ENTORNO DE TRABAJO

Prof. Antonio Gabriel González Casado
2ºDAW
Desarrollo Web en Entorno Cliente

ÍNDICE

EDITORES DE CÓDIGO

QUÉ ES UN EDITOR DE CÓDIGO

Un editor de código es un software específicamente diseñado para editar código fuente de un determinado lenguaje.

Actualmente la variedad de editores es muy amplia. Se pueden dividir en dos tipos:

- Editores de texto con extensiones para código fuente y desarrolladores: Visual Studio Code, Sublime Text, Vim, Notepad++,...
- Entornos de Desarrollo Integrado (IDE): Además de tener editor de código permite compilar, ejecutar, depurar el software y muchas más otras acciones relacionadas con el desarrollo de software: Visual Studio, IntelliJ, PyCharm, Eclipse,...

VISUAL STUDIO CODE (VSCODE)

Visual Studio Code, es un editor/depurador de código liviano desarrollado por Microsoft que en los últimos años ha ganado mucha popularidad entre los desarrolladores de Frontend (entre otros). Esto es debido a las siguientes características:

- Tiene extensiones que lo hace muy personalizable
- Soporta una gran cantidad de lenguajes y formatos de ficheros.
- Es multiplataforma
- Es rápido y liviano.

VSCODE - EXTENSIONES I

Existen multitud de extensiones para usar en VSCode. Estas son algunas de las más útiles para un desarrollador frontend.

Prettier - Code formatter. Es uno de los formateadores de código más conocidos.

Turbo Console Log. Autogenerador de mensajes de log mediante la consola de JavaScript.

Package Json Upgrade. Detecta cuando hay actualizaciones en nuestros paquetes.

Import Cost. Muestra el coste (en tamaño) que implica importar una librería.

VSCODE - EXTENSIONES II

Git Graph. Muestra el gráfico de flujos de un repositorio de git.

Color Highlight. Renderiza los colores que estén especificados mediante un código hexadecimal.

Better Comments. Mejora y estandariza los comentarios sobre nuestro código fuente.

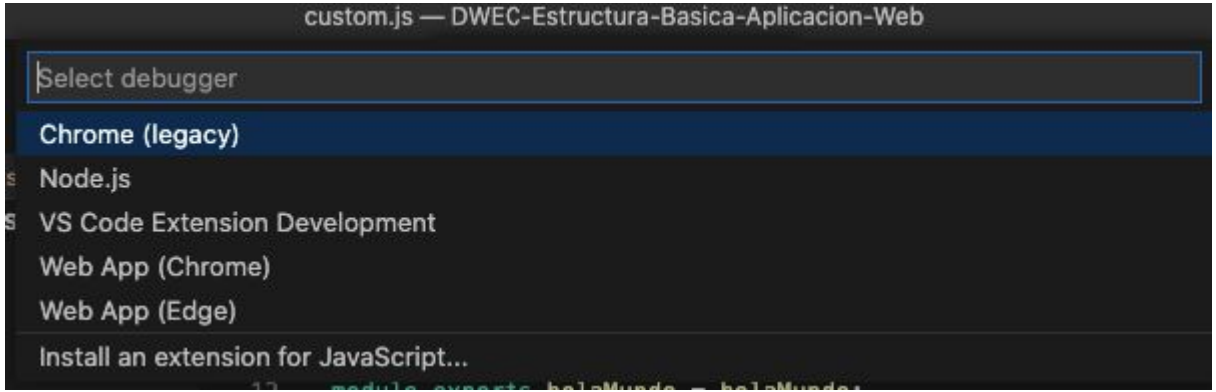
Auto Rename Tag. Renombrado automático de las etiquetas html.

ESLint. Permite detectar errores y valida la calidad del código escrito.

VSCODE - Depurador

El depurador de VSCode permite parar la ejecución del código javascript para inspeccionar variables, expresiones y poder identificar posibles errores del código en tiempo de ejecución.

