Doc-4-JS

Lascario Pacheco, para el CEDAURU

Es una aplicación web que permite la generación de documentación estilo Application Programming Interface (API), con base en cualquier documento de texto (esencialmente scripts de Javascript), que contengan las anotaciones propias de la herramienta, Doc-4-JS.

Doc-4-JS está pensado para facilitar al desarrollador la tarea de documentar sus librerías, de forma que sea más fácil alcanzar la divulgación de las funcionalidades de la misma. No obstante, la aplicación también puede ser utilizada por cualquier amante de la programación y la investigación como medio didáctico de aprendizaje (la herramienta es de código abierto), así como también está abierta a las sugerencias de desarrolladores más experimentados que permitan aumentar las funcionalidades de Doc-4-JS.

1 Anotaciones en Doc-4-JS

Las anotaciones permiten a la aplicación reconocer los metadatos de cualquier segmento del código sometido a procesamiento por el usuario. Doc-4-JS cuenta con un repertorio sólido de anotaciones que cubren las necesidades genéricas de documentación para cualquier librería en Javascript. A continuación se presenta un listado de las anotaciones soportadas por Doc-4-JS en su versión de lanzamiento.

Las anotaciones pueden ser clasificadas de acuerdo con la sección de la librería que pretenden documentar. Esencialmente, un archivo de texto Javascript que describe una librería puede dividirse en cuatro segmentos: cabecera, métodos, variables internas y observaciones.

1.1 Anotaciones de cabecera

Anotación	Significado	Observaciones
@head	Marca el inicio de un comentario de cabecera de documentación	
@author {name}	Hace referencia al autor del archivo	Sólo se presenta una vez por cabecera. Por

		supuesto, no debería haber más de una cabecera por archivo.
@date {date}	Hace referencia a la fecha de creación del archivo o método	Sólo se presenta una vez por cabecera.
@desc {description}	Hace referencia a la descripción del archivo o método	Se presenta cada vez que se hace necesario explicar el objetivo de algún método o del objeto como tal.
@name {name}	Hace referencia al nombre del archivo u objeto que se desea documentar	Se estila que aparezca una sola vez en el archivo, pero de aparecer en más de una ocasión, será omitida en las oportunidades diferentes a la inicial.

1.1.1 Ejemplo de una documentación de cabecera en Doc-4-JS

```
/**
* @author CEDAURU
* @date 15/DIC/2014
* @name Queue.js
* @desc A javascript object that behaves like Queue object, from Java
*/
```

1.2 Anotaciones de métodos

Anotación	Significado	Observaciones
@body	Marca el inicio de un comentario de cuerpo de documentación	
@desc {description}	Hace referencia a la	Se presenta cada vez

	descripción del archivo o método	que se hace necesario explicar el objetivo de algún método o del objeto como tal
@param {type} {name} {description}	Hace referencia a los parámetros de un método en particular	Se espera su presencia en las anotaciones asociadas a un método, ya que los parámetros de los métodos no son documentados automáticamente por la herramienta.
@returns {type} {description}	Hace referencia al tipo del valor de retorno ofrecido por el método	Se recomienda su inclusión una sola vez por método, aunque Javascript lo permita.

1.2.1 Ejemplo de una documentación de método en Doc-4-JS

```
/**
* @desc A function that allows adding a couple of numbers
* @param {Integer | Double} a It is the first term of the sum
* @param {Integer | Double} b It is the second term of the sum
* @returns {Integer | Double} It is the result of the sum of both terms
*/
function add(a, b) { /*Instructions*/}
```

1.3 Anotaciones de variables internas

Anotación	Significado	Observaciones
@body	Marca el inicio de un comentario de cuerpo de documentación	
@var {type} {name}	Hace referencia a la	Se presenta cada vez

{description}	descripción del archivo o método	que se hace necesario explicar el objetivo de algún método o del objeto como tal

1.3.1 Ejemplo de una anotación de variables internas en Doc-4-JS

```
/**
* @var {String} s It is a string that serves no purpose
*/
var s = “I am a string.”;
```

