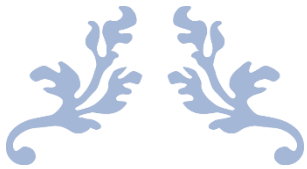


INSTALACIÓN Y CONFIGURACIÓN DE UN ENTORNO DE TRABAJO PARA PROGRAMAR CON JAVASCRIPT

DESARROLLO WEB EN ENTORNO CLIENTE

19 DE SEPTIEMBRE DE 2017

2DAW B
Emiliano Montesdeoca del Puerto

Índice

Enunciado

Realizar la instalación y configuración de un entorno de trabajo para programar con JavaScript.

- Para ello tendrás que buscar e instalar en tu sistema operativo 3 editores de páginas web, de los cuales, uno de ellos debe ser un editor online. Todas las aplicaciones que instales tendrán que ser gratuitas y tendrás que explicar las razones por las que has elegido tu editor web entre muchos otros.
    - multilenguaje
    - chequeo de la sintaxis del lenguaje
    - formateo del código (espaciado o identación, etc.…)
    - selección o agrupación de bloques de código (funciones, etc.…)
    - emparejamiento de llaves
    - autocompletado
    - atajos de teclado
    - paquetes que pueden instalarse y cuáles recomendarías
    - SSOO en los que puede instalarse o usarse. En el caso del editor web navegadores soportados
- Usando la dirección de validación del W3C lleva a cabo la validación de la página de Google e indica los tipos de errores encontrados, cita 3 errores diferentes detectados y la solución propuesta a cada uno de ellos.

Criterios de puntuación. Total 10 puntos

Los 10 puntos que tiene esta tarea están asignados de la siguiente forma:

- 2,5 puntos por cada una de las aplicaciones que tienes que instalar (3) y la justificación de esta elección de acuerdo a los criterios expuestos (mínimo).
- 1,5 puntos por la validación de la página de Google en el validador W3C.
- 1 puntos adicionales por la claridad y presentación de los resultados.

Tendrás que documentar textualmente los pasos que sigues acompañados de las capturas de las pantallas (máximo 4 capturas por cada instalación).

Objetivo

El objetivo principal de esta práctica es documentar brevemente diferentes entornos de trabajos que se utilizaran durante el curso, contar sus características e intentar hacer una comparación entre ellos.

Esta práctica ayudara a descubrir nuevos entornos de trabajos, que posiblemente más adelante, se podrán usar sin ningún tipo de miedo al cambio.

Cabe destacar que, aunque existan entornos de trabajo mejores que otros, siempre existe la subjetividad del desarrollador a la hora de usarlo, y aunque sea mejor, uno prefiere utilizar algo con lo que está más cómodo.

También se hablará un poco sobre los entornos de desarrollo en la nube, que siguen en crecimiento.

Por último, se comentará sobre la validación del W3C, para identificar tipos de errores típicos en la consola de JavaScript.

Introducción

JavaScript

JavaScript (abreviado comúnmente JS) es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como **orientado a objetos**, basado en prototipos, imperativo, débilmente tipado y dinámico.

Se utiliza principalmente en su forma del **lado del cliente**, implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas, aunque existe una forma de JavaScript del lado del servidor.

Su uso en aplicaciones externas a la web, por ejemplo, en documentos PDF, aplicaciones de escritorio (mayoritariamente widgets) es también significativo.

JavaScript se diseñó con una sintaxis similar a C, aunque adopta nombres y convenciones del lenguaje de programación Java. Sin embargo, Java y JavaScript tienen semánticas y propósitos diferentes.

A continuación, un ejemplo de código JavaScript embebido:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Ejemplo sencillo</title>

<h1 id="header">Esto es JavaScript</h1>

<script>
    document.body.appendChild(document.createTextNode('Hola Mundo!'));

    var h1 = document.getElementById('header'); // contiene la referencia al
tag <h1>
    h1 = document.getElementsByTagName('h1')[0]; // accediendo al mismo
elemento <h1>
</script>

<noscript>Tu navegador no admite JavaScript, o JavaScript está
deshabilitado.</noscript>

