Html5, Canvas, Css y JavaScript

Autor 1: Sara Valentina Restrepo Ramírez

*Risaralda, Universidad Tecnológica de Pereira, Pereira, Colombia*

Correo-e: sara.restrepo1@utp.edu.co

***Resumen*—En el siguiente documento veremos un poco de HTML5, Canvas, css y JavaScript, veremos ejemplos.**

***Palabras clave—* JavaScript, programa, sintaxis, html, html5, css**

***Abstract*—** **In the following document we will see a bit of HTML5, Canvas, css and JavaScript, we will see examples.**

INTRODUCCIÓN

Veremos ejemplos y que es cada lenguaje

1. CONTENIDO

HTML5

HTML5 es un lenguaje markup (de hecho, las siglas de HTML significan Hyper Text Markup Language) usado para estructurar y presentar el contenido para la web. Es uno de los aspectos fundamentales para el funcionamiento de los sitios, pero no es el primero. Es de hecho la quinta revisión del estándar que fue creado en 1990. A fines del año pasado, la W3C la recomendó para transformarse en el estándar a ser usado en el desarrollo de proyectos venideros. Por así decirlo, qué es HTML5 está relacionado también con la entrada en decadencia del viejo estándar HTML 4, que se combinaba con otros lenguajes para producir los sitios que podemos ver hoy en día. Con HTML5, tenemos otras posibilidades para explotar usando menos recursos. Con HTML5, también entra en desuso el formato XHTML, dado que ya no sería necesaria su implementación.

HTML5. Se trata de un sistema para formatear el layout de nuestras páginas, así como hacer algunos ajustes a su aspecto. Con HTML5, los navegadores como Firefox, Chrome, Explorer, Safari y más pueden saber cómo mostrar una determinada página web, saber dónde están los elementos, dónde poner las imágenes, dónde ubicar el texto. En este sentido, el HTML5 no se diferencia demasiado de su predecesor, un lenguaje del cual hablamos hace algunos meses en nuestra guía básica de HTML. La diferencia principal, sin embargo, es el nivel de sofisticación del código que podremos construir usando HTML5.

Novedades de HTML5

En términos de Markup, el HTML5 introduce algunos elementos que hacen que se aggiorne a los tiempos que corren. Así, muchas de las novedades están relacionadas con la forma de construir websites que se tiene en la actualidad. Una de las más importantes novedades está relacionada con la inserción de multimedia en los sitios web, que ahora contarán con etiquetas HTML especiales para poder ser incluidos. Por otro lado, algunos aspectos de diseño también son incluidos en el lenguaje, así como también algunos detalles de navegación. Veremos todo esto en algunas líneas.

Con el uso de HTML5, se puede reducir la dependencia de los plug-ins que tenemos que tener instalados para poder ver una determinada web. Caso emblemático, el de Adobe Flash, que se ve claramente perjudicado por la instauración de este estándar. Por otro lado, fue un avance importante para dispositivos que de forma nativa no soportaban Flash, y que no soportaban tampoco plug-ins necesarios para hacerlo. Otro caso emblemático, el del iPhone. Pero, además, con HTML5 se amplía el horizonte del desarrollo de aplicaciones que pueden ser usadas en una multiplicidad de dispositivos.

Gracias a HTML5, los usuarios pueden acceder a sitios web de manera offline, sin estar conectados a internet. Se suma también la funcionalidad de drag and drop, y también la edición online de documentos ampliamente popularizada por Google Docs. La geolocalización es uno de sus puntos fuertes, pero por otro lado, las etiquetas diseñadas especialmente para el audio y el video ahorran la necesidad de tener que tener un plug-in de Flash y, al mismo tiempo, asestan un golpe mortal al producto de Adobe, que cada vez se está usando menos. Sin embargo, es importante destacar que Flash sigue siendo utilizado y HTML5 todavía no hizo el “salto grande”, aunque está en camino.

