Universitat Oberta de Catalunya

Aula

M4.257 - Herramientas HTML y CSS aula 1

M2. Un workflow moderno de desarrollo front-end

Inicio:	Fin:
01/03/21	**11/04/21**

Presentación

En este segundo módulo de la asignatura **Herramientas HTML y CSS I** empezaremos a tocar código y nos familiarizaremos con un conjunto de herramientas moderno que permite que el desarrollo sea más eficiente.

Si alguna vez habéis desarrollado el front-end de una página web habréis trabajado con un editor de texto editando ficheros HTML, CSS y JavaScript. Actualmente los desarrolladores web utilizamos herramientas algo más complejas que el simple editor de texto. Si bien de entrada puede parecer complicar un proceso que es relativamente sencillo, el uso de herramientas avanzadas facilita a la larga el desarrollo, ya que permite la gestión de dependencias, la automatización de ciertas tareas repetitivas y el uso de lenguajes y utilidades más avanzadas.

En primer lugar, será necesario que os familiaricéis con el uso del terminal. Aprenderemos cómo navegar por la estructura de directorios y ejecutar órdenes desde el terminal.

A continuación nos plantearemos qué flujo de trabajo (o workflow) necesita un proyecto en función de sus características. Una vez ya conozcáis su funcionamiento básico, empezaremos a ver algunas de las herramientas básicas para el desarrollo front-end, como los gestores de paquetes como npm y los module bundlers como Parcel y para qué sirven.

Finalmente, desarrollaremos el workflow diseñado, y crearemos un boilerplate con Parcel desde cero, que nos servirá para desarrollar futuras aplicaciones. Un boilerplate es una plantilla de estructura de ficheros y conjunto de herramientas a partir del cual se puede empezar un desarrollo con facilidad. Se convertirá, pues, en el punto de inicio desde donde se realizarán las PECs de este curso.

Como el producto de esta asignatura será una página web, durante este módulo la publicaremos en internet.

Como siempre, los materiales de aprendizaje, disponibles en la sección Recursos básicos, y en la sección Contenidos y actividades encontraréis diferentes ideas y propuestas de actividades sencillas para consolidar los conocimientos adquiridos con los recursos propuestos. Si tenéis cualquier duda sobre alguna de las actividades propuestas o no sabéis la respuesta a alguna de las preguntas, abrid un hilo en el foro y lo comentamos entre todos.

Objetivos de aprendizaje

- Familiarizarse con el uso del terminal.
- Conocer el funcionamiento y la utilidad de los gestores de paquetes como npm.
- Conocer el funcionamiento y la utilidad de los module bundlers como Parcel.
- Conocer las diferencias entre entornos de desarrollo y de producción.
- Crear un boilerplate basado en Parcel que permita desarrollar proyectos web basados en un workflow moderno.
- Publicar una web a internet.

Recursos básicos

En este módulo empezaremos a trabajar con código HTML, CSS y JavaScript. Para editarlo, deberéis disponer de un buen editor de texto, como **Visual Studio Code** , Sublime Text, Atom o cualquier otro que uséis habitualmente y que tenga características similares.

Los usuarios de Windows necesitaréis, en primer lugar, instalar el subsistema de Linux para Windows, siguiendo estas instrucciones . Si sois usuario de MacOS o Linux, ya disponéis de las herramientas necesarias para comenzar.

> **Aviso: Aunque se puede usar directamente el sistema de Windows para parte del desarrollo de esta asignatura, hay algunas de las actividades que vamos a realizar donde Windows puede dar resultados incorrectos o no funcionar. Si dispones de un sistema Windows y decides no instalar el subsistema de Linux, es posible que encuentres problemas para realizar correctamente las actividades**

Las lecturas que os ayudarán a alcanzar los objetivos de aprendizaje de este módulo son las siguientes:

