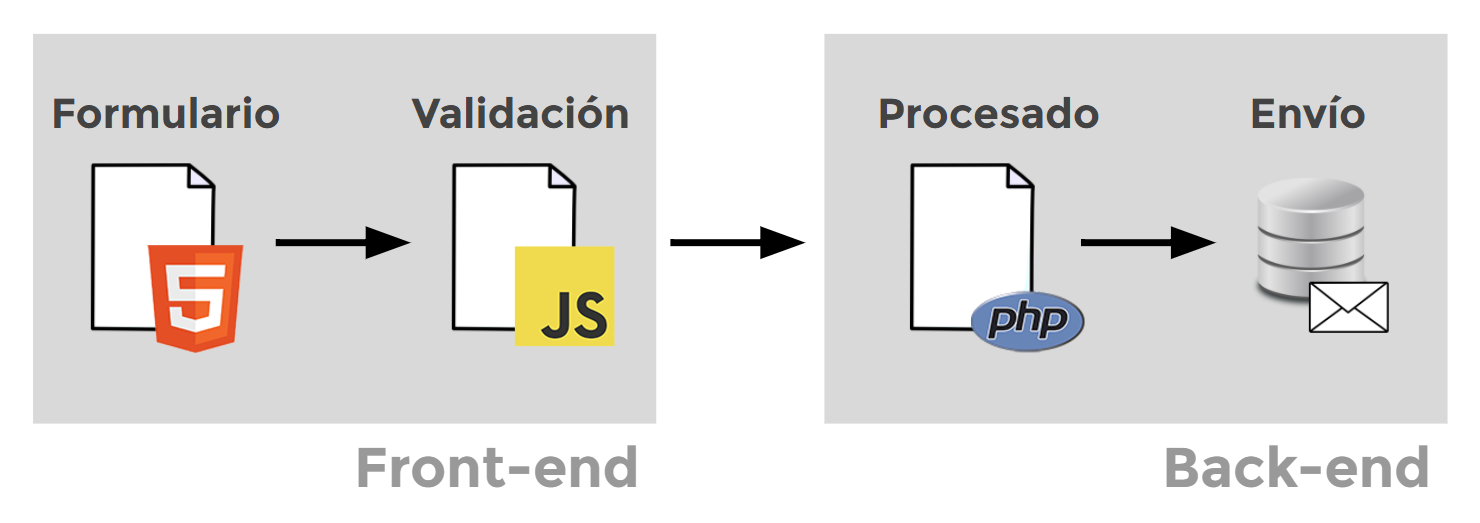
**Validaciones HTML5**

Al crear un formulario en HTML, debemos ser **conscientes** de un detalle ineludible: **los usuarios se equivocan al rellenar un formulario**. Ya sea por equivocación del usuario, ambigüedad del formulario, o error del creador del formulario, el caso es que debemos estar **preparados y anticiparnos** a estos errores, para intentar que los datos lleguen correctamente a su destino y evitar cualquier tipo de moderación o revisión posterior.

Para evitar estos casos, se suele recurrir a un tipo de proceso automático llamado **validación**, en el cuál, establecemos unas pautas para que si el usuario introduce alguna información incorrecta, deba modificarla o en caso contrario no podrá continuar ni enviar el formulario correctamente.

Un **esquema tradicional de validación** de ejemplo podría ser el siguiente, donde tenemos un formulario diseñado en **HTML5** y mediante **javascript** realizamos una validación para comprobar que los datos son correctos. Una vez superada, el formulario se **envía al back-end** de la página para ser procesado y enviarlo por email (por ejemplo).