Ejemplo

En este ejemplo vamos a ver cómo se maqueta una página en HTML5, usando las nuevas etiquetas que nos proporciona HTML5 como son:

<header>…</header>

<section>…</section>

<aside>…</aside>

<figure>…</figure>

<figcaption>…</figcaption>

<footer>…</footer>

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>

<title>EJEMPLO MAQUETACIÓN EN HTML5</title>

<link rel="stylesheet" href="css/estilo.css">

</head>

<body>

<!-- Cabecera -->

<header>

<img src="Media/ejemplo-logotipo.gif" width="189" height="57" alt="logo" />

<h1>EJEMPLO MAQUETACIÓN EN HTML5 - Maqueta #1</h1>

</header>

<!-- Contenido -->

<section>

<figure>

<img src="Media/Koala.jpg" width="200" height="200" alt="foto" />

<figcaption>Figura: descipción foto</figcaption>

</figure>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean ornare sapien ac nibh vulputate feugiat. Morbi euismod velit sit amet libero volutpat ultrices in quis sem. Sed sit amet placerat sem. Curabitur vitae lectus nec purus accumsan faucibus. Donec vitae volutpat tortor. Suspendisse tempor rutrum urna, in consequat est tincidunt vitae. Cras et ligula at felis placerat tempor. Fusce elementum metus non justo luctus iaculis.</p>

<p>Donec ligula arcu, sagittis lacinia nibh et, consequat gravida turpis. Aenean nec scelerisque felis. Integer nec odio sed lorem mattis rhoncus quis quis nibh. Praesent non fringilla nisi. Duis porta placerat mattis. Aenean in mollis odio, at egestas massa. Quisque in lectus magna. Vestibulum vel faucibus massa, vel varius tortor. Maecenas et laoreet neque, ut semper neque. Suspendisse aliquet, nisl vel pharetra ultricies, elit felis molestie sem, ac scelerisque turpis tortor non lacus.</p>

</section>

<!-- Contenido relacionado-->

<aside>

<p>Contenido Relacionado</p>

</aside>

<!-- Pie de página -->

<footer>

<a href="http://www.ejemplocodigo.com">www.ejemplocodigo.com</a>

</footer>

</body>

</html>

La nueva sintaxis HTML5

En HTML5 destaca sobre todo la simplicidad y la permisividad.

El doctype

La primera línea de un documento corresponde a la DTD (Document Type Declaration) o declaración de tipo de documento y sirve para indicar que versión de lenguaje HTML se ha usado al escribirlo.

En HTML5 esta declaración no puede ser más sencilla:

<!DOCTYPE html>

1

<!DOCTYPE html>

Y ya está, no necesitas nada más.

El documento HTML

En una página web, el elemento <html> indica el inicio del contenido HTML. En HTML sólo necesitas indicar el lenguaje de la página. Por ejemplo:

<html lang=es>

1

<html lang=es>

La codificación de los caracteres

Es frecuente indicar qué codificación de caracteres se ha usado en una página web. Con HTML5 se simplifica con lo indispensable para que los navegadores puedan gestionarla correctamente.

<meta charset="UTF-8" />

1

<meta charset="UTF-8" />

Los scripts

Una vez más, se ha simplificado la declaración de scripts de la siguiente forma:

<script src="unscript.js"></script>

1

<script src="unscript.js"></script>

Esto es debido a que ahora se da por sentado que los scripts estan escritos en Javascript.

Los estilos CSS

Seguimos con la simplificación. Ahora para la declaración de los estilos te bastará con:

<link href="tuhojadeestilos.css" rel="stylesheet" />

1

<link href="tuhojadeestilos.css" rel="stylesheet" />

La sintaxis de los elementos

Las comillas en HTML5 pasan a ser facultativas para los valores de los atributos. Puedes escribirlos con comillas dobles, comillas simples o sin ellas. Estas tres sintaxis serían correctas:

<div class="tuestilo">

<div class='tuestilo'>

<div class=tuestilo>

1

2

3

<div class="tuestilo">

<div class='tuestilo'>

<div class=tuestilo>

En los elementos HTML con una sola etiqueta de apertura como por ejemplo br o img ya no es necesario indicar el cierre del elemento usando /. Por ejemplo:

<img src="tuimagen.jpg">

1

<img src="tuimagen.jpg">

HTML5 permite el uso de mayúsculas y minúsculas indistintamente en la sintaxis de los elementos y atributos. De este modo, podrías declarar un script así:

<SCRIPT src="TuJavascript.js"></script>

1

<SCRIPT src="TuJavascript.js"></script>

Como puedes ver, HTML5 es muy permisivo a la hora de escribir código. Ofrece una gran flexibilidad y muy pocas restricciones. Lo cual, también puede ser un problema. Imagina un código en el que hayan trabajado dos o tres desarrolladores, con la permisividad de HTML5, si cada uno ha escrito el código a su manera, lo que tu veas al final, puede resultar un auténtico caos.

Es por eso, que te recomiendo seguir unas sencillas pautas que te harán la vida más fácil a ti y a otros desarrolladores que trabajen sobre tu código.