1.4 Anotaciones de observaciones

Anotación	Significado	Observaciones
@foot	Marca el inicio de un comentario de pie de documentación	
@changelog {version}	Hace referencia al inicio de los cambios de versión en la sección de observaciones	Se utiliza una vez por sección de observación.
@change {date} {description}	Hace referencia a un cambio en particular para la sección de cambios de versión	Puede aparecer en múltiples ocasiones en la sección de cambios de versión.
@depend {name} {version}	Hace referencia a una dependencia de la librería	Puede aparecer en múltiples ocasiones en la sección de cambios de versión, aunque se recomienda que la librería a desarrollar tenga pocas dependencias.

1.4.1 Ejemplo de una documentación de observaciones en Doc-4-JS

```
/**
* @changelog {1.0}
* @change 15/DIC/2014 Added substracting functionality to the application
* @change 18/DIC/2014 Added multiplying functionality to the application
* @depend jQuery-2.1.0
* @depend Bootstrap-3.3.0
*/
```

2 Funcionamiento de Doc-4-JS

Hasta ahora se ha mencionado que Doc-4-JS permite generar la documentación de las librerías u objetos Javascript que los desarrolladores programen, pero aún no se ha explicado la forma en la que Doc-4-JS admite las anotaciones que contienen los metadatos sobre el archivo o el método que se desea documentar. A continuación se explica, paso a paso, la forma en la que Doc-4-JS alcanza el objetivo de generación de documentación.

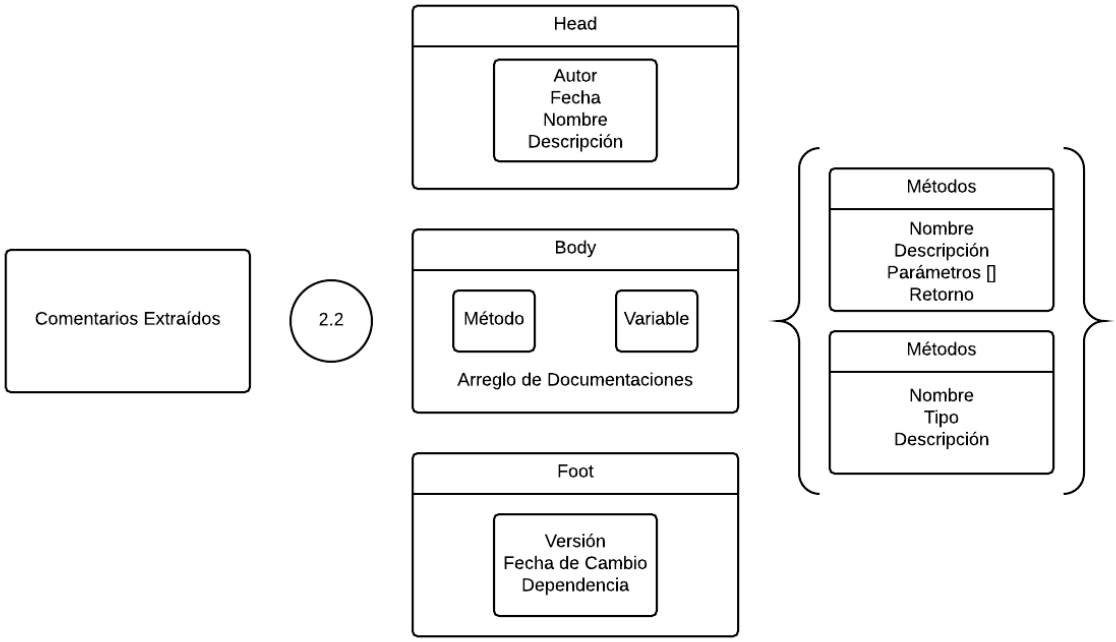
2.1 Extracción de comentarios y nombres de funciones

Los comentarios y nombres de funciones son ubicados y "recortados" del código Javascript original, y son almacenados temporalmente por la herramienta. Este motor sólo busca comentarios en los formatos "/**/" y "//", soportados por Javascript. También cabe mencionar que sólo busca nombres de métodos en los formatos "nombredemetodo = function () …", "function nombredemetodo () …" y "this.nombredemetodo = function () …", generalmente empleados en la programación con Javascript.