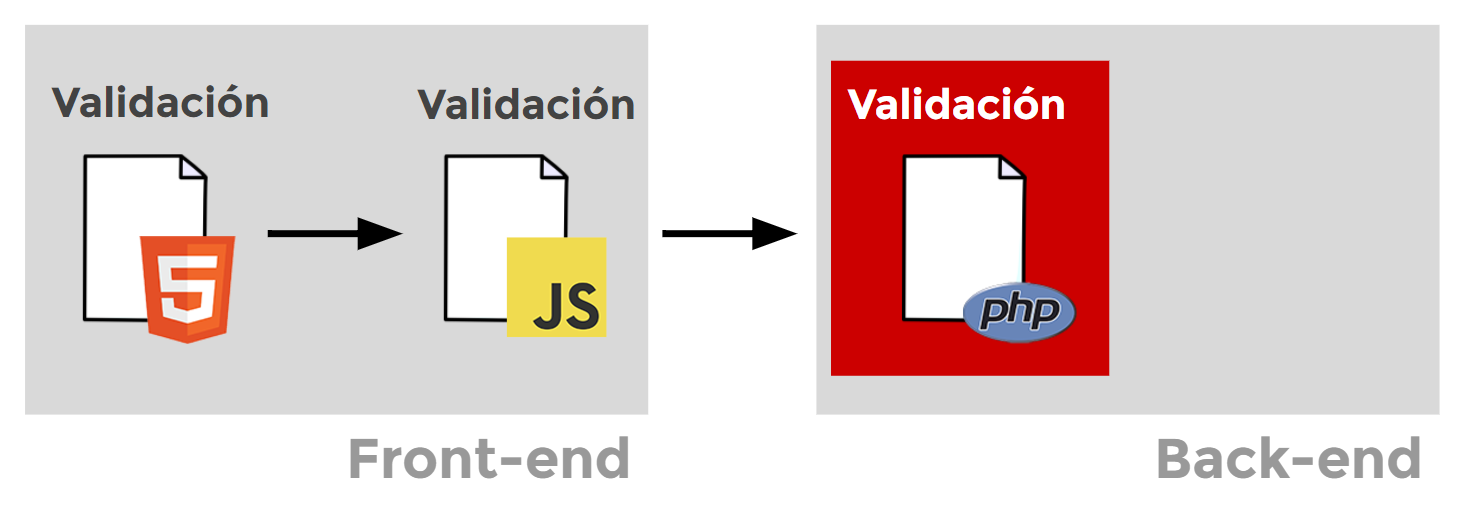


### Tipos de validación

Cada vez que creamos un formulario, la **validación** de los datos introducidos estará situada en uno de los siguientes casos (colocados de peor a mejor):

1. En este primer caso, **el formulario no tiene validación** de ningún tipo. El usuario puede escribir la información y el sistema no comprobará los datos, ni realizará ningún tipo de validación. Es el peor escenario **posible**, puesto que el usuario podría enviar desde **información incorrecta**, hasta **datos malintencionados** que podrían comprometer la seguridad de la página.
2. Otro caso podría ser que **el formulario tiene validación sólo en el front-end** (cliente). De esta forma, los datos son verificados en el navegador del usuario antes de enviarse, pero carecen de validación en el **back-end**, por lo que un usuario malintencionado podría **eliminar la validación del front-end y saltársela**, enviando datos malintencionados que comprometan la seguridad de la página.
3. El tercer caso posible es uno donde **el formulario tiene validación sólo en el back-end**. De esta forma, garantizamos que un usuario malintencionado no podrá eliminar el proceso de validación, y los datos siempre se comprobarán. Sin embargo, la desventaja de este método es que el usuario puede rellenar un formulario y es necesario que lo envíe (con la tardanza que eso puede acarrear), se procese en el back-end y al devolver un error, el usuario tenga que retroceder al formulario y en algunos casos, incluso tener que volver a rellenar todos los campos de nuevo.
4. Por último, tendríamos el **caso ideal**, donde el formulario **tiene validación en el front-end y en el back-end**, también denominado **doble validación**. En este caso, el formulario es sometido a un proceso de validación en la parte del front-end, y si lo supera, vuelve a pasar otro proceso de validación en el back-end. La desventaja de este método es que conlleva más trabajo de validación, pero es el sistema recomendado, puesto que es más estricto y sobre todo, más seguro.

