

JQuery: [jquery.com](http://jquery.com)

```
<script type="text/javascript" src="jquery.js"></script>
```

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
```

```
$(document).ready( function() { ... } ); //accedemos al DOM y cuando todo el código haya cargado ejecuta lo que hay dentro de esta función.
```

*\*\* En JavaScript: **document.addEventListener("load", function load() { ... } );***

De esta forma podemos poner el script dentro de la etiqueta <head> de nuestro documento y no al final.

La versión abreviada de la instrucción anterior es:

```
$(function() { ... } ); //versión abreviada
```

## Función \$()

La **función más usada en jQuery es \$()**. Esta función se utiliza para seleccionar un elemento del árbol DOM. Recibe como parámetro un selector o el nombre de una etiqueta HTML.

La sintaxis es la siguiente: `$(selector).accion();`

- **\$** – Indicamos que vamos a acceder a jQuery.
- **selector** – Indicamos el selector o etiqueta HTML con el que queremos trabajar.
- **accion()** – Lo que queremos hacer con el elemento seleccionado.

## Selección de elementos del DOM

Como se ha comentado en el punto anterior, la función `$()` recibe como parámetro un selector o una etiqueta HTML. En la página oficial de jQuery puedes ver todos los selectores disponibles: [api.jquery.com/category/selectors](http://api.jquery.com/category/selectors). Los más utilizados son:

- **id:** `$("#ID")`
- **Etiqueta:** `$("a"), $("li")`
- **Clase:** `$(".miClase"); $("li.miClase"), $("div.miClase")`
- **Atributo:** `$("a[rel]")`
- **Acceso por un valor para un atributo:** `$("input[@type=radio]"), $("input[name=username]")`
- **Selectores:** `$("p a"), $("*")`
- **Selección de varios elementos separados por comas:** `$("#miParrafo, a")`
- **Pseudo-selectores:** `$("p[img]");` //selección de todos los párrafos que contienen una imagen.

## Eventos

Para utilizar los eventos de jQuery debemos **seleccionar el elemento**, indicar el **tipo de evento** y la **función que determine su comportamiento**. Por ejemplo:

```
$("p").click(function() { alert( 'Hola jQuery!' ); });
```

En la página oficial de jQuery puedes ver la lista completa de eventos disponibles: [api.jquery.com/category/events](https://api.jquery.com/category/events).

Los más utilizados son:

### Eventos del ratón

- **.click()** : pulsar una vez con el puntero sobre un elemento:

```
$(".click").click(function() {  
    alert("click sobre un elemento");  
});
```

- **.dblclick()** : pulsar dos veces seguidas con el puntero sobre un elemento:

```
$(".dblclick").dblclick(function() {  
    alert("doble click sobre un elemento");  
});
```

- **.mouseenter()** : el puntero se sitúa encima de un elemento:

```
$(".mouseenter").mouseenter(function() {  
    $(this).css("color", "red");  
});
```

- **.mouseleave()** : el puntero sale de un elemento:

```
$("#mouseleave").mouseleave(function() {  
    $(this).css("color", "blue");  
});
```

- **.hover()**: admite dos funciones, que se definen separadas por una coma. La primera de ellas se produce cuando el puntero entra sobre el elemento, y la segunda cuando sale de él:

```
$("#hover").hover(function() {  
    $(this).css("color", "yellow");  
},  
function() {  
    $(this).css("color", "green");  
});
```

- **.mousedown()** : pulsar el botón del ratón, independientemente de si se suelta o no. Válido para el botón izquierdo y para el derecho.

```
$("#mousedown").mousedown(function() {  
    $(this).html("<b>mousedown</b>")  
});
```

- **.mouseup()** : soltar un botón del ratón después de hacer click. El evento tiene lugar cuando se suelta el botón.

```
$("#mouseup").mouseup(function() {
```

```
$(this).html("<b>mouseup</b>")
});
```

## Eventos del teclado

- **.keydown()** : El evento se produce en el momento que se presiona una tecla, independientemente de si se libera o se mantiene la presión. Se produce una única vez en el momento exacto de la presión.
- **.keypress()** : Se produce al tener pulsada una tecla. Si se mantiene pulsada la tecla, el evento se produce varias veces.
- **.keyup()** : El evento se produce en el momento de dejar de presionar una tecla que teníamos pulsada.

Los eventos de teclado se suelen aplicar al documento (document), y no a un elemento concreto. Veamos un ejemplo:

