Proyecto BID-MINEC

PROYECTO UPSKILLING & RESKILLING

Orientado a la reconversión y mejora de habilidades digitales y tecnológicas

FORMÁS
CENTRO DE FORMACIÓN AVANZADO PARA PROFESIONALES

FEDISAL
Fundación para la Educación Integral Salvadoreña

BID

TECH SKILLS COUNCIL

GOBIERNO DE EL SALVADOR

MINISTERIO DE ECONOMÍA

FrontEnd básico-intermedio con JavaScript

React

Facilitador: Iván Alvarado

Contenido

1. Recursos básicos (NodeJS)
2. React
    1. Estructura de directorios
    2. Prototypes
    3. Defaultprops
    4. Hooks
    5. useContext
    6. Redux
    7. Autenticación y autorización

1. Recursos básicos (NodeJS)

Introducción

- Node. js es un entorno en tiempo de ejecución multiplataforma, de código abierto basado en el lenguaje de programación ECMAScript, asíncrono, con I/O de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google.
- Con Node.js, se puede ejecutar el código simultáneamente tanto en el lado del cliente como en el del servidor, acelerando todo el proceso de desarrollo. Node.js cierra la brecha entre el desarrollo del front-end y el back-end y hace que el proceso de desarrollo sea mucho más eficiente.

1. Recursos básicos (NodeJS)

Algunas herramientas:

- Mocha.js
- Chai
- Keystone.js
- Kos.js
- Broccoli.js
- Sinon.js
- Express.js
- Socket.io
- Webpack
- PM2
- Electrode.io
- Meteor.js
- Babel
- Mean.js

1. Recursos básicos (NodeJS)

Webpack

- Webpack es un práctico bundler que se utiliza para simplificar el desarrollo del front-end. Detecta los módulos con dependencias y los transforma en assets estáticos que representan los módulos.
- Puedes instalar la herramienta a través del npm o del gestor de paquetes de hilos.

Strapi

- Strapi es un sistema de gestión de contenidos (CMS) sin encabezado, de código abierto. Un CMS sin encabezado es básicamente un software que te permite administrar tu contenido sin un frontend preconstruido. Es un sistema sólo de respaldo que funciona usando APIs RESTful.
- Se puede instalar Strapi a través de paquetes de Yarn o npx.

1. Recursos básicos (NodeJS)

Broccoli

- Broccoli es una poderosa herramienta de construcción que se ejecuta en un módulo ES6. Las herramientas de construcción permiten reunir todos los diferentes assets dentro de un aplicación o sitio web, por ejemplo, imágenes, CSS, JavaScript, etc., en un formato distribuible. Broccoli se marca a sí mismo como el " canal de recursos para aplicaciones ambiciosas".

Danger

- Danger es una herramienta de código abierto muy útil para agilizar las comprobaciones de las solicitudes de extracción (PR). Como dice la descripción de la biblioteca de Danger, la herramienta te ayuda a "formalizar" tu sistema de revisión de código administrando los chequeos PR. Danger se integra con tu CI y te ayuda a acelerar el proceso de revisión.
- La integración de Danger con un proyecto es un proceso sencillo que se realiza paso a paso: sólo hay que incluir el módulo Danger y crear un archivo para cada proyecto. Sin embargo, es más conveniente crear una cuenta en el mismo Danger (a través de GitHub o Bitbucket), y luego configurar los tokens de acceso.

1. Recursos básicos (NodeJS)

Snyk

- La seguridad cibernética es una de las principales preocupaciones de los desarrolladores. Snyk es una de las herramientas más conocidas para arreglar las vulnerabilidades de los componentes de código abierto. Comenzó como un proyecto para arreglar vulnerabilidades en proyectos Node.js y ha evolucionado para detectar y arreglar vulnerabilidades en aplicaciones Ruby, Java, Python y Scala también.

Migrat

- Migrat es una herramienta de migración de datos extremadamente fácil de usar que utiliza texto simple. Funciona a través de una diversa gama de pilas y procesos que lo hacen aún más conveniente.

Clinic.js

- Clinic.js es una herramienta de monitorización de código abierto para los proyectos de Node.js. Combina tres herramientas diferentes -Doctor, Bubbleprof y Flame- que ayudan a monitorizar, detectar y resolver problemas de rendimiento con Node.js.

1. Recursos básicos (NodeJS)

PM2

- La monitorización es uno de los aspectos más importantes de cualquier proceso de desarrollo de backend. PM2 es una herramienta de gestión de procesos para Node.js que ayuda a los desarrolladores a monitorear múltiples aspectos de sus proyectos, tales como registros, retrasos y velocidad. La herramienta es compatible con Linux, MacOS y Windows y soporta todas las versiones de Node.js a partir de Node.js 8.X.