</head>
<body>/*...*/</body>
</html>
```

Un entorno de desarrollo integrado o entorno de desarrollo interactivo, en inglés **Integrated Development Environment (IDE)**, es una aplicación informática que proporciona servicios integrales para facilitarle al desarrollador o programador el desarrollo de software.

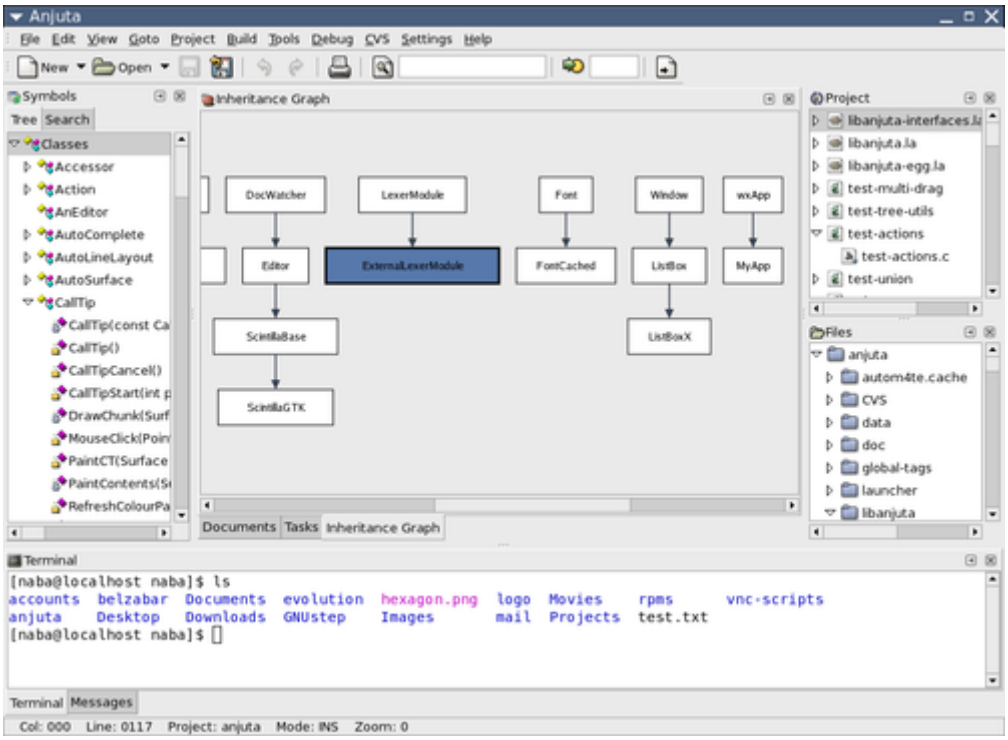
Los IDE están diseñados para maximizar la productividad del programador proporcionando componentes muy unidos con interfaces de usuario similares. Los IDE presentan un único programa en el que se lleva a cabo todo el desarrollo.

Generalmente, este programa suele ofrecer muchas características para la **creación, modificación, compilación, implementación y depuración de software.**

Uno de los propósitos de los IDE es reducir la configuración necesaria para reconstruir múltiples utilidades de desarrollo, en vez de proveer el mismo set de servicios como una unidad cohesiva.

Reduciendo ese tiempo de ajustes, se puede incrementar la productividad de desarrollo, en casos donde aprender a usar un IDE es más rápido que integrar manualmente todas las herramientas por separado.

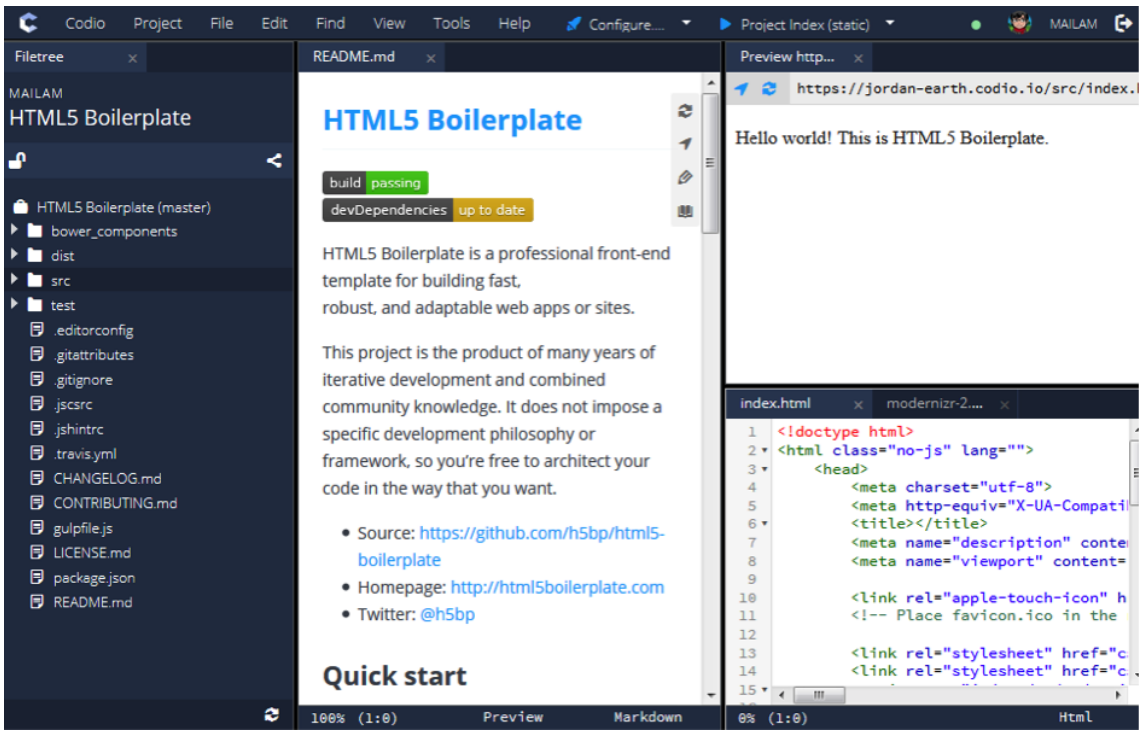
En la siguiente imagen, **Anjuta**, un entorno de desarrollo integrado de C y C++ para el ambiente GNOME.



El concepto de este tipo de herramientas en sencillo. **Son Entornos de Desarrollo Integrado (IDEs)** con las ventajas de una solución alojada en la nube. Estas tecnologías tienen diferentes características

- **Son algo más que un editor de código**: igual que los IDEs se podrían definir como editores de código con esteroides, las plataformas de desarrollo en la nube podrían pasar por ser IDEs con esteroides. Este tipo de herramientas son más bien Plataformas de Desarrollo como Servicio (Development Platform as a Service- dPaaS), que ofrecen una serie de funcionalidades sobre costes y productividad más elevadas.
- **Menos tiempo de implementación**: este tipo de plataformas reducen mucho los tiempos de instalación. Menos tiempo, menos costes.
- **Programa desde cualquier sitio**: no es necesario estar delante de tu máquina para continuar programando tu producto. Al estar alojada en la nube, es posible hacerlo desde cualquier sitio y dispositivo (ordenador o tableta). Único requisito: **estar conectado a la Red**
- **Trabajo colaborativo en tiempo real**: una de sus grandes ventajas es que varios desarrolladores pueden estar trabajando en el mismo proyecto a la vez y utilizar servicios de chat online para comunicarse.
