Universidad de La Laguna

Tecnologías Web: Cliente

Herramientas.

Herramientas para las tareas de desarrollo

- Utilidades para acelerar el trabajo, disminuir los errores, optimizar el código.
  - Generación del proyecto
  - Control de las dependencias
  - Automatización de tareas
  - Empaquetado de código
  - Garantizar la calidad del código
  - Realización de test

Herramientas para las tareas de desarrollo

- Editor de texto
- Navegador
- Generadores de proyectos: Yeoman
- Gestión de paquetes: npm, brower
    - Gestión de los paquetes en el lado del cliente
- Control de versiones: Git
- Automatización de tareas Gulp, Grunt
- Empaquetadores: Webpack, Parcel, Vite, ...
- Linters y formateadores de código: ESLint, Prettier, ...
- Transpiladores de código

Yeoman

- Generación rápida del proyecto:
  - Crear la estructura del proyecto
- Instalación:

  `npm install -g yo`
- La instalación admite paquetes que agregan otras herramientas necesarias para el proyecto
- Agregar generadores disponibles:

```
npm install -g yo //instalar yeoman
npm install -g generator-webapp       // Instalar el generador
mkdir /home/tuUsuario/nuevoProyecto  // Crear la carpeta del proyecto
cd /home/tuUsuario/nuevoProyecto      // Entrar en la carpeta proyecto
yo webapp                             // Ejecutar el generador webapp
```

Gestionar dependencias

- npm: gestión de dependencias en node, se ha extendido su uso en el frontend

  `npm init` //en el directorio del proyecto

  Crea el archivo `package.json`

- bower: gestión de dependencias en el lado del cliente.
  - Creado por twitter
  - Buscar paquetes en: https://bower.io/search/
  - Crea el archivo `bower.json`

```
npm install -g bower
cd mi_proyecto
bower init
bower install <package> --save
```

Tareas repetitivas

- Pre-procesar y minificar CSS
- Unificar y pre-procesar javascript
- Mover ficheros minificados a un directorio
- Preparar pruebas de ejecución y errores
- Recarga automática del navegador
- Optimizar imágenes para la web

Recursos a minificar

- CSS
- JavaScript
- Optimizar imágenes

Gulp

- Herramienta de automatización de tareas repetitivas y/o comunes en el desarrollo del proyecto.
- Gulp convierte los archivos de entrada en flujos de datos en memoria, se minimizan las operaciones de escritura en disco.
- Automatizadores están disponibles para Gulp en forma de plugins
- Se pueden vigilar en tiempo real los archivos que queramos de nuestro proyecto, y ejecutar ciertas tareas de Gulp cada vez que detecte cambios en ellos.
- Todas las tareas que queremos automatizar se programan de forma sencilla en un script que se aloja en la raíz del proyecto:
    - `gulpfile.js`

Gulp

- Instalación: **`npm install --global gulp-cli`**
- Agregar gulp al proyecto: **`npm install gulp –save-dev`**
- Fichero gulpfiles.js básico:

```
var gulp = require('gulp');
// Tareas por defecto
gulp.task('default', function() {
    // tareas que queremos que se ejecuten cada vez que hagamos gulp
});
```

- **`gulp.src().pipe( gulp.dest())`** Crea un flujo de la fuente al destino.

```
var gulp = require('gulp');
var concatCss = require('gulp-concat-css'); //paquete a utilizar

gulp.task(“default”, function() { //tarea por defecto
gulp.src('./src/html/*.*').pipe(gulp.dest('dist'));
gulp.src('./src/css/*.*').pipe(concatCss("super.css")).pipe(gulp.dest('dist'));
gulp.src("./src/js/*.*").pipe(gulp.dest("dist"));
});
```

BrowerSyn: browersync.io

- Permite lanzar diferentes navegadores e ir verificando cambios en proyecto en tiempo real.
- Podemos especificar a qué cambios responderá la recarga
- Podemos automatizar
- Instalación: `npm install -g browser-sync`
- Inicializar : `browser-sync start --server --files "css/*.css"`

```
var gulp = require('gulp');
var browserSync = require('browser-sync').create(); //paquete a utilizar

gulp.task('browser-sync', function() {
    browserSync.init({
        server: {
            baseDir: "./"
        }
    });
});
gulp.watch("*.html").on("change", browerSyc.reload);
```

Minificar:

