

## Interfaz de usuario (UI)

Lunes, 19 de febrero de 2018 11:37

1. Elementos de navegación
  - a. Navegación. Estructura y navegabilidad
  - b. GUI: Navegación basada en textos
    - i. Menús.
    - ii. Submenús anidados
  - c. GUI: Navegación con imágenes
    - i. Botones gráficos y en otras imágenes.
    - ii. Mapas
    - iii. Herramientas y utilidades.
2. Entrada de datos. Formularios
  - a. Elementos HTML
    - i. Estructura: etiquetas y grupos
    - ii. Controles. Botones con imágenes. Ficheros.
    - iii. Secciones numéricas y temporales. Colores.
    - iv. Controles dinámicos.
  - b. Diseño y gestión de formularios
    - i. CSS y formulario
    - ii. Validación HTML5



# Diseño Web: Estructura de un sitio web y navegabilidad

- Presentar claramente la navegación principal o global.
- Acceso claro y rápido a la página principal
- Sacar provecho del uso de encabezados y pies de página.
- Utilizar cautelosamente imágenes como parte de la navegación.
- Mantener consistencia en la selección de colores asignados distintos elementos.
- Ayudar al usuario a saber dónde están

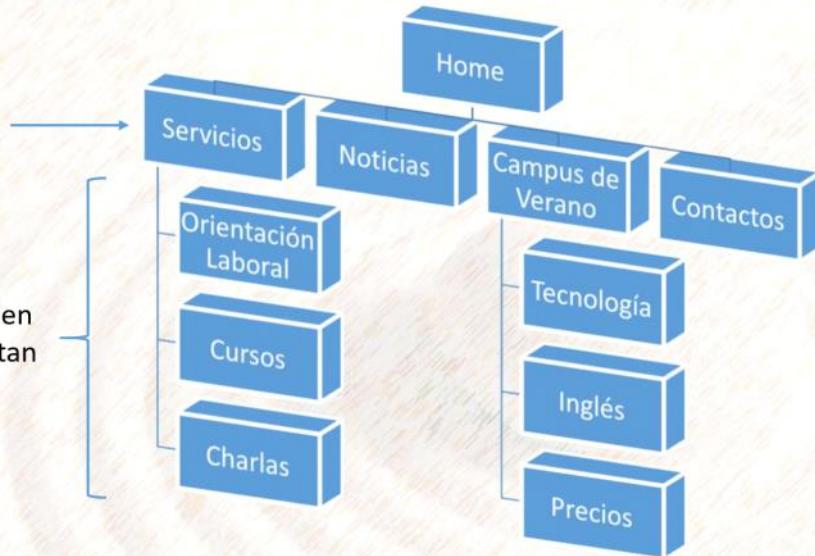
The screenshot shows the homepage of the World Resources Institute (WRI). At the top, there's a navigation bar with links for Home, Newsletter, Contact Us, and Other WRI Sites. Below the navigation is a search bar. The main content area features a large image of a forest fire in Indonesia. To the right of the image, there's a section titled "By the Numbers: The Economic Benefits of a National Climate Action Plan". Below this, there are several news items and links to other resources. The footer contains links for careers, permissions, media, library, WRI logos, contact, privacy policy, and donate. It also includes logos for Corporate Consultative Group and The WRI Center for Sustainable Transport.

# Navegación

Planificación de la estructura del sitio

Primer nivel =  
Menú Principal :  
número limitado de ítems

Algunos se subdividen  
y otros no lo necesitan



## Navegación basada en textos

<nav>  
...  
</nav>

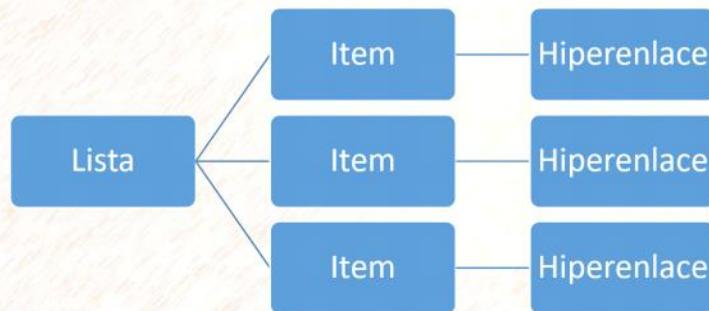
Barra de  
Navegación

Ayuda al navegador y a las CSS a  
identificar que se trata de una barra  
de navegación

En su interior se colocan los hiperenlaces a los elementos del menú,  
generalmente organizados en una lista