- **Personalización del entorno de desarrollo**: este tipo de herramientas permiten instalar dependencias para los proyectos de forma independiente. Cuando se programa en local, en muchas ocasiones se pueden tener complicaciones porque las dependencias para unos proyectos afectan a otros o perjudican a otras aplicaciones web.

A continuación, una imagen de **Codio**, un IDE en la nube:



Entornos de trabajo

Visual Studio Community

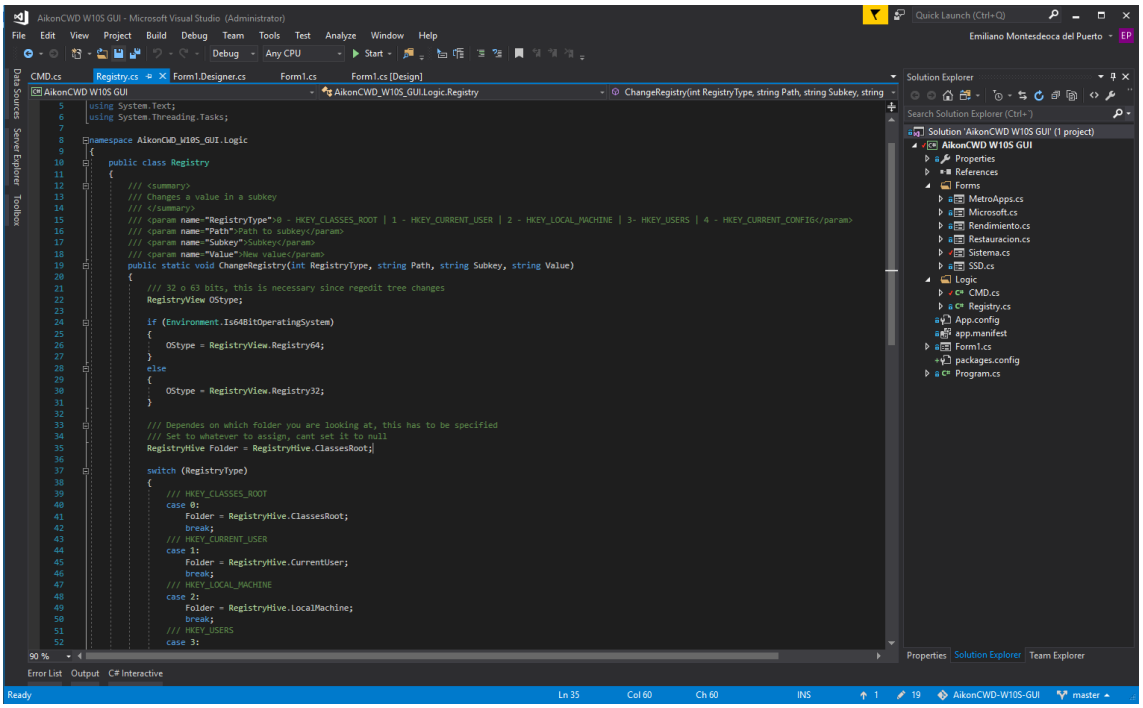
Introducción

Visual Studio Community es el IDE gratuito de Microsoft para sistemas operativos basados en Windows, aunque actualmente Microsoft ha introducido Visual Studio Code para sistemas operativos en Macintosh y Linux, para poder agrandar su mercado a otros OS.

Visual Studio incluye una amplia variedad de paquetes de lenguajes y tiene un aspecto bastante moderno comparado con algunos entornos con la misma experiencia.

Este IDE es perfecto para utilizarlo en casi todos los lenguajes de Windows, aquí se puede realizar aplicaciones en ASP.net, modo consola, Windows Forms, WPF y mucho más.

A continuación, una captura de un proyecto en Visual Studio:



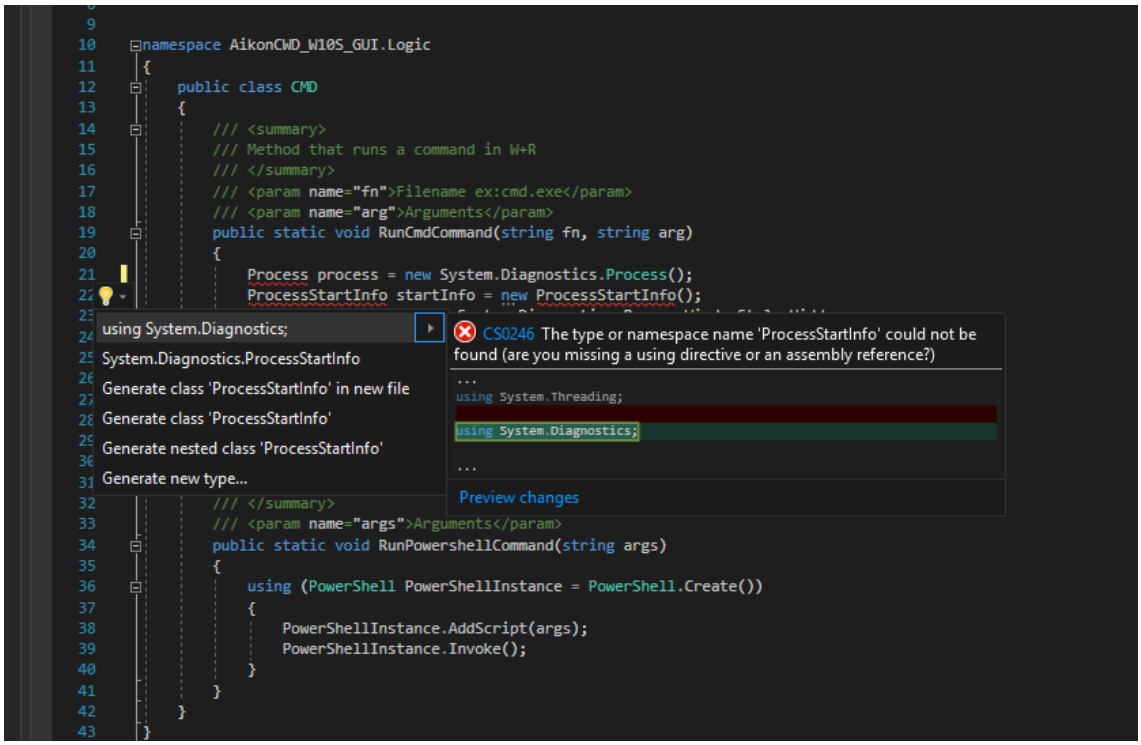
Como podemos ver, es un IDE bastante moderno, algo en lo que Microsoft se ha centrado bastante, también vemos a la derecha el gestor de proyecto, con diferentes pestañas.

Todo esto se puede configurar fácilmente desde el panel de control de Visual Studio, para poder elegir que ver y no ver en la pantalla de trabajo.

Sintaxis

Visual Studio tiene la capacidad de detectar si hay algún error en el código que escribimos, y no solo detectarlo, pero también te da una fácil solución para que puedas rápidamente solucionarlo y seguir adelante.

En la siguiente imagen, se puede ver como el objeto de la clase **System.Diagnostics.Process** no es encontrado por Visual Studio al no estar agregado el paquete a la proyecto, pero rápidamente sugiere una simple solución:



Este comportamiento funciona con casi todos los tipos de errores que se encuentra el Visual Studio, siempre va a sugerir una solución.

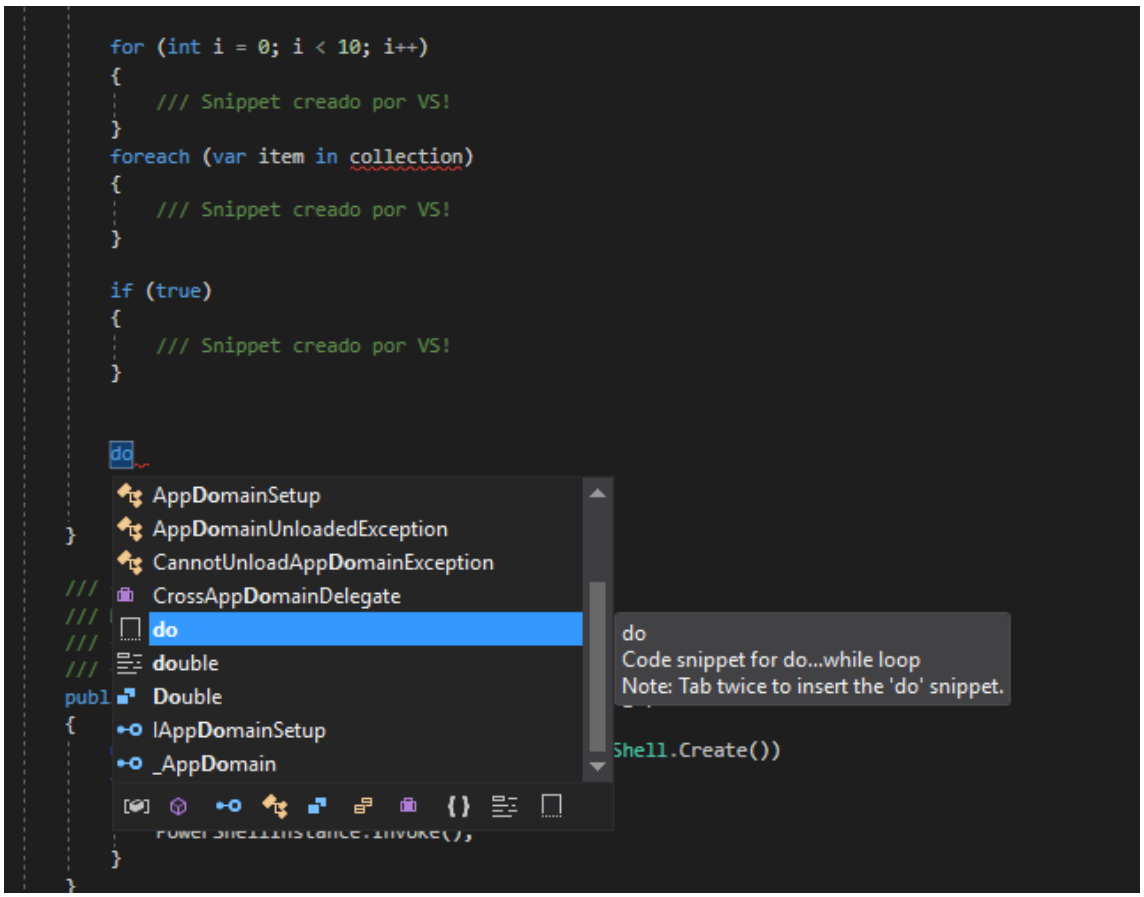
Por ejemplo, si no encuentra una propiedad, te va a sugerir crear una propiedad al inicio de la clase, o si no encuentra una clase, te va a sugerir crear una clase nueva en el proyecto, como es de esperar, esto lo hace el Visual Studio, y no tienes que hacer nada, simplemente agregarlo.

Obviamente, un gran IDE también traer agrupación de contenidos que se puede fácilmente hacer con un atajo de teclado (**Ctrl+M,O**), y se puede separar las funciones o métodos terminados de los que estamos trabajando, se puede ver fácilmente en la foto, donde se ven los **+**, eso sirve para agrupar y desagrupar.

Autocompletado