- Introducción al uso del terminal: https://medium.com/@grace.m.nolan/terminal-for-beginners-e492ba10902a
- Introducción a los gestores de paquetes: http://codylindley.com/techpro/2013_04_12__package-managers-an-introducto/
- Sobre las herramientas de desarrollo front-end: https://medium.freecodecamp.org/making-sense-of-front-end-build-tools-3a1b3a87043b

Las documentaciones de:

- npm: https://docs.npmjs.com/
- Parcel: https://parceljs.org/getting_started.html

Atención : las documentaciones técnicas de los lenguajes y herramientas que figuran aquí son muy extensas y contienen mucha más información de la necesaria para alcanzar los objetivos de aprendizaje de este módulo. Las documentaciones han de ser usadas como referencia y no como lectura secuencial obligatoria. Solo será necesario que leáis y entendáis lo requerido para este curso. Por estos motivos, se recomienda la lectura de estos documentos y la realización de las actividades de acuerdo con las pautas propuestas en la sección Contenidos y actividades. Os animamos a que, además, leáis y aprendáis todo lo que os vaya interesando para complementar los contenidos de este curso.

Contenidos y actividades
Uso del terminal
Sobre los gestores de paquetes
Sobre los module bundlers
Sobre los preprocesadores de código
Publicación de la web en internet

Uso del terminal
Ideas clave

El terminal es la herramienta que nos permitirá ejecutar todas las otras herramientas de este módulo. Por lo tanto es necesario que os habituéis a esta herramienta y conozcáis el funcionamiento básico.

Actividad 1

Si eres usuario de Windows y no lo has hecho todavía, instala el subsistema de Linux para Windows siguiendo estas instrucciones . Cada vez que nos referimos al terminal, en el caso de Windows, querrá decir acceder al subsistema de Linux en modo terminal.

Lee el documento Command line crash course y realiza los ejemplos que allí se proponen hasta la sección `A slightly more complex example`. En la parte superior está la sección Basic built-in terminal commands con los comandos más habituales.

Un poco más exhaustivo, pero también un artículo muy recomendable para empezar con la terminal es el artículo Terminal for beginners! de Grace Nelson. Ambos artículos hacen una introducción similar, pero se complementan bien.

En guide.bash.academy encontrarás mucha más información sobre el uso del terminal.

Sobre los gestores de paquetes
Ideas clave

Un gestor de paquetes nos sirve para instalar y gestionar dependencias externas a nuestro proyecto. Las dependencias son librerías externas que son necesarias para el desarrollo: desde un optimizador de SVG como SVGO a un framework de desarrollo como Vue, React o Angular.

Utilizaremos **npm, el gestor de paquetes más habitual en front y que se instala por defecto con NodeJS.** Una alternativa a npm, aunque con pequeñas diferencias en el funcionamiento es Yarn que también es muy popular.

Una confusión habitual al empezar a trabajar con Node es saber diferenciar entre Node y npm. Node es un entorno de programación que permite ejecutar código Javascript en el servidor. De este modo podemos ejecutar scripts de Javascript tanto en el servidor como en el navegador, que también soporta Javascript. NPM es un repositorio de scripts de Javascript (paquetes) que puedes descargar y ejecutar en Node. Si esto os genera dudas comentadlo en el foro y lo tratamos con más detalle.

Actividad 1

Comprueba si tienes instalado npm en tu ordenador. Para hacerlo, ejecuta la orden `npm -v` en el terminal. Si te sale un número de versión (algo así como `6.14.4`), es que ya tienes npm instalado y por lo tanto puedes ir a la siguiente actividad. Si no, te saldrá un error.

Si no tienes npm instalado, es necesario que lo instales. Para hacerlo, ve a la página de descargas de Node.js e instala la versión adecuada para tu sistema operativo. Busca entre el listado de versiones la LTS (Long Term Support).

Atención : si eres usuario de Windows NO debes instalar la versión para Windows, sino la versión adecuada a la distribución de Linux que hayas instalado, según las instrucciones de esta página .