Permite la depuración de código ejecutando sobre un navegador web o fuera del mismo sobre node.js



VSCODE - Terminal

La terminal de VSCode permite lanzar comandos shell script sobre el proyecto. Por ejemplo, si el proyecto es node.js se pueden lanzar los comandos:

- npm install #Instala las dependencias
- npm test #ejecuta los test unitarios

Hay tres tipos de terminales:

- bash
- zsh
- JavaScript Debug Terminal

NODE.JS

QUÉ ES NODE.JS

NODE.JS es un software que nos permite interpretar código JavaScript fuera del navegador.

Permite la programación de otro tipo de aplicaciones fuera del navegador.

Node.JS es uno de los servidores web más populares y las aplicaciones de backend construidas en JavaScript están en auge.

En el lado cliente permite instalar paquetes con la utilidad npm interesantes para el desarrollo.

NVM

Utilidad que permite administrar de forma cómoda la gestión de las versiones node.js

El comando nvm permite instalar y cambiar fácilmente la versión de node.js para que se puedan realizar distintas pruebas en ellas y evitar problemas de compatibilidad.

Los comandos más interesantes son:

- nvm install <versión>. Dónde <versión> debe ser sustituida por la que se quiere instalar o por latest para instalar la última estable.
- nvm ls. Muestra las versiones de node instaladas.
- nvm ls available (windows) o nvm ls-remote (linux). Muestra las versiones de node.js disponibles para instalar.
- nvm use <versión>. Hace que la versión indicada sea la que funcione como versión por defecto.
- nvm uninstall <versión>. Desinstala la versión indicada.
- nvm root. Muestra la ruta donde están almacenadas las distintas versiones de node.js
- nvm -v Muestra la versión instalada del propio nvm.

NPM

Es la utilidad para instalar los paquetes de software. No confundir con nvm que es el gestor de instalaciones de node.js.

Permite instalar todo tipo de utilidades que facilitan el trabajo de desarrollo.

Los comandos más usados son:

- npm ls. Muestra los paquetes instalados en el workspace.
- npm install <paquete>. Instala el paquete indicado.
- npm uninstall <paquete>. Desinstala el paquete indicado.

A los comandos anteriores si le añadimos **el argumento -g** entonces la instalación será válida para todo el sistema operativo en vez de para el workspace.

- npm root. Muestra la ruta donde están instalados los paquetes.
- npm search <texto>. Busca en los repositorios de internet nombres de paquetes que contengan el texto indicado.
- npm -v Muestra la versión instalada de npm.

LIBRERÍAS Y UTILIDADES

SISTEMAS DE CONTROL DE VERSIONES I

Es un software que permite la gestión de las distintas versiones que se van generando de una aplicación. Además permite que varios desarrolladores puedan integrar sus desarrollos de forma colaborativa.

Hay distintos software para este fin, pero el más utilizado es GIT.

GIT es un software de código abierto que se puede instalar en local o en una instalación remota, pero actualmente existen servicios en la nube muy completos y de uso gratuíto que permiten ahorrarnos este paso, como por ejemplo GITHUB.

SISTEMAS DE CONTROL DE VERSIONES II

Los conceptos básicos de un sistema de control de versiones son:

- Repository (repositorio). Proyecto en git donde se almacena el código fuente y los archivos.
- Branch (rama). Es un entorno donde se dispone de los códigos fuentes para un determinado fin que puede ser:
  - Desarrollar de forma independiente
  - Elaborar una determinada funcionalidad con el objetivo
  - Destinada a arreglar un bug.
  - Probar funcionalidades (testing)
  - Medir la calidad del código (quality)
- Release (liberar). Imagen del código fuente y de los archivos que se van a liberar para la entrega o la puesta en producción.
- Checkout. Descargar el código fuente y los archivos que están en una determinada rama.
- Stage (escenario). Lista de archivos que forman parte de un commit.

Gitflow. Metodología ampliamente extendida para trabajar con ramas de forma colaborativa y liberar versiones con una nomenclatura concreta.

Actualmente está tomando más popularidad la metodología Trunk Based Development que trata de evitar algunas desventajas de Gitflow.