Visual Studio tiene grandes cantidades de palabras clave para el autocompletado, simplemente con escribir esa palabra y apretar el tabulador dos veces, te ahorrara escribir toda la sintaxis.

Por ejemplo, si queremos escribir un bucle **for**, es tan simple como escribir for y apretar el tabulador dos veces para que se genere:



He puesto otros ejemplos como con el bucle **foreach** y la condición **if** para que se pueda ver ejemplos de snippets creado por Visual Studio, a continuación, dejare una lista de todos los **snippets** que están en Visual Studio actualmente:

#if, #region, attribute, checked, class, ctor, cw, do, else, enum, equals, exception, for, foreach, for, if, indexer, interface, invoke, iterator, ireindex, lock, mbox, namespace, prop, ropfull, propg, sim, struct, svm, switch, try, tryf, unchecked, unsafe, using, while.

Atajos de teclado

Existen muchísimos atajos de teclado para Visual Studio, aunque personalmente no utilizo muchos, pero los que voy aprendiendo y me sirven para ahorrar tiempo o mejorar la calidad del código:

DESCRIPCION	ATAJO DE TECLADO
BÚSQUEDA RÁPIDA	Ctrl+F
REEMPLAZO RÁPIDO	Ctrl+H
MODO DE SUGERENCIA DE INTELLISENSE	Ctrl+Alt+Espacio (alternancia)
DELIMITAR CON	Ctrl+K, S
DELIMITAR CON	Ctrl+K, S
INICIE LA DEPURACIÓN	F5
PASO A PASO POR PROCEDIMIENTOS	F10
PASO A PASO POR INSTRUCCIONES	F11
DAR ESTILO AL CODIGO	Ctrl+K,D

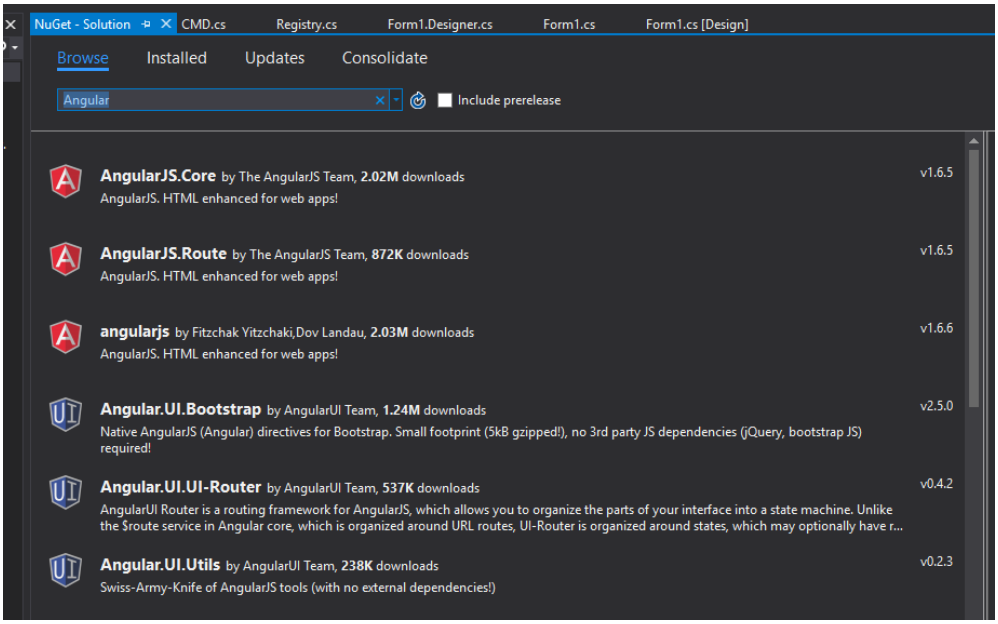
Sin duda el que más utilizo, es el de formatear el código (**Ctrl+K,D**), es bastante útil y te sirve para dejar un código limpio y elegante.

Paquetes

Visual Studio utiliza **NuGet**, es un manejador de paquetes que permite instalar y actualizar librerías y herramientas en Visual Studio. Viene incluido en la instalación de este y se agregar funcionalidades tanto por su parte grafica como por la consola de PowerShell.

Aquí se suelen agregar paquetes a la solución y no tanto al Visual Studio, por ejemplo, si queremos agregar AngularJS a la solución, podemos buscarlo por el gestor de paquetes y de ahí agregarlo, o si queremos agregar Bootstrap.

Dejo una imagen de como es NuGet a la hora de buscar Angular en el buscador:



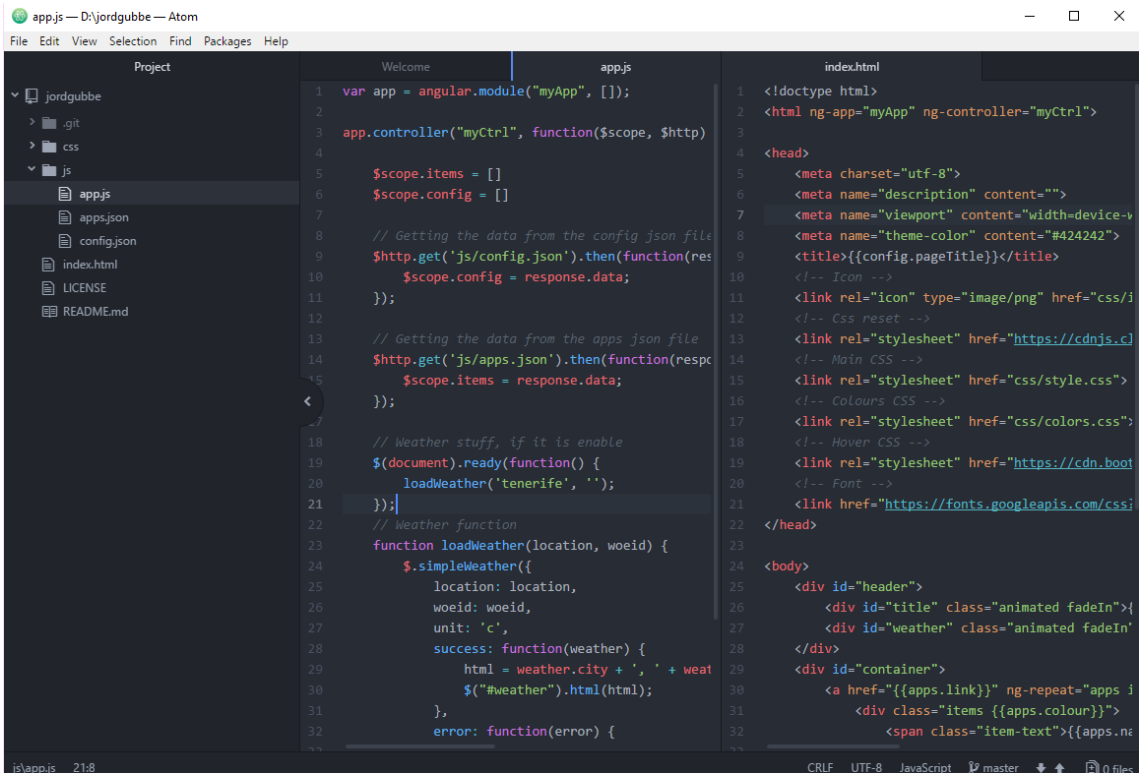
Atom

Introducción

Atom es un IDE que he empezado a utilizar recientemente, me llamo la atención la cantidad de atención que está recibiendo actualmente en sus últimas versiones. Hay que decir que, está desarrollado por la empresa **GitHub**, asi que solo se puede esperar grandes cosas.

Lo que me llamo la atención de este IDE es que parece que puedes hacer lo que quieras con el, puedes meterte en su configuración y tocarle todo, era de esperar, ya que te lo publicitan como un editor totalmente ***hackeable*** y parece que cumple con su prometido.