Electrode

- Electrode es una plataforma de aplicaciones de código abierto de Walmart Labs. La plataforma le ayuda a construir aplicaciones universales React/Node.js a gran escala de forma estructurada.
- El generador de aplicaciones Electrode le permite construir un núcleo flexible centrado en el código, proporciona algunos módulos estupendos para añadir características complejas a la aplicación, y viene con una amplia gama de herramientas para optimizar el paquete Node.js de su aplicación.

1. Recursos básicos (NodeJS)

Dependencias

Hay varias dependencias en las que se basa Node.js para funcionar como lo hace.

- Dependencias
  - Librerías
    - V8
    - libuv
    - llhttp
    - c-ares
    - OpenSSL
    - zlib
  - Herramientas
    - npm
    - gyp
    - gtest

1. Recursos básicos (NodeJS)

Librerias

- V8

La librería V8 proporciona a Node.js un motor de JavaScript, que Node.js controla a través de la API V8 C++. V8 es mantenido por Google para su uso en Chrome.

- Libuv

Otra dependencia importante es libuv, una librería en C que se utiliza para abstraer operaciones I/O sin bloqueo en una interfaz coherente en todas las plataformas compatibles. Proporciona mecanismos para manejar sistema de archivos, DNS, red, child processes, pipes, signal handling, polling y streaming. También incluye un thread pool para descargar el trabajo de algunas cosas que no se pueden hacer de forma asíncrona a nivel del sistema operativo.

- llhttp

El análisis sintáctico de HTTP es manejado por una biblioteca ligera de TypeScript y C llamada llhttp. Está diseñada para no hacer ninguna llamada al sistema (syscalls) o asignación (allocations), por lo que tiene una cuota de memoria por solicitud muy pequeña.

1. Recursos básicos (NodeJS)

Librerias

- c-ares

Para algunas peticiones DNS asíncronas, Node.js utiliza una librería en C llamada c-ares. Se expone a través del módulo DNS en JavaScript como la familia de funciones resolve(). La función lookup(), que es la que utiliza el resto del núcleo, hace uso de las llamadas getaddrinfo(3) en libuv. La razón de esto es que c-ares soporta /etc/hosts, /etc/resolv.conf y /etc/svc.conf pero no cosas como mDNS.

- OpenSSL

OpenSSL se utiliza ampliamente en los módulos tls y crypto. Proporciona implementaciones ampliamente probadas de muchas funciones criptográficas en las que la web moderna confía para su seguridad.

- Zlib

Para una rápida compresión y descompresión, Node.js se basa en la librería zlib, un estándar de la industria conocida también por su uso en gzip y libpng. Node.js utiliza zlib para crear interfaces de compresión y descompresión síncronas, asíncronas y de streaming.

1. Recursos básicos (NodeJS)

Herramientas

- npm

Node.js se basa en la modularidad, y con ello viene la necesidad de un gestor de paquetes de calidad; para este propósito, se creó npm. Con npm viene la mayor selección de paquetes creados por la comunidad de cualquier ecosistema de programación, lo que hace que la construcción de aplicaciones Node.js sea rápida y fácil.

- gyp

El sistema de construcción (build system) es manejado por gyp, un generador de proyectos basado en python copiado de V8. Puede generar archivos de proyecto para utilizarlos con sistemas de compilación en muchas plataformas. Node.js requiere un sistema de compilación porque gran parte de él — y sus dependencias — están escritas en lenguajes que requieren compilación.

- gtest

El código nativo puede ser probado usando gtest, que está tomado de Chromium. Permite probar C/C++ sin necesidad de un ejecutable de node existente desde el que arrancar..

1. Recursos básicos (NodeJS)

Nodemon

Permite reiniciar la aplicación de manera automática. Para ello desde la terminal debe lanzarse:

```
npm i –g nodemon
```

Luego las aplicaciones deben lanzarse con nodemon.

1. Recursos básicos (NodeJS)

Lanzando un hola mundo en VSCode

- Con el mecanismo habitual crear una carpeta y un script js con el contenido:

```
console.log("Hola Mundo");
```

- Ejecutar en la terminal el script (comando node)

1. Recursos básicos (NodeJS)

Dependencias de un proyecto de node.js

- Se requiere en el proyecto un archivo llamado package.json (Este contiene la configuración del proyecto).
- El archivo package.json, se puede construir de manera automática con el comando npm

  npm init

- Crear el script punto de entrada de la app

index.js

```
2  var server = http.createServer();
3  function mensaje(rq, rp){
4      rp.writeHead(200, {'content-type': 'text/plain'});
5      rp.write("Hola Mundo");
6      rp.end();
7  }
8  server.on('request', mensaje);
9  server.listen(3000, function(){
10     console.log("app S1 Formas funcionando");
11 });
```

1. Recursos básicos (NodeJS)

¿Qué es un Módulo en Node.js?