2.2 Lectura de comentarios

El motor de Doc-4-JS lee todos los comentarios extraídos del archivo que se le ha entregado, en busca de anotaciones. Es en este momento cuando se consiguen y extraen las anotaciones y sus metadatos asociados del código, en crudo.

En general, se construyen tres estructuras de datos que conforman las tres partes de cualquier documentación en Javascript: *head* (cabecera), *body* (cuerpo de la documentación) y *footer* (pie de la documentación). Todas las estructuras pueden entenderse como un arreglo de estructuras de un solo elemento, con excepción de la estructura body, que puede contener más de un elemento en su arreglo.



Con esto en mente, se procede a realizar el siguiente paso en el algoritmo de generación de documentación de Doc-4-JS.

2.3 Validación de anotaciones

Una vez que se tienen las anotaciones y los metadatos en crudo, se procede a realizar su validación, se evalúa si la anotación a la que se hace referencia (aquella que viene después del símbolo @) existe en el arreglo de símbolos de la herramienta, las no existentes son ignoradas a partir de este punto. Después de esto se eliminan los espacios al inicio y final de los datos contenidos en las anotaciones válidas, si existen.

Es en este punto, también, cuando se evalúa la existencia de metadatos para los métodos leídos en el paso 1; si no se ha introducido información sobre los mismos, son ignorados a partir de este punto en la ejecución.

2.4 Generación de la estructura de datos

Con los datos extraídos y filtrados se procede a generar la estructura de datos que contiene toda la información anotada por el programador en el código original. Básicamente se genera un conjunto de JSONs que contiene el nombre del metadato y el metadato como tal; esta es la base de la página web generada al final del proceso.

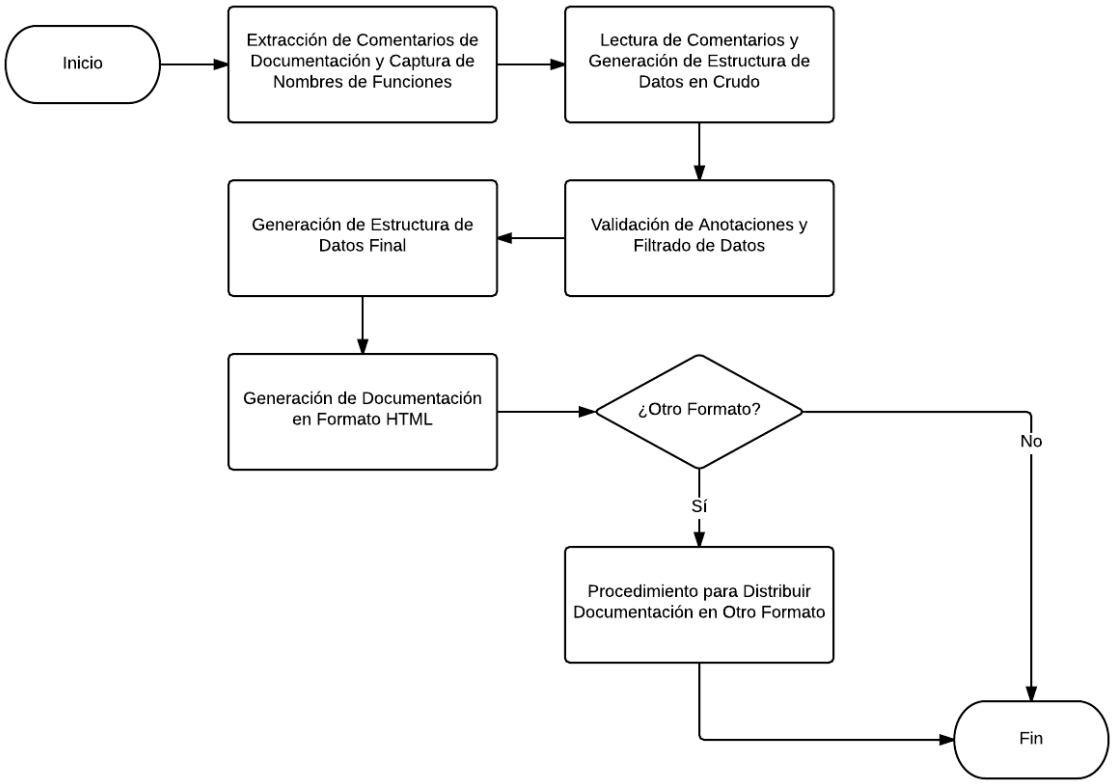
2.5 Generación de la documentación en formato HTML