Una imagen simple de como se ve Atom, el estilo es bastante simplista tirando a un toque moderno, y como siempre, en negro de fábrica.



El estilo obviamente se puede cambiar, hay una galería donde la gente puede crear y subir sus propios estilos. Y el orden de las ventanas también, puedes separar hasta 3 páginas de código, para poder ser más productivo.

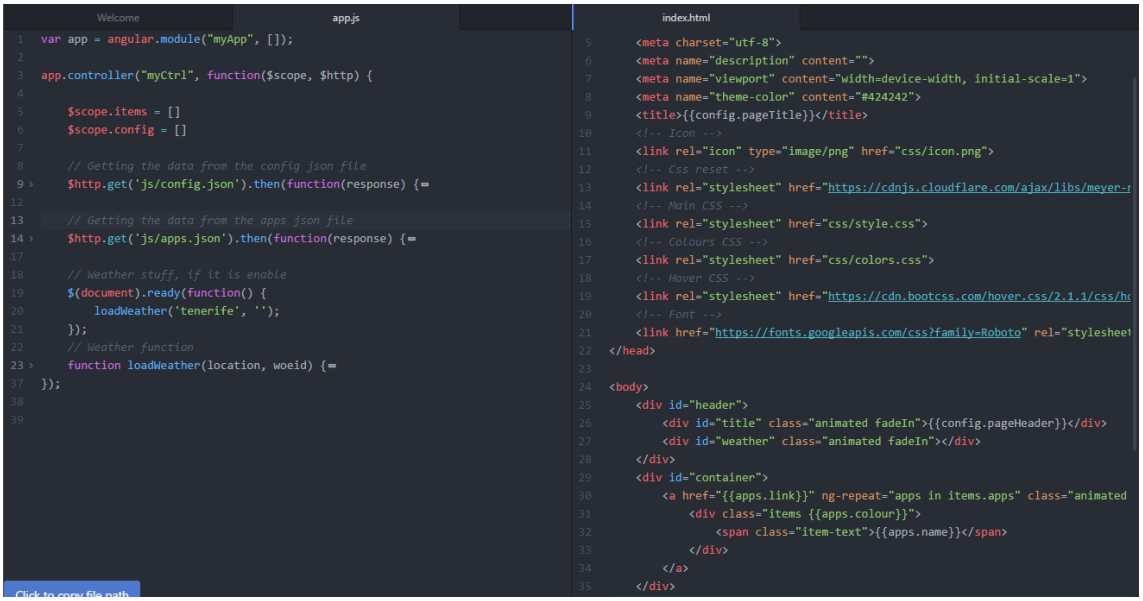
En cuanto al lenguaje, Atom tiene una cantidad de paquetes de lenguajes.

Sintaxis

En cuanto a gestión de sintaxis, Atom no se queda atrás, tiene, como el Visual Studio una gran facilidad en cuanto gestión de funciones/métodos o simplemente condiciones o bucles, simplemente hay que pulsas en el numero de la línea y se contrae todo totalmente.

En cuanto al formateo del código, hace falta un paquete obligatorio llamada **atom-beautify** que sirve para formatear código de muchísimos lenguajes, pero de él se hablara más adelante.

Aun asi, Atom es capaz de escribir código formateado cuando se genera el snippet. A continuación, se mostrará una imagen de código reducido y otro expandido para ver, lo bien formateado que queda el código con Atom:



Atom tiene una ayuda excepcional, que siempre esta cuando la necesitas, una vez palabra y ya está la ayuda lista para poder hacerte el trabajo.

Autocompletado

El auto completado de Atom es lo mejor que tiene, tiene una cantidad de snippets listos para poder usarlos y no perder tiempo rellenando y lo mejor es que tiene para todo tipo de lenguajes, en este caso JavaScript.

Lo único que hay que hacer para ello es escribir **y te aparecerán en las ayudas unas flechitas verdes** que indican que son un snippet, simplemente pulsas o das ***Enter*** sobre y listo, se te reproduce el snippet.

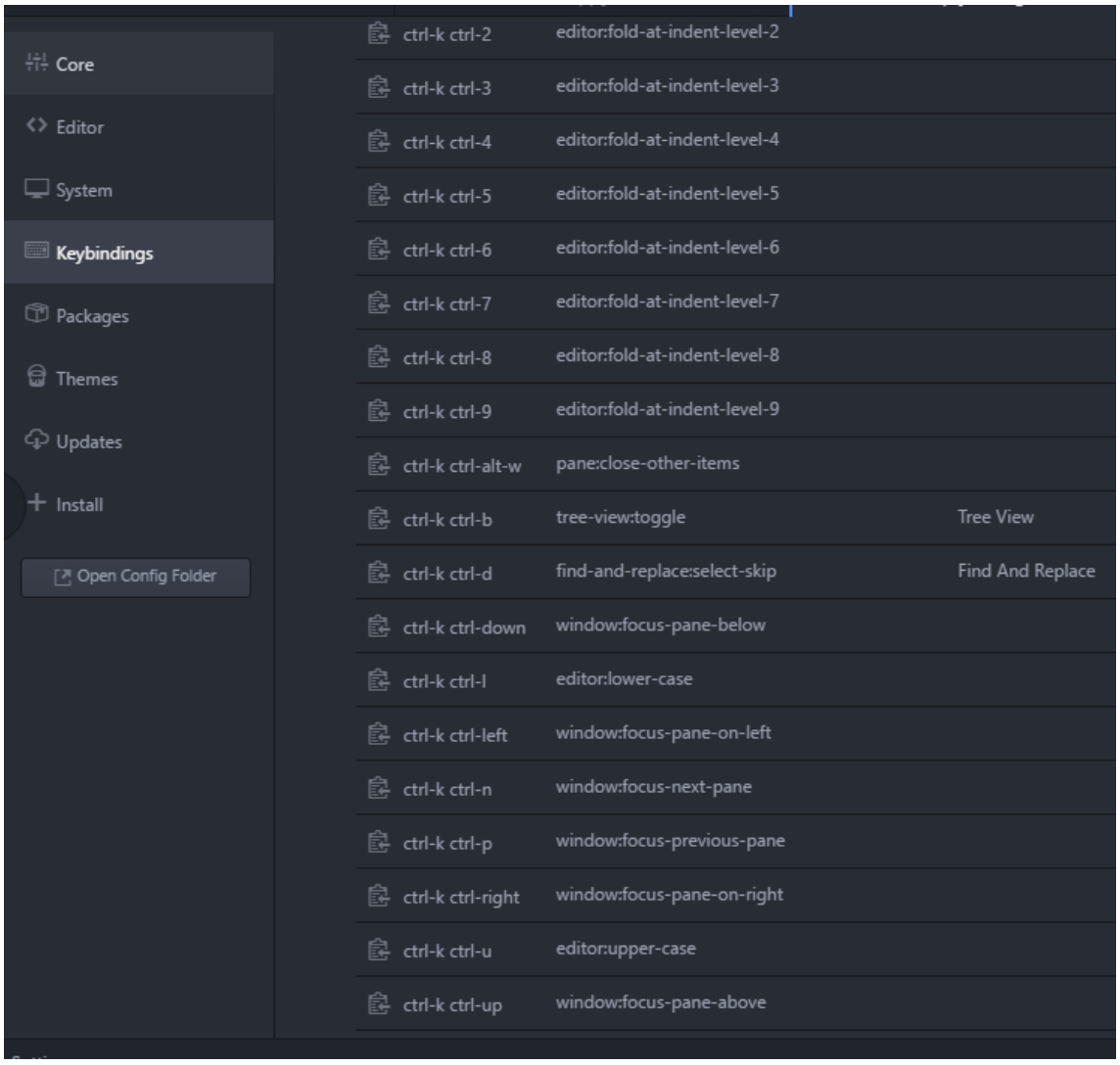
Como se puede ver en la foto, cuando escribes algo en el editor, él ya te saca la ayuda y las que tienen la flecha verde a la izquierda, significa que será código creado por Atom.

Atajos de teclado

En cuanto a los atajos de teclado, Atom **trae de fabrica una cantidad enorme de atajos**, pero realmente **los más útiles son los que añaden los paquetes que instalas**, ya que es lo que más se utiliza en Atom.

Si un usuario quiere ver que atajos de teclado hay actualmente, simplemente se tiene que ir a **Ajustes**.

Para acceder a él, se accede a través del menú o pulsando ***Ctrl + ,*** y ahí accedes **a Atajos de teclado** y se podrán ver todos los atajos existentes.



En este menú también se encuentra los atajos de teclado de los paquetes instalados.

Paquetes

