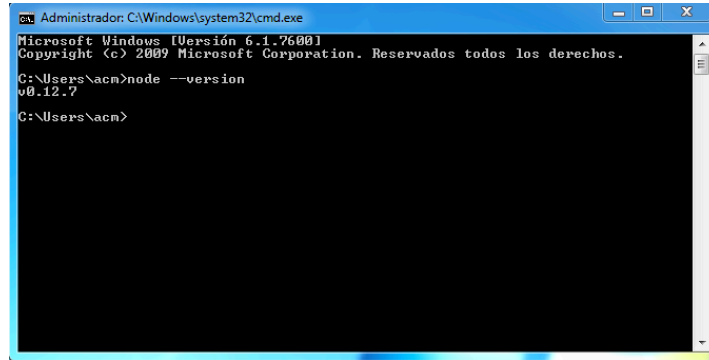


Figura 4 Proceso de instalación de node.js

Para verificar que node.js ha quedado instalado en nuestro equipo, podemos consultar qué versión es la que se ha instalado.

Para ello, vamos a la consola y tecleamos: **node --version <enter>**



```
Administrador: C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.
C:\Users\acm>node --version
v0.12.7
C:\Users\acm>
```

Figura 5 Respuesta de node.js a la consulta sobre el número de versión

¡ Listo ¡ Ya tenemos node.js para trabajar 😊 y ¿ahora? 🤔

Tenemos 2 opciones, podemos comenzar a trabajar con node.js desde la consola, para lo cual bastará desde la línea de comandos de la terminal, teclear **node <enter>**, peeeeeero.... mejor trabajemos desde ese editor de textos favorito que tienes.

Por cierto, si entraste a la consola de node.js, para salir de ahí, solo pulsa 2 veces **<ctrl> + c**.

Creando mi primer programa en node.js

Bueno pues manos a la obra. Utilizando tu editor de textos, vamos a crear un primer programa sencillo que nos imprima un mensaje en la pantalla. Para ello, creamos un nuevo documento y lo vamos a guardar utilizando la extensión js, por ejemplo **programa01.js**. Así es, ya imaginarás que vamos a poner en su interior código Javascript.

Bien, pues lo primero que vamos a hacer es nuestro “Hola mundo” en donde requeriremos escribir la siguiente línea:

```
console.log('Hola mundo');
```

Guardamos nuestro archivo y salimos a la consola en el lugar donde se encuentra nuestro programa que acabamos de crear.

¿Cómo ejecuto un programa en node.js?

Es sencillo, desde la línea de comando escribes: **node <nombre del archivo de extensión js> <enter>**

El resultado de nuestro primero programa lo veremos en la consola.

¡Genial!, ya sabemos como poner en marcha nuestros programas utilizando node.js y Javascript.

Introducción a Javascript

Como mencionamos al inicio, node.js es un entorno que proporciona las condiciones para que el código de Javascript se ejecute de manera eficiente.

Entonces revisemos algunos aspectos importantes de Javascript que nos serán de gran utilidad al momento de crear nuestros programas, pero sobre todo, para que node.js los ejecute correctamente.

Comencemos con algunos conceptos y agreguemos a nuestro primer programa el siguiente código. Importante tener cuidado si intentas copiar – pegar este código, ya que las comillas simples pueden no ser las correctas en el editor de textos al momento de pegar.

```
//Objetos asociados a los tipos primitivos
var cadena = 'Esto es una cadena';
var cadena_obj = new String('Esto es una cadena');
console.log(cadena);
console.log(cadena_obj);

var numero = 12;
var numero_obj = new Number(12);
console.log(numero);
console.log(numero_obj);

var logico = true;
var logico_obj = new Boolean(false);
console.log(logico);
console.log(logico_obj);

//objetos compuestos
var lista = ['0', '1', '2', '3'];
var lista_obj = new Array('0', '1', '2', '3');
console.log(lista);
console.log(lista_obj);

var objeto_literal = {nombre:'Pedro', edad: 30};
var objeto_constructor = new Object();
objeto_constructor.nombre = 'Pedro';
objeto_constructor.edad = 30;
console.log(objeto_literal);
console.log(objeto_constructor);

//Objetos predefinidos del lenguaje
var ahora = new Date();
console.log(ahora);
```