En este punto se construye sobre la plantilla HTML diseñada para la aplicación, rellenando los campos con los datos que existen en la estructura de datos generada en el paso anterior. Este procedimiento es meramente mecánico, y con poca lógica de funcionamiento.

2.6 Más opciones de presentación

Adicionalmente, se ofrece al usuario final la posibilidad de descargar el código HTML con la documentación generada por la aplicación, cambiar los estilos de despliegue de la documentación con CSS o bien descargarla sin estilo alguno, así como también se ofrece la misma en formato PDF, para mayor conveniencia y versatilidad.

2.7 Diagrama de flujo del funcionamiento de Doc-4-JS



3 Fases de Desarrollo de Doc-4-JS

3.1 Fase de diseño

Durante la fase de diseño se describen, definen y estructuran los problemas que se desean resolver. Consecuentemente, esta etapa abre las puertas a las demás fases de desarrollo del proyecto, razón por la cual se recomienda pasar por esta fase una sola vez durante el desarrollo de la aplicación.

3.1.1 Objetivo de la fase de diseño

Desarrollar en un documento las características del problema que se desea resolver y el planteamiento de una o varias posibles soluciones para el problema objeto.

3.1.2 Actividades de la fase de diseño

a. Describir el problema que se tiene presente.

b. Analizar el problema en busca de posibles soluciones que permitan resolverlo o mitigar sus efectos.
c. Definir el funcionamiento de la solución (aplicación) que se ha escogido como solución óptima a desarrollar.
d. Redactar, en un documento, todas las características de la solución que se ha diseñado con base en el análisis de problema que se realizó previamente.

3.2 Fase de programación

3.2.1 Actividades de la fase de programación

a. Dividir la solución del problema en módulos (objetos), que permitan trabajar de forma organizada y hacer un seguimiento del progreso del desarrollo.
b. Repartir la carga del trabajo de programación entre los miembros del equipo de desarrollo.
c. Programar los diferentes módulos de la aplicación.
d. Integrar los módulos programados en una aplicación funcional que se comporte como se ha descrito en el documento entregable, producto de la fase de diseño.

3.3 Fase de pruebas

3.2.1 Actividades de la fase de programación

a. Seleccionar las pruebas que deben ser practicadas en la aplicación, en busca de posibles fallos de estabilidad, desempeño e integridad.
b. Ajustar las pruebas seleccionadas a las características de la aplicación desarrollada en particular.
c. Realizar las pruebas y evaluar los parámetros seleccionados para observación; si se vuelve necesario, volver al paso C de la fase de programación. Esta acción puede repetirse cuantas veces sea necesario en el desarrollo.

3.4 Fase de publicación

3.4.1 Actividades de la fase de publicación

a. Hacer entrega de la aplicación a la autoridad respectiva para que dé su opinión acerca de los resultados del desarrollo del proyecto.

4 Enviar sugerencias o comentarios acerca del bosquejo de Doc-4-JS

Para enviar sugerencias o comentarios acerca del proyecto Doc-4-JS, por favor escribir un correo electrónico a la siguiente dirección: lascariopacheco@hotmail.com; también puedes comunicarte directamente con Lascario Pacheco, si lo conoces en persona, para manifestarle explícitamente la cantidad de detalles que se saltó explicando el funcionamiento de Doc-4-JS, así como también enviarle sugerencias acerca de cómo mejorar el funcionamiento tentativo de Doc-4-JS.

"¡Hasta la próxima!"

Lascario Pacheco