Browserify (práctica)

Tiempo estimado: 20min

El objeto de esta práctica es afianzar, reforzar y consolidar los conocimientos teóricos presentados en la lección.

Al finalizarla, el estudiante:

- Habrá instalado Browserify.
- Habrá realizado empaquetados.
- Habrá excluido archivos del archivo empaquetado.

Instalación de Node.js

Lo primero es instalar Node, la plataforma que utilizaremos para ejecutar las herramientas que nos ayudarán a realizar nuestro trabajo más fácilmente. Para ello, ir a su sitio web oficial, nodejs.org, e instalar la última versión. También se puede instalar mediante los administradores de paquetes Chocolatery y APT.

Instalación de Browserify

A continuación, vamos a instalar Browserify en nuestra máquina:

1. Abrir una consola.
2. Instalar browserify globalmente mediante NPM, instalado automáticamente con Node:

   ```
   > npm install -g browserify
   ```
3. Comprobar que hay acceso al comando browserify:

   ```
   > browserify --version
   ```
4. Mostrar las opciones básicas del comando:

   ```
   > browserify --help
   ```
5. Mostrar las opciones avanzadas del comando:

   ```
   > browserify --help advanced
   ```

Empaquetado

Una vez instalado Browserify, lo siguiente es generar un archivo empaquetado para mostrar su utilización. Browserify *no* está limitado a aplicaciones de navegador, también se puede utilizar en aplicaciones que se ejecutarán con el motor JavaScript de Node. Por el carácter del curso, nos centraremos en aplicaciones clientes webs. Pero no encontrará ningún problema si necesita extrapolar lo aprendido a lo largo del curso a aplicaciones Node.

El objeto es ver cómo generar un archivo empaquetado que usa el módulo crypto de Node en el navegador.

1. Ir a la consola.
2. Crear el directorio de la práctica e ir a él.
3. Crear el archivo crypto.js, mediante su IDE favorito como, por ejemplo, Atom de GitHub o Nuclide de Facebook:

   ```
   > atom crypto.js
   ```
   Contenido:

   ```
   //imports
   const crypto = require("crypto");

   //api
   ```



```
module.exports.sha = sha;

//sha(data) : hash
function sha(data) {
  const hash = crypto.createHash("sha256");

  hash.write(data);
  return hash.digest("hex");
}

//main
if (require.main === module) {
  console.log(sha("buongiorno"));
}
```

Observe cómo se expone la función sha como API del módulo crypto.js:

```
//api
module.exports.sha = sha;
```

4. Ir a la consola.
5. Comprobar el funcionamiento del módulo mediante Node.js:

```
> node crypto.js
e0cb29405bc784372fc6e9f94f8f30be119da6d292d672cebe34c334ea989a79
>
```

Su ejecución debe mostrar la huella SHA-256 del texto buongiorno.

6. Generar el archivo empaquetado para su uso en el navegador:

```
> browserify crypto.js --standalone mycrypto --outfile bundle.js
```

El módulo lo empaquetamos en el archivo bundle.js y le asignamos el nombre mycrypto. Esto nos permitirá el acceso, en el navegador, a la función sha mediante la variable global mycrypto, tal como veremos más adelante.

7. Consultar el número de líneas que tiene el archivo crypto.js.

   En Windows:

```
> (cat -Raw crypto.js | Measure-Object -Line).Lines
18
>
```

   En Linux:

```
$ wc -l crypto.js
18 crypto.js
$
```

8. Consultar el número de líneas que tiene el archivo bundle.js:

   Windows:

```
> (cat -Raw bundle.js | Measure-Object -Line).Lines
21007
>
```

   Linux:

```
$ wc -l bundle.js
21007 bundle.js
$
```

   La diferencia en el número de líneas es enorme. Esto se debe a que Browserify ha añadido automáticamente el código de su implementación particular del módulo crypto, permitiéndonos así su utilización en el navegador.

   El número de línea dependerá de la versión de Browserify.

9. Echar un vistazo al archivo bundle.js.
10. Crear el archivo index.html:

```
<!doctype html>

<html>
<head>
  <meta charset="utf-8">
```