```
$(document).keydown(function(){
    $("#teclado").html("tecla pulsada");
});
$(document).keyup(function(){
    $("#teclado").html("tecla sin pulsar");
});
```

## Eventos de ventana

- **.scroll()**. Este evento se atiende cuando sobre un elemento que tiene barras de desplazamiento se mueve una de ellas.
- **.resize()**. El evento se lanza cuando a un elemento tipo ventana se le cambia el tamaño.

## Efectos

[api.jquery.com/category/effects/](http://api.jquery.com/category/effects/)

### Efectos básicos

- **.show** Muestra el elemento seleccionado.
- **.hide** Oculta el elemento seleccionado.
- **.fadeIn** Cambia la opacidad del elemento seleccionado al 100%.
- **.fadeOut** Cambia la opacidad del elemento seleccionado al 0%.
- **.slideUp** Oculta el elemento seleccionado con un movimiento de deslizamiento vertical.
- **.slideDown** Muestra el elemento seleccionado con un movimiento de deslizamiento vertical.
- **.slideToggle** Muestra u oculta el elemento seleccionado con un movimiento de deslizamiento vertical.

### Ejemplo

- `$('.caja').hide();` //oculta el elemento con la clase caja

Además, podemos cambiar la duración de los efectos:

- `$('.caja').fadeIn(300);` //La caja se va desvaneciendo durante 3 segundos

También podemos ejecutar una función cuando el efecto finalice:

**SINTAXIS:** \$(selector).efecto(velocidad, callback); //callback se ejecuta después de que se haga el efecto

```
$('.caja').hide(3000, function(){ //tarda 3 segundos en desaparecer y luego se muestra lo que
hay en callback
    alert('Caja oculta');
});
```

## Otros efectos

- **bind() y unbind():** Este método permite asociar a un elemento un número ilimitado de eventos, todo de una vez. Gracias a este método se facilita la asignación de eventos a los elementos, haciendo un código más legible y una ejecución más óptima.

```
$( "p" ).bind( "click mouseenter mouseleave", function( e ){  
    $( this ).css( "color", "rgb(0, 0, 255)" );  
})
```

- **.delay( duración [, NombreCola] )**

```
$( document ).ready( function(){  
    $( "#boton" ).click( function() {  
        $( ".caja1" ).fadeIn( 100 );  
        $( ".caja1" ).delay( 800 );  
        $( ".caja1" ).fadeOut( 800 );  
    });  
});
```

- **.animate** (propiedades, duración, callback)

[api.jquery.com/animate/#animation-properties](http://api.jquery.com/animate/#animation-properties).

## Manipulación de CSS

[api.jquery.com/category/manipulation/class-attribute](http://api.jquery.com/category/manipulation/class-attribute).

Los más destacados son:

- **.addClass():** añade una o más clases a los elementos seleccionados.
- **.removeClass():** elimina uno o más clases a los elementos seleccionados.
- **.toggleClass():** cambia entre la adición/eliminación de las clases de los elementos seleccionados.
- **.css():** cambia las propiedades de un elemento.