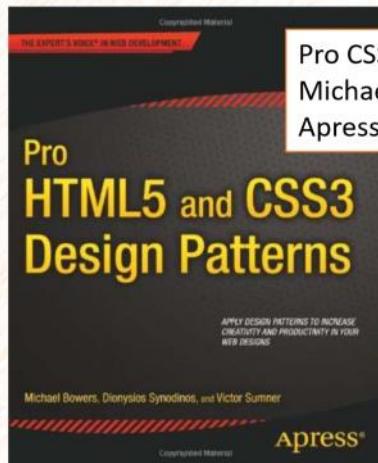
```
<nav id="miMenu">
    <ul>
        <li><a href="index.htm">Home</a></li>;
        <li><a href="services.htm">Servicios</a></li>;
        <li><a href="news.htm">Noticias</a></li>;
        <li><a href="campus.htm">Campus de Verav</a></li>;
        <li><a href="contact.htm">Contactos</a></li>
    </ul>
</nav>
```

## Menús horizontales: HTML



```
<nav>
  <ul>
    <li><a href="opA.html">Opción A</a></li>
    <li><a href="opB.html">Opción B</a></li>
    <li><a href="opC.html">Opción C</a></li>
    <li><a href="opD.html">Opción C</a></li>
  </ul>
</nav>
```

## Patrones de diseño



Pro CSS and HTML Design Pattern  
Michael Bowers  
Apress, 2011

## Menús horizontales: CSS

```
nav → width: x%;  
      overflow: auto  
  
ul  → list-style-type: none;  
     padding: 0px;  
     width: 100%;  
  
li  → display: block;  
     float: left;  
     width: x/items %;           display: inline-block;  
                                width: x/items -1 %;
```

```
<nav>  
  <ul>  
    <li><a href="opA.html">Opción A</a></li>  
    ....  
  </ul>  
</nav>
```

Un patrón para menús sensibles al paso del ratón o la selección mediante teclado

# Estilos para el menú

```
nav ul {  
    list-style-type: none;  
    .....}
```

Eliminamos los marcadores de la lista

```
nav li {  
    float:left;  
    .....}
```

Si el menú es horizontal, aplicamos float a cada uno de los ítems

```
nav a {  
    display: block;  
    ...}
```

Convertimos en bloque los hiperenlaces para tratarlos como "botones"

```
nav a:hover, nav a:focus {  
    ...}
```

Modificamos el aspecto de los ítems ante los eventos *hover* y *focus*

## GUI: Navegación

lunes, 5 de junio de 2017 11:08

# Modelos de menú horizontales

1



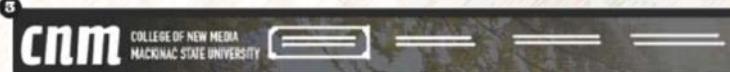
Solapas

2



Bloques

3



Botones

## **Respuesta al ratón**

```
div.caja {  
    margin: 40px;  
    width: 100px;  
    height: 100px;  
    border: 2px solid rgb(200,200,200);  
    background-color: rgb(250,250,250);  
    box-shadow: 5px 5px 5px 2px rgb(200,200,200);}  
div.caja:hover{  
    background-color: rgb(200,200,200);  
    box-shadow: 5px 5px 2px 2px rgb(128,128,128);}  
div.caja:active{  
    width: 110px;  
    height: 110px;  
    border: 4px inset rgb(200,200,200);  
    background-color: rgb(200,200,200);  
    box-shadow: -5px -5px 5px 2px rgb(128,128,128) inset;}
```



# **Submenús desplegables: CSS**

```
li ul {  
    display: none  
}
```

Los submenús (listas dentro de un ítem) permanecen ocultos

```
li:hover > ul {  
    display: block;  
}
```

Si una lista es “hija” de un ítem seleccionado (el siguiente nivel del menú) se muestra

```
li {  
    position: relative;  
}
```

Los li son contenedores, para poder fijar desde ellos posiciones absolutas

```
li ul li ul {  
    position: absolute;  
...}
```

Los submenus de tercer nivel se desplazan mediante coordenadas absolutas

## Navegación basada en botones

<nav>  
...  
</nav>

Barra de Navegación

En su interior se colocan las imágenes (e.g. botones) con los hiperenlaces a los elementos del menú

Sitios donde descargar botones

{ <http://www.aaa-buttons.com>  
[http://www.eosdev.com/eosdev\\_Buttons.htm](http://www.eosdev.com/eosdev_Buttons.htm)

Sitios con programas para crear botones

{ <http://www.crystalbutton.com>  
<http://www.buttongenerator.com>



<http://www.aaa-buttons.com>

<http://www.crystalbutton.com>

<http://www.buttongenerator.com>

## Ejercicio (5)

### Navegabilidad. Botones.

En una página web **HTML5**, añadiremos un menú principal basado en botones, utilizando las posibilidades gráficas de alguno de los sitios Webs que ayudan en la creación de estos elementos.