- Permite eliminar caracteres superfluos del archivo
- Paquete para css: **gulp-minify-css**
- Instalación **npm install --save-dev gulp-minify-css**

```
var gulp = require('gulp');
var browserSync = require('browser-sync').create();
var minifyCss = require('gulp-minify-css');

gulp.task('styles', function(){
     gulp.src(['src/styles/**/*.css'])
     .pipe(minifyCss())
     .pipe(gulp.dest.('dist/styles'))
     .pipe(browserSync.stream());
});


gulp.task('default', function(){
     browserSync.init({
     server:'./'});
gulp.watch('*.html', browserSync.reload);
});
```

Sourcesmaps

- Sourcesmaps nos permiten ver ficheros minificados en el navegador como fueron creados originalmente
- Paquete: **`gulp-sourcesmaps`**
- Instalar: **`npm install --save-dev gulp-sourcesmaps`**

```
var gulp = require('gulp');
var browserSync = require('browser-sync').create();
var minifyCss = require('gulp-minify-css);
var sourcesmaps = require(‘gulp-sourcesmaps’)

gulp.task('styles', function(){
     gulp.src(['src/styles/**/*.css']).
     .pipe(sourcesmaps.init())
     .pipe(minifyCss())
     .pipe(sourcesmaps.write())
     .pipe(gulp.dest.('dist/styles'))
     .pipe(browserSync.stream());
});


gulp.task('default', function(){
     browserSync.init({
     server:'./'});
```

- **Empaquetadores**:
  - Modularización del código en proyectos web complejos.
    - Múltiples archivos: múltiples peticiones al servidor.
    - Carga de código que no se usa
  - Los empaquetadores generan un único archivo con todas las dependencias.
  - No sólo javascript
  - Realizan tareas de optimización
  - Mejoran la estructura del código
  - Aumentan el rendimiento
  - Agilizan los flujos de trabajo de desarrollo.

- Flujo del empaquetado:
  - Especificar el punto de entrada: uno o varios archivos principales
  - Resolución de dependencias:
    - Análisis de los puntos de entrada para encontrar los módulos necesarios: `import(), require()`
      - Análisis estático
      - Tree shaking: elimina código innecesario
    - Se reorganizan módulos, evitando dependencias circulares, facilitar la carga
  - Transformación de código:
    - Minificación, Transpilado, optimización

- Flujo del empaquetado:
  - Empaquetado de activos: CSS, imágenes, ...
    - Importación de activos
    - Procesado de activos para optimización, etc. generando una nueva ruta para la versión procesada.
    - Sustitución de la declaración de importación por las nuevas rutas
    - Generación del bundle incorporando los activos procesados.
    - Servir activos
  - Empaquetado:
    - Generación de un único fichero JavaScript
    - División de código de forma inteligente para sólo servir el que se necesita

Tecnologías Web: Cliente

- Flujo del empaquetado:
  - Empaquetado:
    - Generación de un único fichero JavaScript
    - División de código de forma inteligente para sólo servir el que se necesita
- Webpack configuración compleja, sobrecarga que afecta al rendimiento, generación pesada, ralentiza el desarrollo
- Parcel:
  - Libre de configuración, compatibilidad con varios tipos de activos: HTML, CSS, Javascript, imágenes, fuentes.
  - Automatización de tareas

- Parcel:
  - Instalación:
    - `npm install -g parcel@next`
  - Seleccionar el punto de entrada
  - Ejecución en modo desarrollo:
    - `npm parcel serve --port 8080 src/index.html`
    - `--host, --open, -- no-autoinstall, --no-hmr, --log-level`
    - Opciones: protocolo https, selección ruta para la caché de archivos.
    - Generación de archivos: se crean los archivos empaquetados y archivos .map que los relacionan con los originales. Desactivación: `--no-source-maps`

- Parcel:
  - Instalación:
    - `npm install -g parcel@next`
  - Seleccionar el punto de entrada
  - Ejecución en modo desarrollo:
    - `npm parcel serve --port 8080 src/index.html`
    - `--host, --open, -- no-autoinstall, --no-hmr, --log-level`
    - Opciones: protocolo https, selección ruta para la caché de archivos.
    - Generación de archivos: se crean los archivos empaquetados y archivos .map que los relacionan con los originales. Desactivación: `--no-source-maps`

- Parcel:
  - Producción:
    - `npm parcel build src/index.html`
    - `--no-optimize --no-scope-hoist --public-url url --dist-dir ruta --no-cache --cache-dir --no-content-hash`
    - Parámetros de configuración:
      - fichero **.parcelrc**

```
{
   "extends": "@parcel/config-default",
   "transformers": {
       "*.{png,jpg}": ["@parcel/transformer-raw"],
       "*.{ts,tsx}": ["@parcel/transformer-typescript-tsc"]
   }
}
```


Universidad
de La Laguna