SISTEMAS DE CONTROL DE VERSIONES III

Existen varios clientes gráficos que ayudan a gestionar los repositorios y los flujos entre ramas. Algunos de ellos son:

GitKraken. Excelente herramienta pero es propietaria. La versión gratuíta tiene funcionalidades limitadas.

Sourcetree. La herramienta de open source más extendida.

Si se trabaja con Github. La plataforma tiene su propio cliente: GitHub Desktop.

TEST UNITARIOS

A un desarrollador donde se le va el mayor tiempo es en la depuración y prueba de su código.

Los test unitarios buscan probar todas las casuísticas que se pueden dar sobre un determinado código fuente. A las distintas casuísticas se les denomina caminos de prueba.

En JavaScript existen frameworks para poder testear el código fuente. Las más populares son:

- Jest
- Mocha
- Jasmine

LOGS

Los logs son trazas de texto que un sistema va generando y que dejan constancia de los errores que se han producido o información de los procesos que se han ejecutado. Se usan para detectar patrones y mejorar el sistema, detección de errores o fugas de información,...

En JavaScript los logs se escriben mediante el comando genérico **console.log('mensaje', [parámetros])**

Admite también las siguientes especializaciones:

- **error.** Escribe un mensaje de error. Ideal usarlo dentro de un catch.
- **warn**. Escribe un mensaje de aviso. Ideal usarlo en situaciones controladas anómalas.
- **info**. Escribe un mensaje de información. Ideal usarlo en las entradas y salidas de los métodos y para informar de acciones relevantes en el sistema.
- **trace**. Escribe la traza de log hasta el punto donde está este comando. Ideal para depuración avanzada.

En el lado cliente estos logs no son tan necesarios ya que se mostrarán en la consola del navegador, entonces ayudarán en el proceso de desarrollo y pruebas, **pero antes de pasar a producción hay que anularlos.**

CALIDAD

El código fuente que se escribe debe hacerse siguiendo unos estándares de calidad.

Hay herramientas como ESLint que son imprescindibles que estén instalados en el Editor de Código y que los errores y advertencias sean revisados y corregidos constantemente.

Gestión de Entornos

Cuando se escribe un sistema, algunas veces se tienen que definir variables que están condicionados al entorno en el que se ejecuta la aplicación (desarrollo, pruebas, producción).

Para solventar adecuadamente esta gestión de variables se suele recurrir a herramientas como **dotenv**

Por ejemplo, podemos desactivar los logs de la consola si el entorno es producción.