```
$( "p" ).css( "color" );  
$( "p" ).css( "color", "red" );  
$( "p" ).css( { "color" : "red", "background" : "blue", "font-weight" : "bold" } );
```

## Librerías jQuery

Podemos enriquecer nuestras interfaces mediante diferentes librerías. Veamos algunas de ellas:

- [jqueryui.com](http://jqueryui.com)
- [jquerymobile.com](http://jquerymobile.com)
- [jqueryvalidation.org](http://jqueryvalidation.org)

## Ejercicios

- En base al contenido anterior hay que:
- Crear una agenda que permita ordenar la lista de registros
- Ordenar y clasificar imágenes de alimentos según sean: verduras, frutas, carne
- Crear un puzzle de 9 piezas: Que sale un aviso cuando se encuentre la solución
- Crear palabras de dos sílabas

\*Al no tener back implementado se trabajará con arrays.

## 1.1.2. Concepto de accesibilidad web

La **accesibilidad web** consiste en desarrollar **aplicaciones web** que puedan ser **utilizadas** por el mayor número de **usuarios con necesidades específicas**. Estas necesidades pueden ser debidas a **limitaciones derivadas del entorno o derivadas de problemas visuales, auditivos, motrices y neurológicos** (dislexia, trastornos de atención, falta de memoria...).

### Barreras derivadas del entorno

- **Navegadores antiguos:** Proporcionar alternativas cuando se utilizan elementos que no tienen soporte en tecnologías antiguas. Si usamos Javascript para mostrar un menú, este debe funcionar igualmente aunque la tecnología no esté disponible.
- **Navegadores de texto:** Incluir un equivalente textual para todos los elementos no textuales(imágenes, vídeos o sonidos)
- **Conexiones lentas:** minimizar el tiempo de carga de los elementos de la web.
- **Pantallas pequeñas o muy grandes:** diseño web responsive.
- **Monitores monocromos:** evitar funcionalidades que se interpretan por su color, etc.

### Barreras por tipo de perfil

- **1.2.1. Ceguera:**
  - Las imágenes sin texto alternativo no pueden ser leídas por los lectores de pantalla.
  - Las imágenes que incluyen gráficos que representan datos o textos insertados mediante imágenes tampoco son leídos por los lectores de pantalla.
  - Elementos multimedia sin descripción textual.
  - Tablas en las que el contenido es incomprensible cuando se lee de forma secuencial.
  - Falta de independencia de dispositivo, la web debe ser funcional cuando no se utilice ratón.
  - Formatos no accesibles de documentos que pueden dar problemas a los lectores de pantalla si no cumplen las normas de accesibilidad (por ejemplo en documentos pdf que no cumplen las normas).
- **1.2.2. Baja visión:**
  - Tamaño de letra con medidas absolutas que no permiten su cambio.
  - Maquetación desajustada al modificar los tamaños de la fuente y que complican la navegabilidad.
  - Poco contraste entre textos, fondos e imágenes.
  - Texto insertado mediante imágenes.
- **1.2.3. Daltonismo:**
  - Color para el resaltado de los textos sin utilizar otro formato adicional como la cursiva, la negrita o el subrayado.
  - Poco contraste entre textos, fondos e imágenes.
- **1.2.4. Auditivas:**
  - Falta de subtítulos o transcripciones de los contenidos.
  - Obligatoriedad del uso de micrófono sin posibilidad de desactivación.

- **1.2.5. Motrices:**
  - Elementos de interacción muy pequeños: botones, enlaces, etc.
  - Falta de independencia de dispositivo, la web debe ser funcional cuando no se utilice ratón.
- 1.2.6. Neurológicas o cognitivas (dislexia, trastornos de atención, falta de memoria...):
  - Tamaño de letra fijo que no se puede cambiar.
  - Elementos sonoros o visuales que no se pueden desactivar.
  - Falta de estructuración y organización del contenido que impide entenderlo correctamente.
  - Lenguaje muy enrevesado y frases muy complejas.
  - Destellos o parpadeos frecuentes que pueden provocar ataques de epilepsia.

Enlace de interés sobre soluciones técnicas:

- <http://accesibilidadweb.dlsi.ua.es/?menu=persona-discapacitada-web>

### 1.1.3. W3C y WCAG

#### W3C

Desarrolla estándares que aseguran el crecimiento futuro de la web y vela por conseguir webs disponibles para todo el mundo y desde cualquier dispositivo.

Desde W3C se establecen ciertas recomendaciones, normas, estándares y pautas para contribuir en el desarrollo de un estándar sobre accesibilidad web. Estas recomendaciones o pautas son lo que llamamos WCAG (Web Content Accessibility Guidelines o Pautas de Accesibilidad para el contenido Web).

#### WCAG

Pautas de Accesibilidad para el Contenido en la Web (WCAG)

Las Pautas de Accesibilidad para el Contenido Web (WCAG) cubren un amplio rango de recomendaciones para crear contenido Web más accesible. Tal y como hemos visto en los apartados anteriores, seguir estas pautas permite crear un contenido más accesible para un mayor número de personas con barreras visuales, auditivas, motrices, etc.. Además, seguir estas pautas puede, a menudo, ayudar a que el contenido web sea más usable.