Por ejemplo, si has instalado Ubuntu, es necesario que ejecutes las órdenes:

```
curl -sL https://deb.nodesource.com/setup_10.x
sudo -E bash -sudo apt-get install -y nodejs
```

Una vez finalizada la instalación, comprueba que se ha realizado correctamente ejecutando la orden `npm -v`.

Actividad 2

Como lectura introductoria lee el artículo I finally made sense of front end build tools. You can, too . Esta lectura presenta los tipos de herramientas que utilizaremos en este curso y también mucha de la terminología que usaremos, por lo tanto es importante que lo leas con atención

Actividad 3

Lee las siguientes secciones de la documentación de npm:

- About npm
- Packages and modules > About packages and modules
- Packages and modules > Creating a package.json file
- Packages and modules > Downloading and installing packages locally
- Packages and modules > Uninstalling packages and dependencies
- Packages and modules > Updating packages downloaded from the registry

Ten a mano la documentación de la sección Resolving EACCES permissions errors when installing packages globally por si tienes problemas más adelante.

Recuerda que cada vez que instalamos un paquete de npm con el comando `npm i` se crea una nueva carpeta en `node_modules` y podemos ver la biblioteca que hemos descargado.

También te recomendamos las secciones relativas al concepto de gestores de paquetes y sobre npm del documento Package Managers: An Introductory Guide For The Uninitiated Front-End Developer.

Sobre los module bundlers
Ideas clave

- **Un module bundler nos permite automatizar una serie de tareas así como gestionar las dependencias que hemos instalado con npm.**
- Utilizaremos Parcel como module bundler. Otro module bundler muy conocido, y el más usado en la industria, es Webpack. Sin embargo, resulta más complicado de configurar.
- Las tareas que automatiza un module bundler pueden variar en función del entorno de destino de nuestra aplicación. Actualmente, los module bundlers realizan estas tareas de forma transparente al usuario, aunque permiten la configuración para casos especiales.
- Si estamos preparando nuestro código para un entorno de producción, el module bundler realizará tareas que compilen, concatenen y minifiquen las hojas de estilos y los scripts, además de otros procesos de optimización sobre imágenes y demás assets.
- Si por el contrario estamos preparando el entorno de desarrollo, el objetivo del module bundler es ofrecer la mejor experiencia posible al desarrollador. Una de las principales funcionalidades en este entorno es lanzar un servidor de desarrollo y monitorizar los cambios en los archivos para refrescar el navegador automáticamente. En este entorno, el bundler no optimiza nuestros assets, pero a cambio nos ofrece una mayor velocidad de desarrollo.

Actividad 1
Lee y entiende los conceptos básicos de Parcel .

Sigue los pasos hasta **Multiple entry files** (sin incluir este apartado). Al final, deberías haber podido lanzar un servidor de desarrollo. Fíjate que deberás usar varias herramientas, como npm, además de instalar Parcel y crear algunos archivos HTML y JavaScript.

Antes de continuar, si has instalado Parcel en global (`npm install -g parcel-bundler`) en la lectura anterior, desinstálalo antes de continuar. Instalar paquetes en global está desaconsejado excepto para paquetes muy específicos. Lee qué diferencias hay y por qué

Para desinstalar ejecuta:

```
npm uninstall -g parcel-bundler
```

Sigue ahora las instrucciones del apartado Adding parcel to your project . Si te fijas, ahora volvemos a instalar Parcel, pero en local.

- ¿Sabrías explicar para qué sirve cada script que has añadido a tu `package.json` ?
- ¿Qué otros cambios has introducido o han aparecido en el archivo `package.json` ?
- ¿Qué diferencias hay entre instalar en global y en local?¿Por qué recomendamos instalar en local?