Usa sólo minúsculas

Usa siempre comillas para los valores de los atributos

Cierra con / las etiquetas con una sola etiqueta de apertura.

Cierra los elementos, aunque su etiqueta sea facultativa.

Utiliza el sangrado para que el código sea más legible.

Los elementos de la estructura en HTML5

Con la llegada de HTML5 han llegado nuevos elementos de estructura para mejorar la semántica. Esto es, dotar a estos elementos de un significado.

¿Cuáles son esos elementos? No te preocupes, los vemos a continuación.

El elemento <header>

El nuevo elemento <header> se utiliza para definir una zona de visualización para las cabeceras. Puedes definir cabeceras tanto a nivel de página como de una zona determinada (un artículo, un menú, etc…)

De este modo, puedes tener una cabecera para toda la página que sería la típica cabecera en la parte alta de la página con un logo, un menu, etc… O puedes tener una cabecera para una zona determinada como por ejemplo un artículo, en el que la cabecera podría incluir el título, el autor y la fecha.

El elemento <footer>

Funciona de un modo muy parecido al <header>. Del mismo modo, podrás determinar un <footer> para toda la página o utilizarlo en diferentes secciones de la web como por ejemplo un sidebar o un artículo.

A nivel de página sería la típica zona en la parte baja de la web, donde se suelen incluir datos de contacto, enlaces, etc… A nivel de zonas, y siguiendo con el ejemplo anterior, en el <footer> de un artículo podrías incluir por ejemplo la categoría a la que pertenece ese artículo o el autor si no lo has puesto en el <header>

El elemento <nav>

Como su nombre ya deja intuir, el elemento <nav> sirve para definir una zona de navegación con vínculos. Lo que vendría a ser un menú de toda la vida. Este elemento está pensado para la navegación principal de la web y para definir vínculos entre las diferentes páginas.

El elemento <section>

La función del elemento <section> es agrupar elementos relacionados entre sí. De este modo, podrás, por ejemplo, agrupar dentro de un mismo elemento un contenido con su título y su pie de página.

El elemento <article>

El <article> sirve para definir un contenido autónomo e independiente, que pueda ser usado en otra parte de la web sin que por ello pierda su significado.

A modo de ejemplo, usarías el elemento <artícle> para un artículo de un blog, como este que estás leyendo ahora mismo, o una noticia de un periódico o incluso un comentario de un blog.

El elemento <aside>

La función del elemento <aside> es mostrar un contenido relacionado al contenido al cual está vinculado. Puede tratarse de sidebars, zonas de widgets, complementos sobre un artículo, etc…

En la mayoría de los casos un <aside> se ubicará en un lateral del contenido con el que están relacionados. Pero recuerda que no tiene por qué ser así. La finalidad de las nuevas etiquetas es dotar de semántica al contenido de una web, para la ubicación de los elementos debes usar CSS.

Los <div>

Por supuesto, la llegada de HTML5, no significa que tengas que erradicar las cajas <div>. Todavía puedes usarlas y aún tienen su utilidad.

Si deseas ampliar esta información o profundizar más en alguno de los elementos, te recomiendo que leas la recomendación del 28 de octubre de 2014 de la W3C.

Y por supuesto, no te preocupes si todo esto te ha sonado un poco a chino.

Ahora veremos algunos ejemplos de utilización de estos elementos estructurales para que entiendas un poco mejor el uso de cada uno de ellos.

Ejemplos de estructura en HTML5

Estructura simple

Elementos estructura simple HTML5

En este primer ejemplo tendrías un <header> con los elementos del encabezado de la página como el logo o el eslogan de la web.

En la sección <nav> tendrías el menú con los vínculos que permiten navegar por la web.

Dos secciones <article> que evidentemente serán los artículos de la web.

Y terminas con un <footer> que podría contener por ejemplo el copyright y los avisos legales con sus vínculos.

Estructura un poco más compleja

Elementos estructura más elaborada HTML5

En este segundo ejemplo, tendríamos un <header> como en el anterior.

Debajo del <header> aparece el primer cambio. Un <nav> horizontal que contendría el menú principal de la web con los vínculos a las diferentes páginas.

A la izquierda, otro elemento <nav> con el menú secundario que contendría vínculos directamente relacionados con el contenido de esa página.

El contenido principal, en el centro, con dos elementos <section> para distinguir dos tipos de contenido diferentes. Cada <section> contiene un <header> para el título, un <article> con el contenido textual y un <footer> con la información de la categoría y el autor, por ejemplo.