Objetivo Crear **menús** basados en **botones** en el marco de la nueva etiqueta <nav> de HTML5

# Utilidades en la Web

En <http://www.dynamicdrive.com/> podemos encontrar multitud de opciones para construir sistemas de navegación basados en scripts o en CSS.

The screenshot shows the Dynamic Drive website's navigation bar with links for New, Revised, CSS Library, Forums, Web Tools, Search, FAQ, Awards, Usage Terms, and Contact. Below the navigation bar, a breadcrumb trail shows Home > Menu and Navigation. A sidebar on the left lists categories like Calendars, Date & Time, Document Effects, Dynamic Content, User/System Preferences, Window and Frames, Other, XML and RSS, Partners, JavaScript Reference, DOM Reference, and Sweet Ads. The main content area displays several menu and navigation scripts:

- Menu and Navigation Scripts**
  - CSS Based** (selected)
  - Multi-level menus**
- Page Sidekick Menu** FF2+ IE8+ Opr7+  
This menu displays itself prominently on the page with the help of CSS3 transforms and transitions. The menu glides in from the left edge of the screen while shrinking the rest of the page content into the background, bringing the user's focus squarely on the menu itself. Clicking anywhere on the page again hides the menu and returns the page back to its original state.
- Omni Slide Menu** FF1+ IE8+ Opr7+  
User Submitted  
An Omni Slide Menu is an super versatile slideout menu that reacts to the mouse hovering over and out of it. It can be positioned to the left, top, or right edge of the browser window, supports static positioning, plus multiple instances of the same menu on a page and more. Very cool.
- Dynamic-Fx Slide-In Menu** FF1+ IE8+ Opr7+  
User Submitted  
A multi-featured slide-in menu script that supports some of the most requested features such as frame targeting, static positioning, and header(s) displays.

## Navegación basada en imágenes

```
<map>...</map>
```

Se puede utilizar una única imagen en la que distintas áreas (*hotspots*) corresponden a los hiperenlaces a los elementos del menú

```
<map name="nombreImagen">  
  <area ...>  
  <area ...>  
</map>
```

Las áreas “activas del mapa se definen con la etiqueta `<area>`

Atributos

`name`

→ nombre de la imagen a utilizar

La propia imagen a utilizar se indica mediante la etiqueta `<img>` en la que se incluye el atributo *usermap*, con el mismo nombre utilizado al definir el mapa precedido por #

```
<img usermap="#nombreImagen">
```

# Navegación basada en imágenes

## <area>

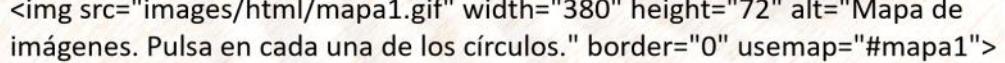
Atributos

- shape** → rect  
circle  
poly
- coords** → (rect):  $X_{\text{inicio}}, Y_{\text{inicio}}, X_{\text{final}}, Y_{\text{final}}$   
→ (circle):  $X_{\text{centro}}, Y_{\text{centro}}, r$   
→ poly:  $X_{\text{punto 1}}, Y_{\text{cpunto 1}}, \dots, X_{\text{punto n}}, Y_{\text{cpunto n}}$
- href** → URL

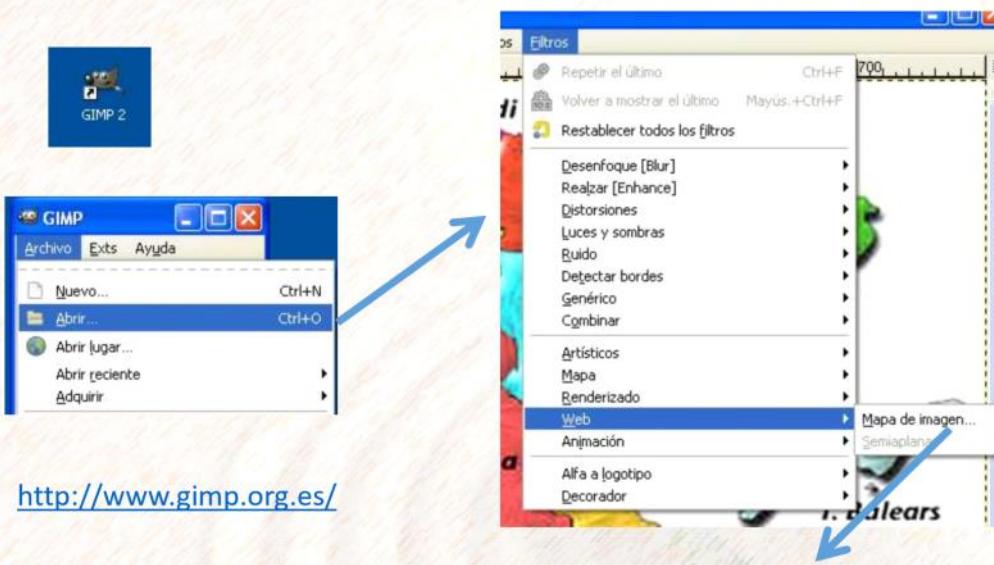
```
<area shape="rect" coords="284,170,352,314" href="enter.htm">
```