Actividad 2
Realiza las siguientes actividades y responde las siguientes preguntas:

1. Ejecuta la orden `npm run build`.
   - ¿Qué ocurre en el terminal?
   - ¿Qué ocurre en la carpeta de trabajo?
   - ¿Qué crees que hay en la nueva carpeta que ha aparecido?
   - Inspecciona los archivos generados. ¿Qué diferencias hay?
2. Para comparar las diferencias abre algún archivo (por ejemplo, el `index.html` en ambas rutas) con tu editor de texto. Observa de nuevo el terminal. Habiendo explorado la carpeta, ¿cómo interpretas la información que aparece en el terminal?
3. Ejecuta la orden `npm run dev`.
   - ¿Qué ocurre?
   - ¿Qué diferencias hay con la ejecución de la orden anterior?
   - Mantén esta ventana del terminal abierta. Si no se te abre el navegador automáticamente, abre una ventana del navegador e introduce la URL que te ha aparecido en el terminal (normalmente, `http://localhost:1234`).
4. Prueba a editar el archivo HTML de la raíz del proyecto. Por ejemplo, añade el siguiente código justo después de la etiqueta:

```
<h1>Hello world</h1>
```

1. Guarda el archivo y, sin recargar la pestaña del navegador que has abierto en el paso anterior, observa si el cambio aparece en la página.
   - ¿Cómo crees que esto beneficia a los programadores y programadoras que trabajan en aplicaciones con module bundlers?

Actividad 3

Realiza las siguientes actividades y responde las siguientes preguntas:

1. Vuelve a ejecutar la orden `npm run build`.
   - ¿Qué ocurre en la carpeta de dist/?
   - Como habrás podido comprobar, Parcel genera una nueva versión de archivos listos para producción, pero no elimina los antiguos. Si no los vamos borrando, crecerían indefinidamente.
2. Ejecuta la siguiente orden: `npm install --save-dev rimraf npm-run-all`.
   - ¿Qué cambios se han producido en el fichero package.json? En el mismo package.json, añade las siguientes líneas dentro del apartado "scripts":

     ```
     "parcel:dev": "parcel index.html",
     "parcel:build": "parcel build index.html",
     "clean": "rimraf dist .cache .cache-loader"
     ```

     Finalmente, sustituye las tareas anteriores "dev" y "build" por las siguientes:

     ```
     "dev": "npm-run-all clean parcel:dev",
     "build": "npm-run-all clean parcel:build",
     ```

     Vuelve a ejecutar la orden `npm run build`.
   - ¿Qué ocurre ahora en la carpeta de dist/?
   - ¿Qué diferencia hay respecto la ejecución anterior?
   - ¿Qué ventajas crees que aporta este cambio?
   - ¿Sabrías descifrar cuál es el objetivo de cada una de las órdenes que hemos escrito en los puntos anteriores?
   - ¿Y los paquetes adicionales que hemos instalado, para qué sirven?