- Las pautas de las WCAG se pueden consultar en esta web: <https://www.w3.org/TR/WCAG20/>
- La versión en castellano se puede consultar en esta web: <http://sidar.org/traducciones/wcag20/es/>

Las catorce pautas se redujeron a cuatro principios fundamentales en la versión 2.0:

- Perceptibilidad.
- Operatividad.
- Comprensibilidad.
- Robustez.

Herramientas para comprobar tu código

- <https://webaim.org/resources/contrastchecker/>

### 1.1.4. Niveles de prioridad y puntos de verificación




La **WCAG** ofrece varios **niveles de prioridades** a la hora de aplicar las pautas de accesibilidad. De esta forma podemos saber cuáles son las acciones más importantes que debemos realizar. Son tres niveles de prioridades:

- **Prioridad 1:** Nivel mínimo exigible y requisito esencial que se debe satisfacer. De otra forma, uno o más grupos de usuarios encontrarán imposible acceder a la información.
- **Prioridad 2:** Satisfacer este punto de verificación eliminará importantes barreras de acceso a la información Web.
- **Prioridad 3:** Satisfacer este punto de verificación mejorará la accesibilidad a la información web a usuarios que tengan dificultades para utilizar la interfaz.

Si se cumplen los niveles de prioridades se establecen varios niveles de conformidad que sirven de sello de accesibilidad.

- **Nivel A:** Prioridad 1.
- **Nivel AA:** Prioridades 1 y 2.
- **Nivel AAA:** Prioridades 1, 2 y 3.

Logos de conformidad

nivel A	Doble-A	Triple-A
		

## Nivel A:

Para lograr conformidad con el Nivel A (el mínimo), la página web satisface todos los Criterios de Conformidad del Nivel A, o proporciona una versión alternativa conforme.

## Nivel AA:

Para lograr conformidad con el Nivel AA, la página web satisface todos los Criterios de Conformidad de los Niveles A y AA, o se proporciona una versión alternativa conforme al Nivel AA.

## Nivel AAA:

Para lograr conformidad con el Nivel AAA, la página web satisface todos los Criterios de Conformidad de los Niveles A, AA y AAA, o proporciona una versión alternativa conforme al Nivel AAA.

## 1.1.5. Herramientas de evaluación, análisis y testeo de accesibilidad web

### Herramientas de evaluación, análisis y testeo de accesibilidad web:

- [Taw](#): Aplicación web que permite hacer un análisis de accesibilidad de forma gratuita tan solo introduciendo la URL.
- [Cynthia Says](#): Informe bastante completo sobre accesibilidad.
- [Color Contrast Checker](#): aplicación para comprobar el contraste entre fondo y texto.
- [Wave](#): Informe muy visual sobre la página a evaluar. Prestaciones parecidas a la extensión para Chrome creada por la misma empresa.
- [Tenon.io](#): nos permite analizar un sitio web o un trozo de código.
- [Textise](#): simulador online para ver una página web en modo “Sólo texto”.



- [Colour Contrast Analyser](#): aplicación para comprobar el contraste entre fondos y textos.
- [WAI HTML Table Linearizar Entry Form](#): aplicación para evaluar tablas.
- [Examinator](#): aplicación para evaluar una página.
- [W3C Markup Validator](#): Validador HTML.
- [CSS Validator](#): Validador CSSVALIDATOR.

Curso de interés

<https://www.udemy.com/course/aprende-accesibilidad-web-paso-a-paso/>

<https://www.udemy.com/course/analisis-ux/>

## 1.1.6. Usabilidad Web

La usabilidad en el diseño consiste en facilitar al usuario la utilización de una interfaz de la forma más fácil e intuitiva. Pero ¿cómo diseñamos aplicaciones web usables?

En primer lugar, debemos tener en consideración al usuario en todo momento, es lo que se entiende por “usuario en el centro”. De esta forma, los equipos de desarrollo, y más concretamente el equipo de diseño, debe comprobar continuamente si la modificación de un elemento repercute en la usabilidad de la aplicación web.

### 1. Reglas y principios de usabilidad

A la hora de diseñar una interfaz para un proyecto, es de especial importancia tener en consideración las reglas básicas de usabilidad. A continuación se muestran algunas de ellas.

#### 2. Regla de los dos segundos

Cuando un usuario debe esperar más de dos segundos para obtener una respuesta a una acción realizada en una aplicación web, es muy posible que se impaciente y abandone la página en busca de otra plataforma que le ofrezca mejores resultados. Lo ideal es que la aplicación sea de **carga rápida y evitar toda sobrecarga innecesaria**.

#### 3. Feedback de información

En el caso de disponer de procesos que tengan un tiempo de respuesta elevado, se debe **diseñar el feedback de información para el usuario**. Por ejemplo: en el proceso de envío de un formulario puede llegar a ser necesario utilizar una **animación de precarga o spinner** para mantener al usuario informado del estado del procesado de la información por parte de la plataforma.