Lo mejor sin duda de Atom, es la cantidad de paquetes que se le pueden instalar. Sin duda para exprimir lo mejor de este IDE es agregar mínimo tres o cuatro paquetes indispensables que harán que la experiencia sea muchísimo mejor y más eficiente.

Para acceder a los paquetes, hay que ir a **Opciones** e pulsar en **Paquetes**, sin embargo, para instalar un paquete hay que pulsar en **Instalar**, que abrirá un buscador donde se introduce el nombre del paquete para luego pulsar sobre instalar. Después de esto el paquete se habrá instalado.

Mi **recomendación** de paquetes para Atom son los siguientes:

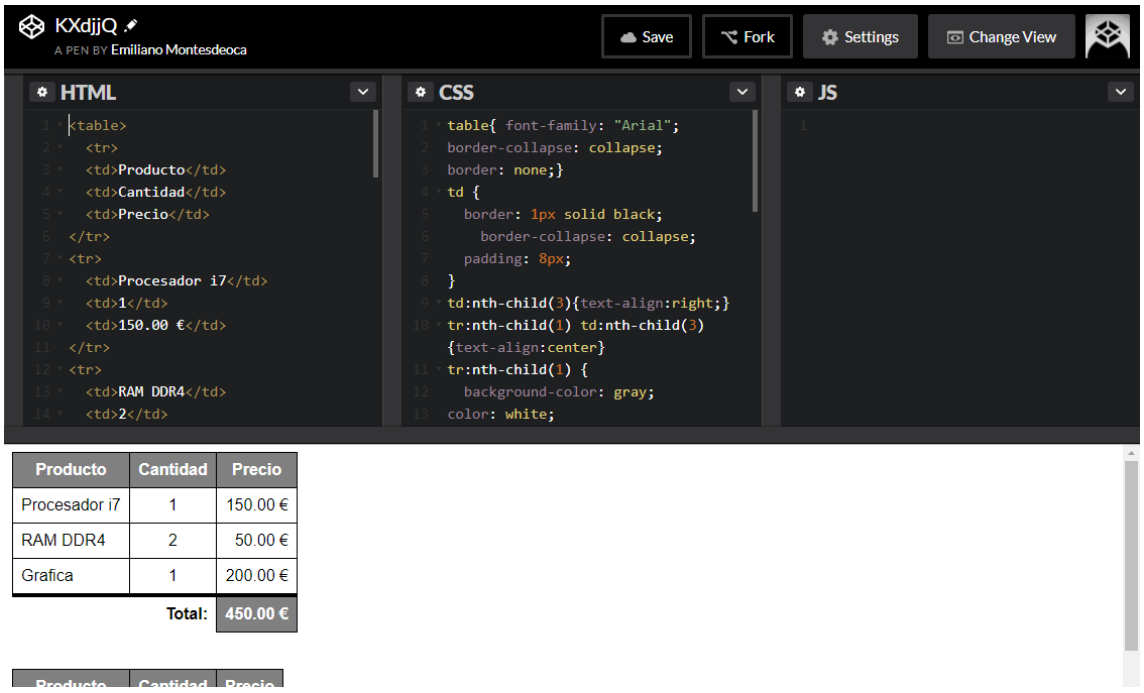
1. **Color-picker:** un paquete para escribir el código de color sin herramientas externas.
2. **Atom-beautify:** el mejor paquete sin duda, sirve para tabular el código y dejarlo elegante.
3. **W3c-validation:** paquete que sirve para validar tu código.
4. **Minimap:** un paquete que añade un mapa total del fichero para poder acceder más fácilmente.
5. **Highlight-selected:** un paquete necesario cuando pulsas sobre una palabra y te la sobresalta en todos los lugares en la que se encuentra.

CodePen

Introduccion

CodePen es una plataforma de edición de código online la cual solo necesita conexión a internet y **funciona desde el navegador**, permite el teste de HTML, CSS y JavaScript.

Lo mejor de CodePen es lo fácil que es de usar, ya que es tan intuitiva que es imposible perderse y no entender cómo utilizarla. También se pueden agregar frameworks y sin duda lo mejor es lo fácil que es compartir un proyecto.



Como se puede ver en la imagen, recalco la facilidad a la hora de realizar las cosas en CodePen, el primer cuadrado es para HTML, el segundo para CSS y el tercero para JavaScript, no hay ninguna dificultad en identificar esto.

Sintaxis

En cuanto a la sintaxis, CodePen ayuda a identificar cada etiqueta, cada identificador y cada función, pero no es algo extremadamente inteligente, simplemente está hecho para funcionar bien y nada más.

Cabe destacar que CodePen no es un IDE en la nube, **sino que simplemente es una herramienta para hacer pequeñas cosas.**

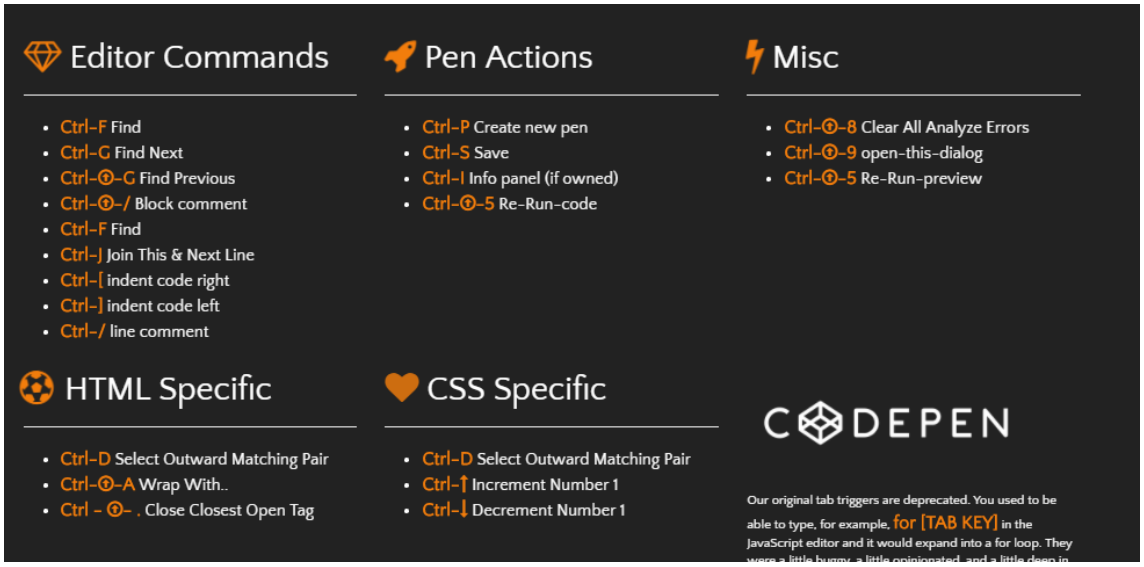
Autocompletado

Punto negativo, pero no indispensable, CodePen no incluye ningún tipo de autocompletado, todo el código tiene que ser escrito por el autor.

Es una pena, pero, **como he dicho antes**, considero a CodePen como una herramienta para mostrar o hacer pruebas rápidamente y no para realizar desarrollo serio.

Atajos de teclado

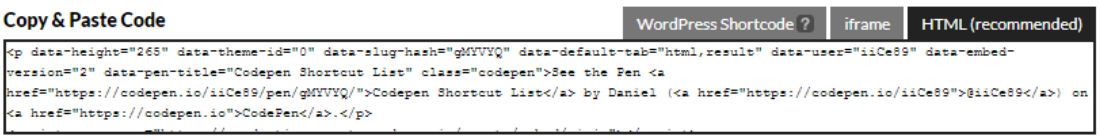
CodePen incluye atajos de teclado, que se activa con el **Tabulador** y luego pulsando alguna combinación. Aquí dejo una imagen donde se puede ver alguno de los atajos de teclado, no hay nada importante y que se pueda sobresaltar, pero simplemente que tenga atajos lo hace una herramienta que se puede mejorar la efectividad.