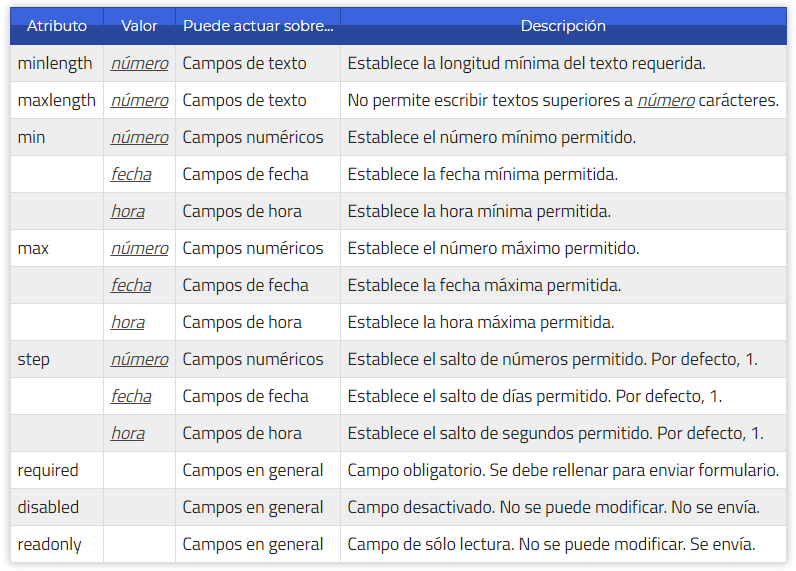
Tradicionalmente, la validación de un formulario se ha hecho siempre en Javascript, sin embargo, HTML5 introduce unos nuevos atributos para formularios que permiten realizar la validación del formulario directamente en HTML5, sin necesidad de recurrir a Javascript (aunque es posible hacerlo si se desea):



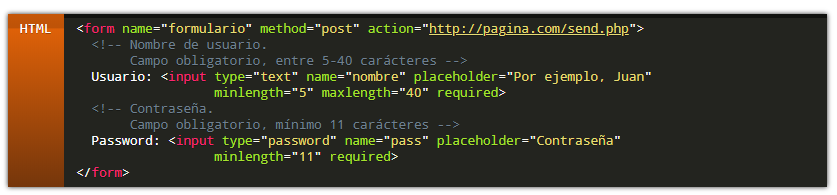
Recuerda siempre realizar la **validación en el servidor o back-end** (en rojo), ya que puede evitar problemas graves de seguridad.

### Atributos básicos de validación

En nuestros campos de entrada de datos, se pueden utilizar ciertos **atributos**para realizar validaciones sencillas. Algunos de estos atributos ya lo hemos visto en apartados anteriores, sin embargo, vamos a comentarlos uno por uno:



Con estos atributos, podemos crear validaciones básicas en nuestros campos de entrada de datos, obligando al usuario a rellenar un campo obligatorio, forzando a indicar valores entre un rango numérico o permitiendo sólo textos con un tamaño específico, entre otros.



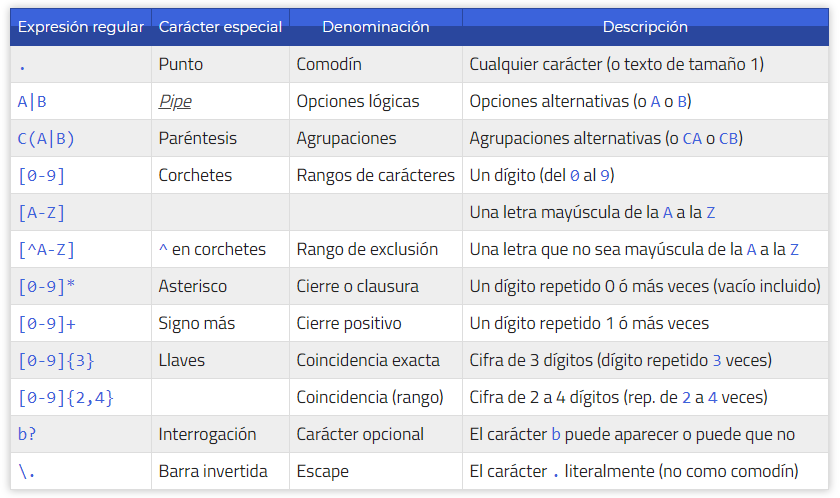
Ej 28

### Patrones de validación HTML5

No obstante, aunque los atributos de validación básicos son muy interesantes y pueden facilitarnos la tarea de validación, en muchos casos son insuficientes. Para ello tenemos los **patrones de validación HTML5**, mucho más potentes y flexibles, que nos permitirán ser mucho más específicos utilizando expresiones regulares para validar datos.