Sobre los preprocesadores de código

Ideas clave

- Históricamente, cada navegador ha ejecutado HTML, CSS y JavaScript de formas ligeramente diferentes. Para complicarlo aún más, también existen variaciones entre versiones del mismo navegador. Afortunadamente, en los últimos años los navegadores han unificado mucho cómo ejecutan el código y hoy en día estas diferencias se han reducido.
- Los navegadores solo son capaces de procesar y ejecutar HTML, CSS y JavaScript. Por lo tanto, cualquier lenguaje debe terminar convertido en uno de estos tres para ser comprendido por los navegadores.
- Entendemos por preprocesadores (a veces también llamados compiladores o transpiladores, aunque técnicamente hay ciertos matices) aquellas herramientas que reciben archivos escritos en varios lenguajes y los convierten en código comprensible por el navegador.
- Los preprocesadores que generan CSS se usan entre otras cosas para añadir funcionalidades que no existen en CSS estándar o para generar otra sintaxis distinta: por ejemplo, en su momento permitieron el uso de variables (hoy ya existen las custom properties) o la posibilidad de anidación (que todavía no forma parte del estándar). En este módulo trabajaremos con PostCSS.
- Similar a lo que hacen los preprocesadores de CSS, los preprocesadores que generan JavaScript suelen ser usados para transformar lenguajes completamente diferentes a JavaScript, y también para transformar versiones modernas del lenguaje en versiones aptas para todos los navegadores. En este módulo configuraremos Babel, aunque en esta asignatura no lo veremos a fondo.
- Existen otros preprocesadores tanto para CSS y JS como para HTML: TypeScript , para JavaScript tipado; Sass , un lenguaje para escribir CSS; o Pug , un lenguaje para escribir HTML, son algunas de las alternativas más populares a día de hoy.
- Un workflow moderno de desarrollo front-end debe incluir la configuración de estas herramientas, de forma que la persona que esté trabajando pueda aprovechar sus beneficios de forma transparente.
- Tan importante como la configuración del workflow es su documentación. Hay que tener en cuenta el contexto de un equipo, donde varias personas y roles deben poder entrar en el proyecto y comprender qué herramientas tienen a su disposición. Si os interesa el tema, la tercera charla del Modern Web Event 2018 organizado por la UOC, llamada Front-end, herramientas y consenso, hace un buen repaso a la evolución de la industria en este aspecto.
- A día de hoy, es muy habitual que los proyectos, por pequeños que sean, incluyan preprocesadores para CSS y JS, y en menor frecuencia, de HTML.

Actividad 1

Parcel ya incorpora una configuración básica de Babel por defecto. Así pues, no es necesario instalar nada para usarlo.

Realiza las siguientes actividades y responde las siguientes preguntas

- Ejecuta la orden `npm run build` . Observa cómo, además del código JavaScript que hemos escrito en la actividad anterior, el archivo resultante incluye más líneas de código que Parcel añade automáticamente. Habitualmente, el código escrito por el usuario comienza en la segunda línea del archivo.
- Ahora, sustituye el contenido del archivo index.js en la raíz del proyecto por el siguiente:

```
const name = 'world';
console.log(`Hello ${name}`);
```

- Ejecuta la orden `npm run dev` . Comprueba que la funcionalidad (escribir un texto en la consola de las herramientas de desarrolladores) no ha cambiado.
- Ejecuta la orden `npm run build`
  - ¿Qué diferencias hay respecto el código JavaScript anterior generado por Parcel?

Actividad 2

Una práctica habitual es indicar sobre qué versiones de navegadores debe ser compatible el código que procesan los preprocesadores por medio de los module bundlers. Esta funcionalidad se implementa utilizando Browserslist .

Añade un archivo llamado **.browserslistrc** (¡ojo!, el nombre empieza por un punto) en la raíz del proyecto. En este archivo, introduce las siguientes cuatro líneas:

```
last                                                4
version
>               2               %
not               dead
IE               11
```

Aquí tenéis un listado completo de las posibles combinaciones que ofrece Browserlists para elegir el soporte a navegadores. A partir de este momento, Babel tendrá en cuenta esta configuración a la hora de transformar el código JavaScript de nuestro proyecto.

- ¿Sabrías explicar en una frase qué navegadores soportará nuestro boilerplate?

Actividad 3

En esta actividad entenderemos el uso de los preprocesadores de CSS y los añadiremos a nuestro boilerplate. Al igual que con Babel, Parcel ya incluye postCSS así que no es necesario instalarlo.

A continuación os proponemos unas lecturas recomendadas. Estás lecturas no están pensadas para ejecutar los comandos en el proyecto (eso lo haremos luego). De momento solo lee estos artículos:

- El artículo An Introduction to PostCSS hasta la sección `Installing PostCSS` (sin incluir este apartado).
- El artículo Autoprefixer: A Postprocessor for Dealing with Vendor Prefixes in the Best Possible Way .
- El apartado PostCSS de la documentación de Parcel.