A la derecha, un elemento <aside> que podría mostrar otra información relacionada con el contenido como por ejemplo contenidos relacionados.

Termina la estructura nuevamente con un <footer> típico donde podría aparecer, como no, el copyright y los avisos legales con sus vínculos.

CANVAS

Canvas (o "lienzo" traducido al español) es un elemento HTML incorporado en HTML5 que permite la generación de gráficos dinámicamente por medio del scripting. Entre otras cosas, permite la renderización interpretada dinámica de gráficos 2D y mapas de bits, así como animaciones con estos gráficos. Se trata de un modelo de procedimiento de bajo nivel, que actualiza un mapa de bits y no tiene una gráfica de escena integrada.

Historia

Canvas fue introducido inicialmente por Apple para su uso dentro de su propio componente de Mac OS X Surgido en 2004, para empujar aplicaciones como widgets de Dashboard y el navegador Safari. Más tarde, en 2005, se adoptó en la versión 1.8 de los navegadores Gecko y Opera en 2006. Fue estandarizado por el Grupo de Trabajo de Tecnologías de Aplicación de hipertexto Web (WHATWG por sus siglas en inglés) sobre las nuevas especificaciones propuestas para tecnologías web de última generación.

Utilización

El Canvas consiste en una región dibujable definida en el código HTML con atributos de altura y ancho. El código JavaScript puede acceder a la zona a través de un conjunto completo de funciones similares a las de otras APIs comunes de dibujo 2D, permitiendo así que los gráficos sean generados dinámicamente. Algunos de los usos previstos incluyen construcción de gráficos, animaciones, juegos, y la composición de imágenes.

Ejemplo

El siguiente código crea un elemento canvas en una página HTML:

<canvas id="example" width="200" height="200">

Este texto se muestra si su navegador no soporta el lienzo (canvas) de HTML5.

</canvas>

Utilizando JavaScript, usted puede dibujar en el lienzo (canvas):

var example = document.getElementById('example');

var context = example.getContext('2d');

context.fillStyle = 'red';

context.fillRect(30, 30, 100, 100);

Este código dibuja un rectángulo rojo en la pantalla.

Soporte

El elemento se apoya en las versiones actuales de Mozilla Firefox, Google Chrome, Internet Explorer, Safari, Konqueror y Opera. Las versiones anteriores de Internet Explorer, como la versión 8 y anteriores no son compatibles con la lona, sin embargo, plugins desarrollados por Google y Mozilla lo hacen posible.

CSS

Antes de comenzar, debes tener claro un concepto clave: una página web es realmente un documento de texto. En dicho documento se escribe código HTML, con el que se que crea el contenido de una web. Por otro lado, existe el código CSS, que unido al código HTML permite darle forma, color, posición (y otras características visuales) a una página.

En resumen, se trata de un idioma como podría ser el inglés o el alemán, que los navegadores web como Chrome o Firefox conocen y pueden entender. Nuestro objetivo como diseñadores y programadores web es precisamente ese: aprender el idioma.

Las siglas CSS (Cascading Style Sheets) significan «Hojas de estilo en cascada» y parten de un concepto simple pero muy potente: aplicar estilos (colores, formas, márgenes, etc...) a uno o varios documentos (generalmente documentos HTML, páginas webs) de forma masiva.

Se le denomina estilos en cascada porque se aplican de arriba a abajo (siguiendo un patrón denominado herencia que trataremos más adelante) y en el caso de existir ambigüedad, se siguen una serie de normas para resolverla.

Para implementar el CSS en alguna página HTML, debemos escribirlo dentro del elemento style:

<style type=”text/css”>

//Estilo CSS aquí

</style>

o mediante el atributo class de los elementos HTML:

<p class=”estilo-css”>Texto de prueba</p>

Ejemplos de CSS

Colocar una imagen de fondo:

<html>

<head>

<style type=”text/css”>

body {background-image: url(‘TU\_IMAGEN.jpg’)}

</style>

</head>

<body>

</body>

</html>

Con este extracto de código añadimos una imagen al elemento body (cuerpo de la página), por medio de la URL de origen de la imagen, la cual aparecerá las veces que usemos la etiqueta <body> en todo el desarrollo web.