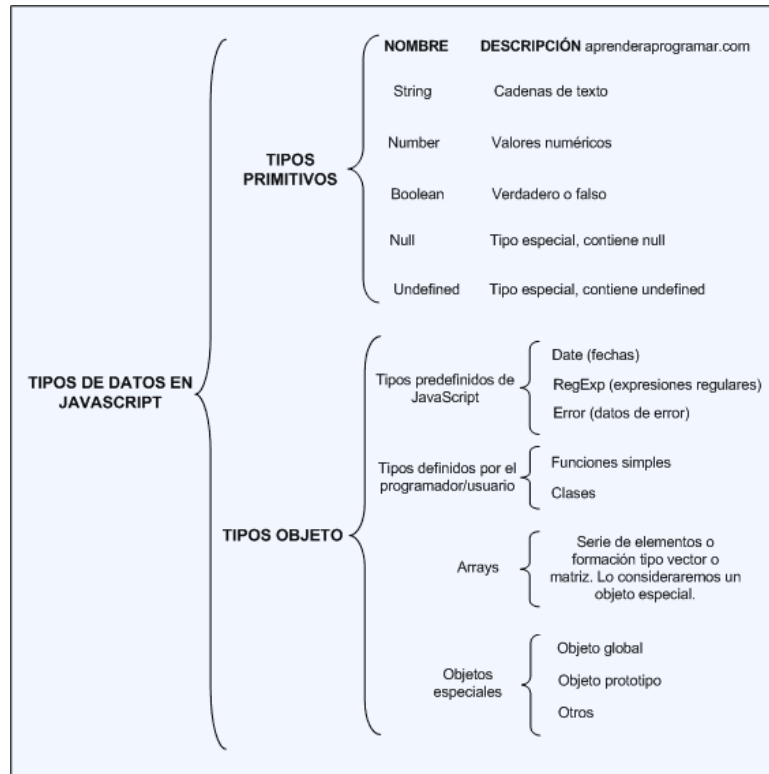



Figura 6 Tipos de datos en Javascript

Conceptos básicos sobre variables

Una variable se declara con la palabra clave **var**. Por ejemplo **var precio;** constituye la declaración de una variable denominada precio. En la declaración no figura qué tipo de variable es (por ejemplo si es texto tipo String o si es numérica tipo Number). Entonces, ¿cómo se sabe de qué tipo es una variable? JavaScript decide el tipo de la variable por inferencia. Si detecta que contiene un valor numérico, la considerará tipo Number. Si contiene un valor de tipo texto la considerará String. Si contiene true ó false la considerará booleana.

El nombre de una variable deberá estar formado por letras, números, guiones bajos o símbolos pesos (\$), no siendo admitidos otros símbolos. El nombre de la variable no puede empezar por un número: obligatoriamente ha de empezar con una letra, un signo de pesos o un guión bajo. Por tanto son nombres de variables válidos precio, \$precio, _precio_, _\$dato1, precio_articulo, etc. y no son nombres válidos 12precio ni precio# ó pre!dato1.

Una variable se inicializa cuando se establece su contenido o valor por primera vez. Por ejemplo precio = 22.55; puede ser una forma de inicializar una variable.

Una variable se puede declarar e inicializar al mismo tiempo. Por ejemplo podríamos escribir var precio = 22.55; con lo que la variable ha sido declarada e inicializada en una misma línea.

JavaScript no requiere declaración del tipo de las variables, e incluso permite que una variable almacene contenido de distintos tipos en distintos momentos. Por ejemplo podríamos usar `precio = 22.55`; y en un lugar posterior escribir `precio = 'muy caro'`; Esto, que en otros lenguajes generaría un error, es aceptado por JavaScript.

JavaScript distingue entre mayúsculas y minúsculas (no sólo para las variables): por tanto no es lo mismo `precio = 22.55` que `Precio = 22.55`. `Precio` es una variable y `precio` otra.

JavaScript permite hacer uso de una variable sin que haya sido declarada. En muchos lenguajes de programación es necesario declarar una variable antes de poder hacer uso de ella, pero JavaScript no obliga a ello. Cuando JavaScript se encuentra una variable no declarada, crea automáticamente una variable y permite su uso.

Significado de `undefined`, `null` y `NaN`

- El contenido de una variable no inicializada es `undefined`. En este caso decimos que la variable es de tipo `Undefined`.
- El contenido de una variable puede ser `null` y en ese caso decimos que la variable es de tipo `Null`.
- Si intentamos realizar una operación matemática con una variable cuyo contenido no es numérico sino texto, la variable toma el valor `NaN`. Para JavaScript `NaN` (abreviatura de “Not-a-Number”) es un valor numérico especial, que representa “número ilegal o no representable”. La asignación de `NaN` que realiza JavaScript automáticamente cuando se intentan realizar operaciones numéricas ilegales evita la aparición de errores explícitos o que el código JavaScript deje de ejecutarse.