#### 4. Eliminar cualquier funcionalidad que no suma valor real

Siempre debemos facilitar la interacción del usuario en nuestra plataforma y ofrecerle las funcionalidades necesarias acorde a la temática del sitio. Por este motivo **eliminaremos todos los elementos que no aporten valor real y justificado**. Por ejemplo: sería completamente innecesario incluir un widget del tiempo o un calendario en una tienda online de zapatos.

#### 5. Sí al espacio entre elementos

Tal y como ocurre con toda composición visual, el espacio es necesario. Los espacios bien utilizados dan a los usuarios tiempo para pensar y observar el espacio sin estar bombardeados de textos e imágenes. **La confusión**

**visual es muy perjudicial, y los proyectos que están muy cargados son tremendamente molestos.** El espacio puede ser de cualquier color de fondo.

## 6. Legibilidad de los textos

Los textos son la base de la mayoría de sitios web ya que lo más normal es transmitir la información mediante letras. Por este motivo, para mejorar la experiencia del usuario debemos prestar mucha atención en la legibilidad de los textos: **tipos y tamaño de letra, contraste entre texto y fondo, textos con una longitud adecuada, interlineado, etc.**

## 7. Coherencia y consistencia

La coherencia y consistencia en el diseño nos ayuda a crear interfaces con elementos relacionados de forma lógica y sin contradicciones. Por ejemplo: en una plataforma es esencial **establecer correctamente los colores que representan los errores y aciertos**, así como **utilizar justificadamente los símbolos e iconos** de los elementos. Por lo tanto, los colores, los tipos de fuente, la distribución de los contenidos, etc. deben ser homogéneos a lo largo de toda la aplicación.

## 8. La regla de los tres clics

El usuario debe poder acceder de forma sencilla a todo el contenido de una plataforma. De este modo, se considera que **el contenido que se encuentra a más de tres clics no es importante**. Es por ello que los contenidos más visitados o las funcionalidades más útiles deben situarse a tan solo un clic para conseguir interacciones lo más eficientes.

## 9. Manejar los errores

Una plataforma debe estar diseñada de forma que si hay una incidencia al acceder a una página, bien porque se haya eliminado o cambiado la url, aparezca la **redirección correspondiente o el mensaje de error informativo** (denominado "error 404" o "página no encontrada"). En esta página los usuarios deben poder ver el mensaje de error y volver a inicio o *home*. Además, esta "página no encontrada" también debe estar diseñada conforme a la estética de la [guía de estilo](#).

## 10. El principio del "número siete, más o menos dos"

Este principio está basado en un estudio llevado a cabo por el psicólogo George A. Miller en el que se descubrió que **la memoria a corto plazo trabajaba mejor cuando se empleaban conjuntos de siete (más o menos dos) datos**. Por ejemplo: si a los usuarios se les ofrece demasiadas opciones en un menú de navegación estarán confusos y no sabrán qué seleccionar. Por lo tanto, lo ideal será descartar más de 7 opciones.

## 11. Protección del trabajo de los usuarios

La protección del trabajo de los usuarios es algo prioritario, se debe **asegurar que el usuario nunca pierda el trabajo realizado** como consecuencia de un error. Por ejemplo: cuando un usuario está finalizando un proceso de compra y está rellenando todos sus datos, si hubiera algún error en el proceso se debería recuperar el carrito de compra y los datos que fueran necesarios.

## 12. Correspondencia entre los contenidos y el mundo real

El contenido de un sitio web debe estar escrito en el lenguaje de los usuarios con palabras, frases y conceptos familiares. Es decir, el contenido debe seguir las convenciones del mundo real y el diseñador debe ser capaz de **mostrar la información de forma natural y lógica**.

## 13. Curva de aprendizaje mínima

Los usuarios se encuentran más cómodos en sitios web que resultan fáciles de utilizar. En este sentido, debe ser tan sencillo utilizar la interfaz que la curva de aprendizaje sea mínima. Por ejemplo: se puede **aprovechar la asociación para facilitar el aprendizaje** de los diferentes procesos. En el caso de Apple, fueron los primeros en utilizar un slider para desbloquear la pantalla con el dedo y posteriormente lo utilizaron para todas las demás interacciones de la interfaz gráfica.