Paquetes

Para CodePen, en vez de hablar de paquetes, deberíamos de hablar de frameworks. CodePen tiene la cualidad de dejarte agregar frameworks como por ejemplo Bootstrap y Angular, asi puedes realizar tus pruebas sin tener que agregar toda la librería.

Aparte de esto, se puede compartir el CodePen, en la barra que se encuentra abajo un botón de compartir, el cual nos dejará un código que se podrá insertar en una página web o red social.



Comparación final

A continuación, una tabla comparativa entre los tres entornos analizados:

Característica	Visual Studio	Atom	CodePen
Multilenguaje	Si	Si	No
Chequeo de sintaxis	Si	Si	Si
Formateo de código	Si	Si*	No
Agrupación de código	Si	Si	Si
Emparejamiento de llaves	Si	Si	Si
Control de versiones	Si	Si	No
Autocompletado	Si	Si*	No
Atajos de teclado	Si	Si	Si
Paquetes	Si	Si	Si

(*) Hay que agregar un paquete para esta característica.

Ahora una comparación de estos entornos para diferentes sistemas operativos:

Sistema operativo	Visual Studio	Atom
Windows	Si	Si
Linux	Si*	Si
Macintosh	Si*	Si

(*) Hay una versión especial para este sistema operativo, **Visual Studio Code**.

Finalmente, para CodePen una comparación de navegadores:

Navegador	CodePen
Google Chrome	Si
Mozilla Firefox	Si
Internet Explorer	Si
Microsoft Edge	Si
Safari	Si
Opera	Si

Opinión personal

Sin duda, ambos los dos entornos de desarrollo locales cumplen y sobrepasan los requisitos mínimos, como se ha visto en la tabla comparativa. Un punto a favor de estas IDEs es que están respaldadas por empresas en las que se puede confiar, que se sabe que no van a dejar tirado al usuario y siempre van a responder con actualizaciones inmediatas.

No solo cumplen con los puntos expuesto, sino que incluyen una característica que, en mi opinión es de lo mejor, es que incluyen control de versiones.

Esto era de esperar de Atom, ya que está desarrollada por la empresa más grande encargada de control de versiones, aun asi **Microsoft Visual Studio** en su versión **Professional o Enterprise**, incluye control de versiones no solo para **Team Fundation Server**(TFS), sino que también lo hace para todos los gestores de versiones, como puede ser Github o GitLab.

Aunque Microsoft da versión gratuita de **Azure, su solución DevOps,** para poder guardar nuestro código en sus servidores y asi poder usarlo con el control de versiones.