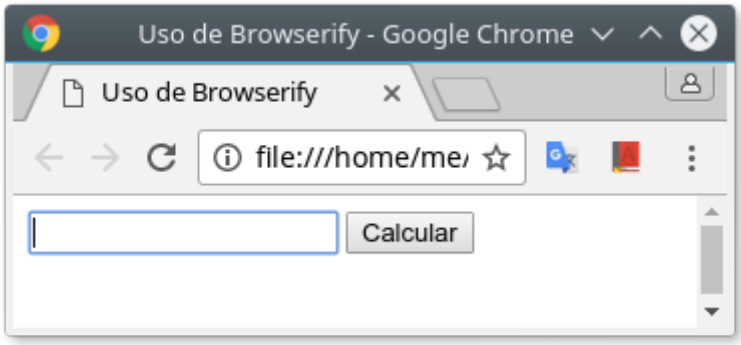


```
  <title>Uso de Browserify</title>

  <script type="text/javascript" src="./bundle.js"></script>
  <script type="text/javascript">
    function sha() {
      var txt = document.querySelector("#text").value;

      if (txt) {
        document.querySelector("#hash").innerHTML = mycrypto.sha(txt);
      } else {
        alert("Por favor, indique un texto.");
      }
    }
  </script>
</head>

<body>
  <input id="text" type="text" title="Texto a pasar a la función de huella." autofocus>
  <button onclick="sha()">Calcular</button>

  <pre id="hash">
  </pre>
</body>
</html>
```

Observe cómo se accede a la función sha, mediante la variable global mycrypto, cuyo nombre se indica mediante la opción --standalone, la cual crea automáticamente Browserify en el archivo empaquetado:

```
mycrypto.sha(txt)
```

11. Abrir el archivo index.html en el navegador:



12. Escribir el texto buongiorno y hacer clic en Calcular para obtener su huella SHA-256:

Éste es un ejemplo muy sencillo que permite ilustrar cómo utilizar algunos paquetes de Node en el navegador. Una de las utilidades más interesantes de Browserify. Si lo desea, puede utilizar, por ejemplo, los módulos http y request de Node en el navegador para acceder a recursos webs, aunque es preferible el uso de la API Fetch.

Empaquetado de varios archivos

Ahora, vamos a empaquetar un módulo personalizado con varios archivos. Vamos a jugar con el mismo módulo y la misma idea. Vamos a recrear nuestro módulo mediante dos archivos y ver cómo hay que comunicárselos a Browserify cuando genere el archivo empaquetado.

1. Modificar el archivo crypto.js.

```
//imports
const sha = require("./sha.js");

//api
```



```
module.exports.sha = sha;

//main
if (require.main === module) {
  console.log(sha("buongiorno"));
}
```

Ahora, no hay nada en este archivo de la función de huella. Lo delegamos al archivo sha.js.

2. Crear el archivo sha.js que expone la función de huella SHA:

```
//imports
const crypto = require("crypto");

//api
module.exports = sha;

//sha(data) : hash
function sha(data) {
  const hash = crypto.createHash("sha256");

  hash.write(data);
  return hash.digest("hex");
}
```

3. Ir a la consola.
4. Comprobar el funcionamiento del módulo mediante Node:

```
> node crypto.js
e0cb29405bc784372fc6e9f94f8f30be119da6d292d672cebe34c334ea989a79
>
```

5. Generar el archivo empaquetado:

```
> browserify crypto.js --standalone mycrypto --outfile bundle.js
```

Como puede observar, no hace falta indicar los dos archivos del módulo. Sólo el archivo principal, recordemos, conocido como punto de entrada. Browserify recorre el archivo de entrada, detecta la importación del archivo sha.js y lo añade al grafo de dependencias y, a partir de ahí, al archivo empaquetado.

6. Ir al navegador y refrescar la página para que se cargue el nuevo archivo empaquetado, F5.
7. Indicar el texto buonasera y hacer clic en Calcular:

Exclusión de archivos

A continuación, vamos a ver cómo excluir módulos. Concretamente, vamos a indicarle a Browserify que no añada su implementación del módulo crypto:

1. Ir a la consola.
2. Generar el archivo empaquetado sin añadir el módulo crypto:

```
> browserify crypto.js --standalone mycrypto --outfile bundle.js --exclude crypto
```

Cuando *no* se desea añadir ningún módulo predefinido (*built-in module*), recordemos, aquellos módulos de Node para los que Browserify tiene una implementación particular para su uso en el navegador, la manera más correcta de excluir su añadidura al empaquetado es mediante la opción --no-builtins, no así mediante --exclude.

3. Consultar el número de líneas del archivo empaquetado:

En Windows:

```
> (cat -Raw bundle.js | Measure-Object -Line).Lines
28
>
```

En Linux:

```
$ wc -l bundle.js
```



```
28 bundle.js
$
```

Observe que el número de líneas se ha reducido considerablemente. Ha pasado de más de 20000 a no llega a treinta.

4. Echar un vistazo al archivo empaquetado.

   El contenido de los dos archivos se encuentra en él, no así el del módulo crypto de Node.

5. Ir al navegador y refrescar la página para que se cargue el nuevo archivo empaquetado.
6. Indicar el texto buonasera y hacer clic en Calcular.

   Ahora se producirá un error, que se puede observar mediante la consola del navegador, debido a que el empaquetado no contiene la implementación del módulo predefinido crypto.