## 14. Navegación rápida entre secciones

Diseñar un sistema de navegación que facilite las tareas a realizar es esencial para que **el usuario se encuentre cómodo visitando todo el contenido** de nuestra plataforma. Así por ejemplo, se deben implementar correctamente las opciones de desplazamiento entre los diferentes productos o contenidos, agregar migas de pan que ayuden a volver a otros contenidos fácilmente, el menú principal siempre debe estar visible o utilizar adecuadamente los sistemas de paginado, entre otras opciones.

## 15. Simetría / asimetría

Jugando con la simetría y la asimetría podemos conseguir diferentes efectos. Así pues, mediante la simetría podemos conseguir **diseños equilibrados**. Un ejemplo de este diseño podría ser colocar 4 iconos, uno al lado del otro, del mismo tamaño y color. Por otro lado, la asimetría puede conseguir que el usuario **focalice su mirada justo hacia el punto que destaca fuera de una composición simétrica**. En el caso anterior, podríamos romper el formato y el equilibrio cambiando el tamaño y el color de uno de los iconos.

## Sitios de inspiración para el diseño web

- [Webdesign-inspiration.com](http://Webdesign-inspiration.com)
- [Dribbble.com](http://Dribbble.com)
- [Siteinspire.com](http://Siteinspire.com)
- [Thebestdesigns.com](http://Thebestdesigns.com)
- [Awwwards.com](http://Awwwards.com)
- [Themeforest.net](http://Themeforest.net)
- [Behance.net](http://Behance.net)
- [Canva.com](http://Canva.com)

## Posicionamiento web SEO (*Search Engine Optimization*)

Que tu empresa esté bien posicionada implica que el usuario pueda encontrarte fácilmente, puesto que con una buena **estrategia SEO** te encontrarás en las primeras páginas de búsqueda de buscadores como Google. Así pues, podemos definir el **posicionamiento en buscadores** como el resultado de una optimización de los motores de búsqueda.

Algunas pautas a seguir para un buen posicionamiento:

- Comprobar que tu sitio web cumple las directrices de calidad que requieren los buscadores.
- Detectar problemas de accesibilidad de tu sitio web que impidan a los buscadores rastrear e indexar todas las páginas y contenidos.
- Identificar oportunidades de mejora en el rendimiento de tu web tales como aumento de la velocidad de carga.
- Reconocer los obstáculos más frecuentes para resolverlos lo antes posible.
- Diseñar y gestionar una arquitectura de la información flexible y orientada al SEO.
- Identificar las palabras clave que más tráfico pueden atraer a tu web.
- Definir la estructura óptima de una página de cara a su posicionamiento.
- Identificar patrones y tendencias de búsqueda.
- Utilizar herramientas automatizadas que nos avisen de los problemas que surgen para así resolverlos a tiempo.

## Velocidades de carga

El factor más importante es el **tamaño del sitio**, teniendo en cuenta la cantidad de ficheros de los que dispone: hojas de estilo CSS, ficheros JavaScript, imágenes, etc.

Por otra parte, también debemos tener en cuenta los **recursos externos que se cargan** en la aplicación web, como por ejemplo *Analytics*, banners de publicidad o librerías externas como jQuery, Bootstrap o Angular y repositorios de fuentes o iconos, entre otros recursos.

## Técnicas WPO (Web Performance Optimization).

Como decíamos, las técnicas WPO nos van a permitir **acelerar la velocidad de carga** de un sitio web optimizando su rendimiento. Algunas de las técnicas más importantes son las siguientes:

- Reducir el número de peticiones al servidor siempre que sea posible. Debemos tener en cuenta que por cada fichero disponible en una web, el navegador debe establecer una conexión diferente.
- Minimizar las hojas de estilo, el código HTML y el código JavaScript: eliminar comentarios innecesarios y espacios en blanco.
- Optimizar las imágenes: reducir su tamaño y adecuarlas para que el navegador no las redimensione.
- Evitar incluir código CSS y JavaScript de forma *inline*: traspasar las etiquetas *style* a las hojas de estilo y las etiquetas *script* a los ficheros JavaScript.
- Usar módulos de compresión en el servidor.
- Elegir un buen hosting y hacer uso de un CDN (Content Delivery Network) que sea accesible de forma óptima por el mayor número de usuarios.

## Herramientas para comprobar la velocidad de carga de un sitio web

- **PageSpeed Insights** (en <https://developers.google.com/speed/pagespeed/>)
- **GT Metrix** (en <https://gtmetrix.com/>)
- **Pingdom Tools** (en <https://tools.pingdom.com/>)

## ¿Cómo realizar las mejoras en la velocidad de carga?

A la hora de realizar las mejoras oportunas, en el caso de **WordPress** hay **multitud de plugins** que nos pueden facilitar tareas de minificación de código y la optimización de imágenes. Algunos de los plugins más destacados son WP Rocket, Smush o Imagify.

- En el caso de **no usar ningún tipo de CMS** en la creación de tu sitio web, hay herramientas como **Grunt o Gulp** que nos pueden ayudar a optimizar nuestro código y mejorar la velocidad de carga de nuestra página web.