Cambiar el color de varios elementos de la página:

<html>

<head>

<style type=”text/css”>

body {background-color: yellow}

h1 {background-color: #00ff00}

h2 {background-color: transparent}

p {

background-color: rgb(250,0,255)

}

</style>

</head>

<body>

<h1>Título 1</h1>

<h2>Título 2</h2>

<p>Texto de ejemplo.</p>

</body>

</html>

JavaScript

JavaScript (abreviado comúnmente JS) es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos,3​ basado en prototipos, imperativo, débilmente tipado y dinámico.

Se utiliza principalmente en su forma del lado del cliente (client-side), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas4​ y JavaScript del lado del servidor (Server-side JavaScript o SSJS). Su uso en aplicaciones externas a la web, por ejemplo en documentos PDF, aplicaciones de escritorio (mayoritariamente widgets) es también significativo.

Desde 2012, todos los navegadores modernos soportan completamente ECMAScript 5.1, una versión de javascript. Los navegadores más antiguos soportan por lo menos ECMAScript 3. La sexta edición se liberó en julio de 2015.5​

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.

Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Document Object Model (DOM).

Tradicionalmente se venía utilizando en páginas web HTML para realizar operaciones y únicamente en el marco de la aplicación cliente, sin acceso a funciones del servidor. Actualmente es ampliamente utilizado para enviar y recibir información del servidor junto con ayuda de otras tecnologías como AJAX. JavaScript se interpreta en el agente de usuario al mismo tiempo que las sentencias van descargándose junto con el código HTML.

Desde el lanzamiento en junio de 1997 del estándar ECMAScript 1, han existido las versiones 2, 3 y 5, que es la más usada actualmente (la 4 se abandonó6​). En junio de 2015 se cerró y publicó la versión ECMAScript 6.7​.

Historia

Nacimiento de JavaScript

JavaScript fue desarrollado originalmente por Brendan Eich de Netscape con el nombre de Mocha, el cual fue renombrado posteriormente a LiveScript, para finalmente quedar como JavaScript. El cambio de nombre coincidió aproximadamente con el momento en que Netscape agregó compatibilidad con la tecnología Java en su navegador web Netscape Navigator en la versión 2.002 en diciembre de 1995. La denominación produjo confusión, dando la impresión de que el lenguaje es una prolongación de Java, y se ha caracterizado por muchos como una estrategia de mercadotecnia de Netscape para obtener prestigio e innovar en el ámbito de los nuevos lenguajes de programación web.8​9​

«JAVASCRIPT» es una marca registrada de Oracle Corporation.10​ Es usada con licencia por los productos creados por Netscape Communications y entidades actuales como la Fundación Mozilla.11​12​

Microsoft dio como nombre a su dialecto de JavaScript «JScript», para evitar problemas relacionadas con la marca. JScript fue adoptado en la versión 3.0 de Internet Explorer, liberado en agosto de 1996, e incluyó compatibilidad con el Efecto 2000 con las funciones de fecha, una diferencia de los que se basaban en ese momento. Los dialectos pueden parecer tan similares que los términos «JavaScript» y «JScript» a menudo se utilizan indistintamente, pero la especificación de JScript es incompatible con la de ECMA en muchos aspectos.

Para evitar estas incompatibilidades, el World Wide Web Consortium diseñó el estándar Document Object Model (DOM, o Modelo de Objetos del Documento en español), que incorporan Konqueror, las versiones 6 de Internet Explorer y Netscape Navigator, Opera la versión 7, Mozilla Application Suite y Mozilla Firefox desde su primera versión.

En 1997 los autores propusieron13​ JavaScript para que fuera adoptado como estándar de la European Computer Manufacturers 'Association ECMA, que a pesar de su nombre no es europeo sino internacional, con sede en Ginebra. En junio de 1997 fue adoptado como un estándar ECMA, con el nombre de ECMAScript. Poco después también como un estándar ISO.

Ejemplos sencillos

Las variables en JavaScript se definen usando la palabra clave var:38​

var x; // define la variable x, aunque no tiene ningún valor asignado por defecto

var y = 2; // define la variable y y le asigna el valor 2 a ella

A considerar los comentarios en el ejemplo de arriba, los cuales van precedidos con 2 barras diagonales.

No existen funcionalidades para I/O incluidas en el lenguaje; el entorno de ejecución ya lo proporciona. La especificación ECMAScript en su edición 5.1 hace mención:39​

... en efecto, no existen provisiones en esta especificación para entrada de datos externos o salida para resultados computados.