En cuanto a CodePen, no se le puede pedir perfección a una aplicación que **ya está limitada por el navegador**, pero aun asi, se puede decir que CodePen, **junto a JSFiddle**, son una herramienta espectacular para desarrolladores que **necesitan una solución rápida** para enseñar a clientes o simplemente para guardar cosas que van a necesitar más adelante.

También sirve para persona que le gusta realizar pruebas **y no cuentan con un ordenador potente**, ya que no CodePen no exige altos recursos para ser ejecutado.

Me gustaría remarcar que, esta selección de entornos de desarrollo es puramente subjetiva, **y concuerdo con otros desarrolladores que puede haber mejores entornos** para desarrollar en otros lenguajes.

Validación del W3C

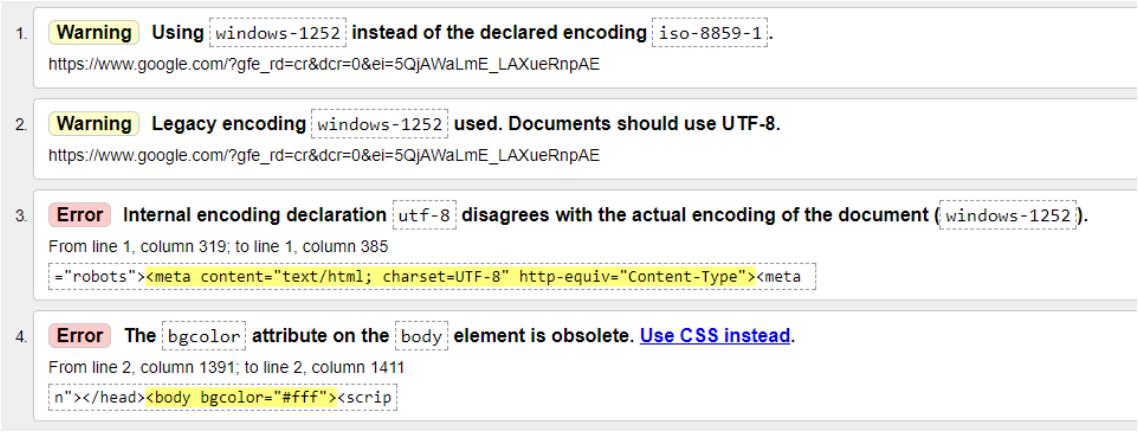
Introducción

W3C son las siglas de **World Wide Web Consortium**, y es una comunidad internacional donde los estados miembros trabajan para poder desarrollar estándares para el desarrollo web y asi ayudar a un mejor desarrollo del Internet a nivel mundial.

¿Por que hay que validar?

1. Mejora el ranking en motores de búsqueda.
2. Enseña las mejores prácticas.
3. La experiencia de usuario mejora.
4. Menos probabilidad de fallo en diseño responsivo.
5. Mayor accesibilidad.
6. Validar ayuda a la programación y mantenimiento.
7. Sirve para depurar errores.

Tipos de errores



Podemos separar los errores encontrados en tres tipos:

1. Errores de codificación.
2. Errores de atributos de elementos que se debería de usar en CSS.
3. Errores de referencia a elementos no permitidos en algunas etiquetas.

Soluciones

Para solucionar los tres errores anteriores, las soluciones son:

1. Cambiar en la cabecera el tipo de codificación, agregar en la etiqueta **HEAD**: **<meta charset='utf-8'>**
2. Para arreglar los elementos con atributos, simplemente hay que añadir una clase el correspondiente atributo CSS valido y el error desaparecerá, ya que W3C sugiere utilizar CSS y no etiquetas HTML.
3. La solución al problema de elementos no permitidos por ser obsoletos es cambiarla completamente, eliminarlos de HTML.

Bibliografía

1. https://es.wikipedia.org/wiki/JavaScript
2. https://es.wikipedia.org/wiki/Entorno_de_desarrollo_integrado
3. https://es.wikipedia.org/wiki/Computaci%C3%B3n_en_la_nube
4. https://bbvaopen4u.com/es/actualidad/desarrollo-en-la-nube-ventajas-y-plataformas-para-programadores
5. https://msdn.microsoft.com/en-us/library/z41h7fat.aspx
6. https://eduardosojo.com/2011/07/19/usuando-nuget-que-es-nuget/
7. https://codepen.io/iiCe89/pen/gMYVYQ
8. http://sistemasyinternet.blogspot.com.es/2011/06/que-es-w3c.html
9. https://www.loginradius.com/engineering/need-validate-site-w3c/
10. https://validator.w3.org/nu/?doc=https%3A%2F%2Fwww.google.com%2F%3Fgfe_rd%3Dcr%26dcr%3D0%26ei%3D5QjAWaLmE_LAXueRnpAE