- Considere que los módulos son lo mismo que las bibliotecas de JavaScript. Un conjunto de funciones que desea incluir en su aplicación.

Módulos incorporados

- Node.js tiene un conjunto de módulos integrados que puede usar sin más instalación.

¿Cómo se incluyen?

- Para incluir un módulo, se utiliza la función require(), indicándole como parámetro el nombre del módulo.

1. Recursos básicos (NodeJS)

Creando módulos

index.js | modulo.js

modulo.js > ...

```
1  exports.MyModulo = function(){
2      return "Este es el mensaje de mi modulo";
3  };
```

index.js | modulo.js

index.js > mensaje

```
1  var md = require('./modulo');
2
3  var http = require('http');
4  var server = http.createServer();
5  function mensaje(rq, rp){
6      rp.writeHead(200, {'content-type': 'text/plain'});
7      rp.write("Hola Mundo " + md.MyModulo());
8      rp.end();
9  }
10 server.on('request', mensaje);
11 server.listen(3000, function(){
12     console.log("app S1 Formas funcionando");
13 });
```

1. Recursos básicos (NodeJS)

1	Módulo	Descripción
1	assert	Proporciona un conjunto de pruebas de aserción
2	buffer	Para controlar datos binarios
3	child_process	Para ejecutar un proceso secundario
4	cluster	Para dividir un único proceso de nodo en varios procesos
5	crypto	Para controlar las funciones criptográficas de OpenSSL
6	dgram	Proporciona la implementación de sockets de datagramas UDP
7	dns	Para realizar búsquedas DNS y funciones de resolución de nombres
8	domain	Obsolescente. Para controlar los errores no controlados
9	events	Para controlar eventos
10	fs	Para manejar el sistema de archivos
11	http	Para hacer que Node.js actúe como un servidor HTTP
12	https	Para hacer que Node.js actúe como un servidor HTTPS

1. Recursos básicos (NodeJS)

1	Módulo	Descripción
1	net	Para crear servidores y clientes
2	os	Proporciona información sobre el sistema operativo
3	path	Para controlar las rutas de acceso de los archivos
4	punycode	Obsolescente. Un esquema de codificación de caracteres
5	querystring	Para controlar las cadenas de consulta de URL
6	readline	Para manejar flujos legibles una línea a la vez
7	stream	Para controlar los datos de streaming
8	string_decoder	Para decodificar objetos de búfer en cadenas
9	timers	Para ejecutar una función después de un número determinado de milisegundos
10	tls	Para implementar protocolos TLS y SSL
11	url	Para analizar cadenas de URL
12	vm	Para compilar código JavaScript en una máquina virtual

1. Recursos básicos (NodeJS)

Eventos

- Muchos objetos en Node emiten eventos: un net.Server emite un evento cada vez que se establece una conexión, un fs.readStream emite un evento cuando se abre un fichero. Todos los objetos que emiten eventos son instancias de events.EventEmitter. Puedes usar este módulo haciendo require("events");
- Normalmente, los nombres de los eventos siguen la notación camel-case, sin embargo, no hay ninguna restricción en este aspecto y se aceptará cualquier cadena.
- Se pueden adjuntar funciones a objetos, para que sean ejecutadas cuando se emita un evento. Estas funciones reciben el nombre de *oyentes* .

1. Recursos básicos (NodeJS)

events.EventEmitter

- Para usar la clase EventEmitter, haz de importarla haciendo require('events').EventEmiter.
- Cuando una instancia de la clase EventEmitter se encuentra con un error, la acción típica es emitir un evento de error. Los eventos de error son tratados como un caso especial en nodo. Si no tiene un oyente asociado la acción por defecto será imprimir la traza de la pila y salir del programa
- Todos los EventEmitters emiten el evento 'newListener'cuando se añaden nuevos oyentes.

```
var events = require('events');
var emitter = new events.EventEmitter();
emitter.on("algo", function(a1, a2){
    console.log("Oyente1", a1, a2);
});
emitter.on("algo", function(a1, a2){
    console.log("Oyente2", a1, a2);
});
emitter.emit("algo", 'algo1', 'algo2');
```