NAVEGADORES

ARQUITECTURA DE UN NAVEGADOR

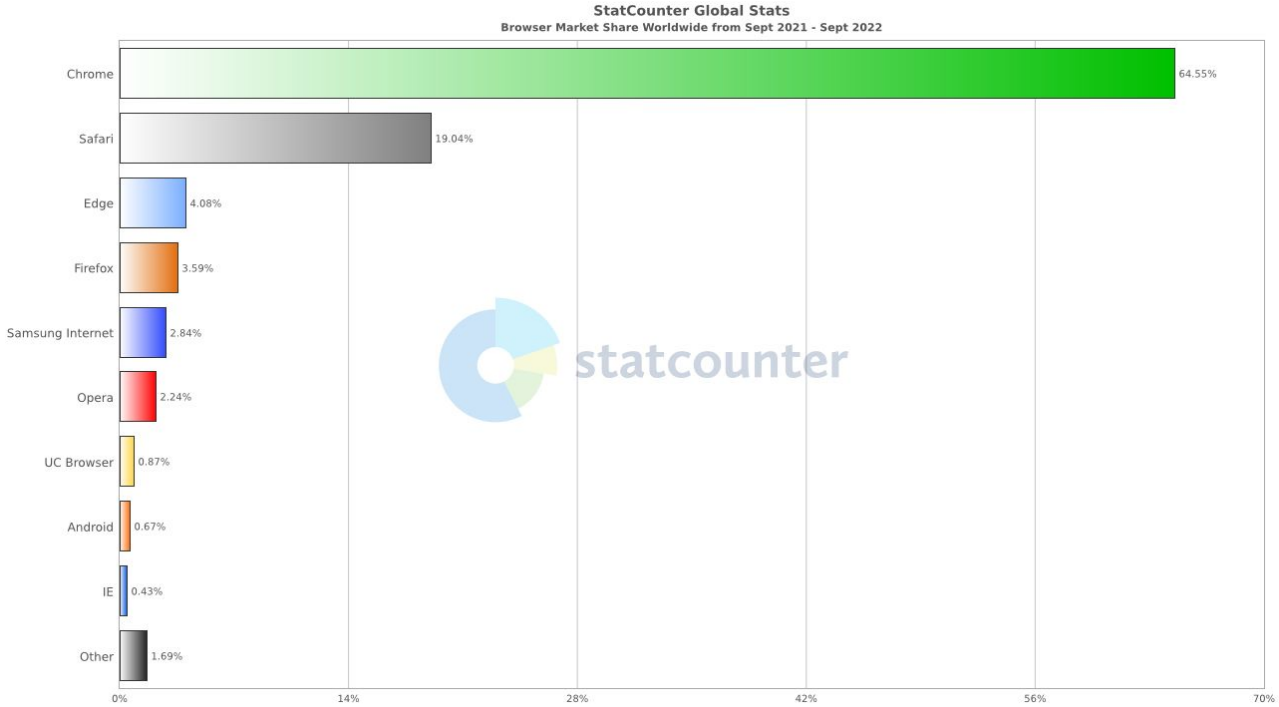
Un navegador es un Software que solicita a un servidor el acceso a uno o varios recursos (a través de una url) que están alojados en el mismo y recibe distintas fuentes en HTML, CSS y JavaScript, los cuales los renderiza para que el usuario vea toda la información en distintos formatos texto, secciones, imágenes,...

Un navegador está formado por los siguientes componentes principales:

- **Interfaz de usuario**: barra de direcciones, el menú de marcadores, el botón de avanzar y retroceder,...
- **Motor de renderización**: transforma los documentos HTML y otros recursos de una página web en una representación visual interpretable por un ser humano en el dispositivo que lo ha solicitado.
  - Webkit (Safari), Blink (Chromium, Opera), Gecko (Firefox), Quantum (Firefox), EdgeHTML (Microsoft Edge)
- **Motor de búsqueda**: Coordina las acciones entre la interfaz y el motor de renderización
- **Servidor de la interfaz**: permite mostrar widgets básicos, tales como ventanas y cuadros combinados
- **Red:** realiza las llamadas de red; por ejemplo, las solicitudes HTTP.
- **Almacenamiento de datos**: en cookies, web storage,...
- **Motor de JavaScript**: se encarga de interpretar, compilar y optimizar el código JavaScript:
  - SpiderMonkey (Firefox), V8 (Chrome, Android browser), Nitro (Safari), JavaScriptCore (Safari, Chrome para IOS).

NAVEGADORES MÁS USADOS

Actualmente Google Chrome es el navegador más usado, pero al desarrollar se deben tener en cuenta pruebas de compatibilidad de la aplicación en los navegadores más usados.



CHROME DEVTOOLS

Chrome DevTools es un conjunto de herramientas para desarrolladores web integradas directamente en el navegador Google Chrome.

DevTools puede ayudar a editar html, css y JavaScript sobre la marcha y diagnosticar problemas rápidamente.

https://developer.chrome.com/docs/devtools/overview/

FIREFOX DEVTOOLS

Al igual que Google Chrome Firefox tiene un conjunto de herramientas para desarrolladores de aplicaciones web.

Ambas son muy similares.

https://firefox-source-docs.mozilla.org/devtools-user/

CONCLUSIÓN

- El editor de código más popular para frontend es VSCode por sus extensiones y ser liviano.
- Node.js permite ejecutar código JavaScript fuera del Navegador
- Un buen desarrollo debe usar un sistema de control de versiones bajo una metodología, test unitarios, buena calidad de código, logs y gestión de los entornos.
- Google Chrome es el navegador más usado pero hay que desarrollar para todos.
-