Una **expresión regular** es una cadena de texto que representa un posible patrón de coincidencias, que aplicaremos mediante el atributo **pattern** en los campos que queramos validar.

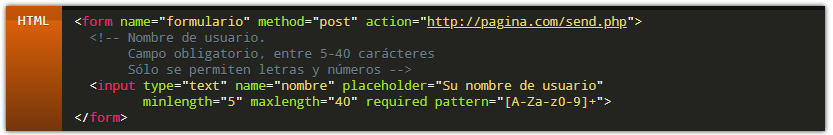
Características básicas de las expresiones regulares:



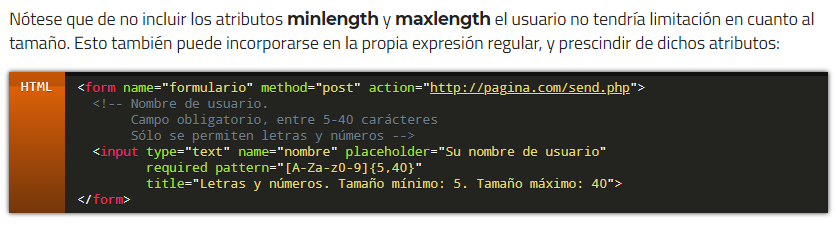
#### Ejemplos de patrones HTML5

Casos reales, para afianzar conceptos sobre expresiones regulares y su aplicación en los atributos **pattern**:

* Tipo de campo: **Nombre de usuario**
* Campo obligatorio: **required**.
* Entre 5-40 carácteres: **minlength="5" maxlength="40"**
* Sólo se permiten letras (mayúsculas y minúsculas) y números: **pattern="[A-Za-z0-9]+"**



EJ 29.

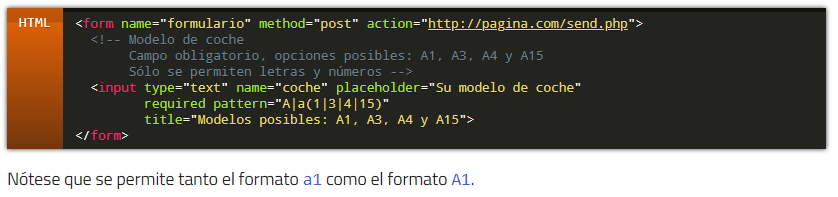


Sin embargo, en este caso, no se limitará al usuario a la hora de escribir, como hace **maxlength**, sino que permitirá al usuario escribir la información que desee y en caso de no pasar la validación, mostrará un mensaje de advertencia y no lo dejará continuar hasta que termine. Podemos ampliar el mensaje de advertencia incluyendo el texto en el atributo **title**.

EJ 30.

En el siguiente caso, se pide al usuario que indique el modelo de coche que posee, en un posible formulario de servicio técnico. Los modelos posibles son A1, A3, A4 y A15. En lugar de mostrar una **lista de selección**, podemos mostrar un campo de texto y colocar una validación como la siguiente:

* Tipo de campo: **Modelo de coche**
* Campo obligatorio: **required**.
* Sólo se permiten las opciones: **A1, A3, A4 y A15**



EJ 31

**Herramientas para expresiones regulares**

A continuación, algunas herramientas útiles para crear o probar **expresiones regulares**:

| Herramienta | Descripción |
| --- | --- |
| [Debuggex](https://www.debuggex.com/) | Herramienta de creación gráfica de expresiones regulares. |
| [RegExR](http://regexr.com/) | Herramienta para construir y testear expresiones regulares. |
| [RegEx101](https://regex101.com/) | Herramienta para construir y testear expresiones regulares. |
| [RegExBuddy](https://www.regexbuddy.com/create.html) | Herramienta profesional para el trabajo con expresiones regulares. |
| [RegExplained](https://leaverou.github.io/regexplained/) | Herramienta visual para explicar el funcionamiento de una expresión regular. |
| [RegExper](https://regexper.com/) | Herramienta para generar diagramas visuales de expresiones regulares. |
| [Rubular](http://rubular.com/) | Editor de expresiones regulares. |