Realiza las siguientes actividades y responde las siguientes preguntas:

1. Vamos a instalar autoprefixer:
    1. Debido a una issue de compatibilidad entre versiones de Parcel 1 y postCSS 8 vamos a instalar una versión específica de autoprefixer.
    2. Ejecuta `npm install --save-dev autoprefixer@9.8.6`. Esto instalará la versión 9.8.6 que es la última compatible con Parcel 1 y postcss 7.
    3. Comprueba que se ha instalado la versión correcta en tu `package.json`
2. Añade un archivo llamado `.postcssrc` (ojo, el nombre comienza con un punto!) en la raíz del proyecto. En este archivo, introduce las siguientes líneas:

```
{               "plugins"               : {             "autoprefixer"
:               true            }}
```

1. Añade las siguientes líneas al index.html dentro de la etiqueta `head`

```
<               link                                                    rel
=               "stylesheet"
href               =               "style.css"              >
```

1. A continuación, crea un archivo llamado style.css en la raíz del proyecto, y añade las siguientes líneas:

```
body              {             height              :             100vh
;             color              :             #1f1f1f              ;
background              :              linear-gradient              (to bottom,
#bada55, #c0ffee);    }
```

1. Ejecuta la orden `npm run build`. Como hemos hecho anteriormente, comprueba el resultado del archivo style.css creado en la carpeta dist/.
2. ¿Qué diferencias ves?

3. ¿Cuáles de los cambios crees que son responsabilidad de PostCSS y el Autoprefixer?
4. ¿Sabrías encontrar otra propiedad que necesite ser modificada por el Autoprefixer y otra que no lo necesite?

Publicación de la web en internet

Ideas clave

- Para que una web sea visible de cara al público, no solamente desde el entorno de desarrollo local, tiene que estar publicada en un servidor web.
- La versión de la web que tiene que estar en el servidor es, evidentemente, la versión de producción que genera Parcel.
- Actualmente existen decenas de servicios donde publicar webs sencillas de forma gratuita y cómoda. La mayoría utilizan el contenido publicado en un servicio de hosting de repositorios Git.
- Git es un sistema de control de versiones. Los contenidos se guardan en repositorios. El contenido de estos repositorios es abierto al público, de forma que cualquier usuario podrá ver el código del proyecto.
- En este módulo nos basaremos en GitHub como plataforma donde crear el repositorio Git, y en Netlify como proveedor del servidor donde publicar la web.
- No hay problema si conoces y quieres utilizar servicios alternativos a GitHub (GitLab, Bitbucket) o a Netlify (GitHub Pages, Now). En cualquier caso, no hace falta gastar ni un euro en esta actividad.

Actividad 1

En esta actividad vamos a crear un repositorio Git para la web.

- Asegúrate de haberte dado de alta gratuitamente en GitHub .
- A continuación, crea un repositorio Git y vincula tu carpeta local (donde tienes el código en tu ordenador). En esta guía oficial encontrarás explicaciones paso a paso con las instrucciones que debes ejecutar.

Actividad 2

En esta actividad crearemos un espacio web en Netlify.

- Asegúrate de haberte dado de alta gratuitamente en Netlify .
- A continuación, crea un nuevo sitio web a partir del código disponible en GitHub. En este vídeo puede seguir paso a paso el proceso para vincular los dos servicios.
- **Accede a la URL generada** y comprueba que la web es visible.

Actividad 3

Una de las muchas funcionalidades interesantes de Netlify es que actualiza tu web a medida que actualizas el contenido del repositorio Git (a esta práctica se la conoce como Continuous Deployment).

- Para probarlo, realiza cualquier cambio a tu código local (un texto, una imagen, un color...). A continuación, sube este cambio a tu repositorio Git. Para hacerlo, vuelve a seguir los pasos commit y push que has visto anteriormente en la guía de Git. Cada vez que hagas cambios significantes en el código, debes repetir estos pasos para almacenar los cambios en el repositorio.
- Accede al panel de control de Netlify, y comprueba que la plataforma ha detectado el cambio y ha empezado el proceso para publicar una versión actualizada de tu web.
- Accede a la URL generada y comprueba que la nueva versión es visible.