Sin embargo, la mayoría de los entornos de ejecución tiene un objeto40​ llamado console que puede ser usado para imprimir por el flujo de salida de la consola de depuración. He aquí un simple programa que imprime “Hello world!”:

console.log("Hello world!");

Una función recursiva:

function factorial(n) {

if (n === 0) {

return 1;

}

return n \* factorial(n - 1);

}

Ejemplos de función anónima (o función lambda) y una clausura:

var displayClosure = function() {

var count = 0;

return function () {

return ++count;

};

}

var inc = displayClosure();

inc(); // devuelve 1

inc(); // devuelve 2

inc(); // devuelve 3

Las expresiones con invocación automática permiten a las funciones pasarle variables por parámetro dentro de sus propias clausuras.

var v;

v = 1;

var getValue = (function(v) {

return function() {return v;};

}(v));

v = 2;

getValue(); // 1

Ejemplos más avanzados

El siguiente código muestra varias características de JavaScript.

/\* Busca el mínimo común múltiplo (MCM) de dos números \*/

function LCMCalculator(x, y) { // función constructora

var checkInt = function (x) { // función interior

if (x % 1 !== 0) {

throw new TypeError(x + " no es un entero"); // lanza una excepción

}

return x;

};

this.a = checkInt(x) // puntos y coma son opcionales

this.b = checkInt(y);

}

// El prototipo de las instancias de objeto creados por el constructor es el de la propiedad “prototype” del constructor.

LCMCalculator.prototype = { // objeto definido como literal

constructor: LCMCalculator, // cuando reasignamos un prototipo, establecemos correctamente su propiedad constructor

gcd: function () { // método que calcula el máximo común divisor

// Algoritmo de Euclides:

var a = Math.abs(this.a), b = Math.abs(this.b), t;

if (a < b) {

// intercambiamos variables

t = b;

b = a;

a = t;

}

while (b !== 0) {

t = b;

b = a % b;

a = t;

}

// Solo necesitamos calcular el MCD una vez, por lo tanto 'redefinimos' este método.

// (Realmente no es una redefinición—está definida en la propia instancia, por lo tanto

// this.gcd se refiere a esta 'redefinición' en vez de a LCMCalculator.prototype.gcd).

// Además, 'gcd' === "gcd", this['gcd'] === this.gcd

this['gcd'] = function () {

return a;

};

return a;

},

// Los nombres de las propiedades del objeto pueden ser especificados con cadenas delimitadas con comillas simples (') o dobles (“).

"lcm" : function () {

// Los nombres de las variables no colisionan con las propiedades del objeto. Por ejemplo: |lcm| no es |this.lcm|.

// No usar |this.a \* this.b| para evitar problemas con cálculos en coma flotante.

var lcm = this.a / this.gcd() \* this.b;

// Sólo necesitamos calcular MCM una vez, por lo tanto "redefinimos" este método.

this.lcm = function () {

return lcm;

};

return lcm;

},

toString: function () {

return "LCMCalculator: a = " + this.a + ", b = " + this.b;

}

};

// Definimos una función genérica para imprimir un resultado; esta implementación solo funciona en los navegadores web

function output(x) {

document.body.appendChild(document.createTextNode(x));

document.body.appendChild(document.createElement('br'));

}

// Nota: Los métodos.map() y.forEach() del prototipo Array están definidos en JavaScript 1.6.

// Estos métodos son usados aquí para demostrar la naturaleza funcional inherente del lenguaje.

[[25, 55], [21, 56], [22, 58], [28, 56]].map(function (pair) { // construcción literal de un Array + función de mapeo.

return new LCMCalculator(pair[0], pair[1]);

}).sort(function (a, b) { // ordenamos la colección por medio de esta función

return a.lcm() - b.lcm();

}).forEach(function (obj) {

output(obj + ", gcd = " + obj.gcd() + ", lcm = " + obj.lcm());

});

El siguiente ejemplo muestra la salida que debería ser mostrada en la ventana de un navegador.

LCMCalculator: a = 28, b = 56, gcd = 28, lcm = 56

LCMCalculator: a = 21, b = 56, gcd = 7, lcm = 168

LCMCalculator: a = 25, b = 55, gcd = 5, lcm = 275

LCMCalculator: a = 22, b = 58, gcd = 2, lcm = 638

REFERENCIAS

<https://webdesdecero.com/tutorial-html5-la-sintaxis-y-la-estructura/>

<https://es.wikipedia.org/wiki/Canvas_(HTML)>

<https://lenguajecss.com/p/css/introduccion/que-es-css>

<https://ejemplos.net/5-ejemplos-de-css/>

<https://es.wikipedia.org/wiki/JavaScript>