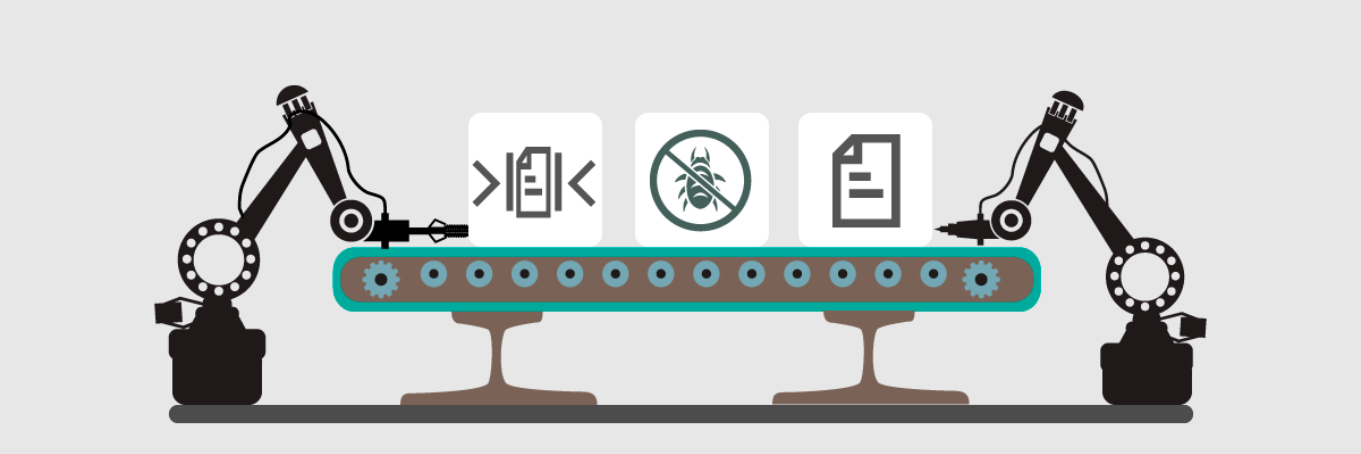
Lección 1: Task Runners (Automatización de tareas)

Lectura: Introducción a la automatización de tarea

La Automatización de Tareas



¿Qué es?

La automatización nos permite ejecutar tareas que por lo general están escritas como plugins. Veamos cuáles podemos ejecutar y cómo llamarlas.

Las siguientes son algunas de las tareas más usadas en el desarrollo en Javascript. Úsalas como inspiración para integrarlas a los proyectos en los que creas que puedan ser beneficiosas pero recuerda: siempre debes pensar si implementar una nueva tarea facilitará tu flujo de trabajo o, por el contrario, lo complica. ¿Por qué? A veces podemos tener infraestructuras complejas de tareas que en lugar de reducirnos el trabajo lo incrementan.

Tareas más comunes de automatización

1. Minify

1. Minify

Consiste en comprimir código, ya sea js - css - html o cualquier otro archivo, en un archivo de lógica, estilo y estructura equivalente que pesa menos, eliminando espacios innecesarios, reduciendo los nombres de variables a pocas letras y eliminando comentarios.

2. Linting

Se refiere a mostrar posibles bugs del código sin ejecutarlo y advertencias o errores que se pueden agregar, para ayudarnos a seguir guías de estilo o evitar patrones de código que indican malas prácticas.

3. Transpiling

Es el proceso de convertir un lenguaje interpretado a otro, es equivalente a compilar.
En este caso el lenguaje objetivo será Javascript pero podemos partir desde otras versiones de Javascript (ES6), código en otros lenguajes ya existentes (Java, Scala, Ruby, Python, C/C++) o lenguajes creados exclusivamente para ser transpilados a js (Coffeescript o Typescript).

4. Licensing and Versioning

Podemos aprovechar cada vez que construimos nuestro proyecto para escribir en el compilado final (build) la licencia de nuestro código, información de autoría y versiones.

5. Sourcemaps

Si hacemos procesos de minificacion o transpiling podemos generar mapas de código que nos permitan ver cómo se transformó nuestro código y utilizar esta información para debugear el código.

6. Rename, delete, create y map functions through files

También es muy fácil ejecutar tareas propias de los sistemas de archivos, reemplazar, cortar pedazos, renombrar archivos, borrarlos, crear nuevos archivos copiando anteriores o procesar archivos con funciones y luego guardarlos nuevamente con otros nombres. Todas estas tareas son sencillas y muy poderosas.

7. Documentación

Existen múltiples herramientas que nos permiten crear la documentación en el momento de compilación para tenerla siempre al día.
Usos más avanzados nos permiten, por ejemplo, generar la docu
desplegando a producción, actualizar el sitio estático que mantie
versión de la documentación y además, medir cómo es el cubrim

versión de la documentación y, además, medir cómo es el cubrimiento de la documentación de la base de código, a medida que crece.

Lección 1: Task Runners (Automatización de tareas)

8. Ejecución y cubrimiento de pruebas

Así como podemos generar la documentación cada vez que construimos el proyecto, también podemos correr las pruebas unitarias que hayamos creado y ver el cubrimiento, así como podemos detener la carga del código a la producción si éstas fallan y emitir alarmas a los desarrolladores.

9. Despliegue automático de integración continua

La integración continua es sencilla de implementar si utilizamos algunas tareas que ejecuten las herramientas respectivas. La rama 'production' de tu repositorio puede ser probada (tests unitarios) y luego desplegada así: si todo sale bien, todos los días en la madrugada, justo cuando el tráfico en el sitio es el mínimo, y si algo sale mal, puedes configurar emails o alertas automáticas a los desarrolladores.

10. Uso automático de repositorios

Imagina configurar útiles metodologías de trabajo como por ejemplo: que cada vez que tu código compile y estén todas las pruebas bien, automáticamente se actualice tu rama de desarrollo de git, o con un sólo comando combine la rama de producción, active hooks de git en los servidores relevantes y despliegue el código.

11. Servir aplicaciones

Existen también múltiples herramientas que fácilmente nos dan servidores de desarrollo con características como sockets, o que incluso pueden emular bases de datos de mockup o ambientes particulares.

12. Descargar URL's

Podemos escuchar URLs específicas donde se desplieguen los recursos y descargarlos para ponerlos en nuestra versión actual de la aplicación.

13. Watchers

Nos permiten observar continuamente archivos y actuar ante el cambio en estos. Entonces, si guardamos el archivo en el editor o un archivo en particular, por ejemplo, podemos aprovechar para hacer un git pull - merge - push y servir de nuevo la versión de desarrollo.

14. Livereload

Existen plugins que también nos permiten empujar nuevas versiones de la aplicación, a clientes que ya están ejecutándose, usando sockets de manera automática.

Lección 1: Task Runners (Automatización de tareas)

15. Procesamiento de imágenes

Muchas tareas como descargar, generar thumbnails, cortar o procesar imágenes para añadir marcas de agua o efectos, pueden ser ejecutadas en tiempo de compilación de las tareas.

16. Servir y subir archivos vía FTP / SSH

También podemos crear servidores FTP directamente desde nuestras tareas o pedir y enviar archivos a un servidor vía SSH, lo que nos permite desplegar nuestra aplicación directamente en nuestras tareas.

17. Renderizar componentes web en el servidor

Podemos utilizar emulación del browser en el lado del servidor para que si nuestros clientes no pueden ejecutar el Javascript de nuestras páginas les sirvamos una versión ya renderizada en html. Estas versiones son generadas por las tareas con herramientas como phantomjs.

18. Construir múltiples versiones de nuestras aplicaciones