## Mapas de imágenes: ejemplo



```
<map name="mapa1">
    <area alt="Pulsa para ver la página de mis amigos" shape="CIRCLE"
        coords="44,36,29" href="#">
    <area alt="Pulsa para ver mi novia" shape="CIRCLE" coords="140,35,31"
        href="#">
    <area alt="Pulsa para conocer a mi Familia" shape="circle"
        coords="239,37,30" href="#">
    <area alt="Pulsa para conocer mi trabajo" shape="CIRCLE"
        coords="336,36,31" href="#">
</map>

```

# *Creación de mapas con gIMP (1)*



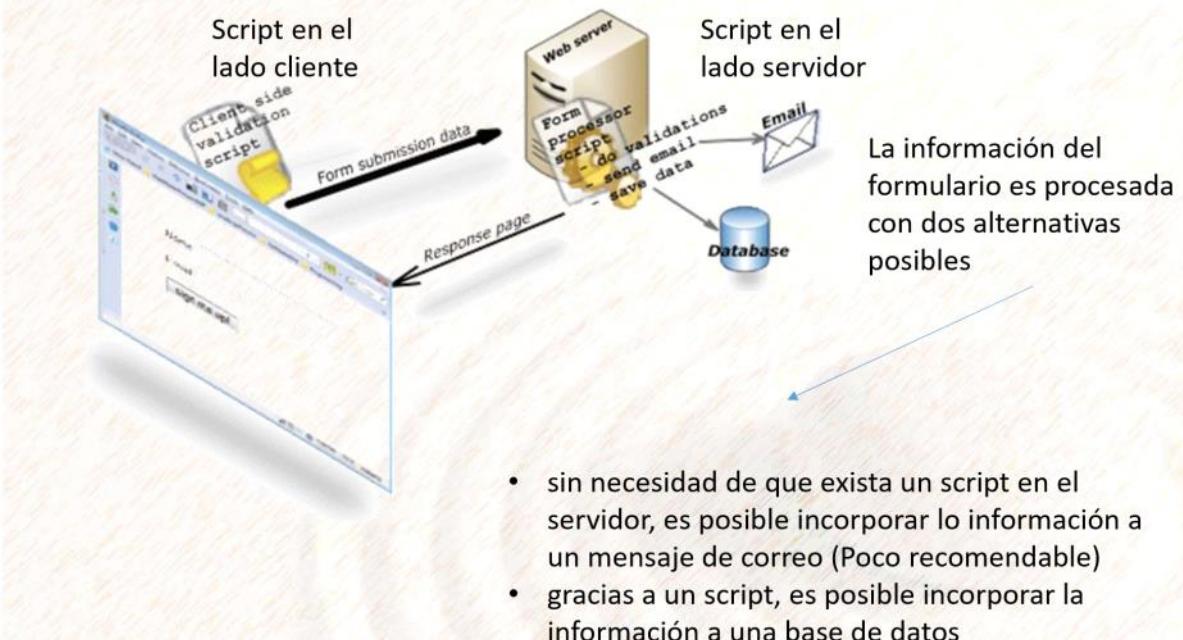
El resultado será un ficheo .map: un txt con el código HTML correspondiente al mapa

# Aspecto del cursor

<cursor>

alias	ew-resize	nwse-resize	text
all-scroll	grab	no-drop	URL
auto	grabbing	none	vertical-text
cell	help	not-allowed	w-resize
context-menu	move	pointer	wait
col-resize	n-resize	progress	zoom-in
copy	ne-resize	row-resize	zoom-out
crosshair	nesw-resize	s-resize	initial
default	ns-resize	se-resize	inherit
e-resize	nw-resize	sw-resize	

# Formularios



# *Elementos: controles*

**Text input**

Ivy

**Password input**

\*\*\*\*\*

**Text area**

Enter your comments...

**Radio buttons**

Rock  Pop  Jazz

**Checkboxes**

iTunes  Last.fm  Spotify

**Drop-down boxes**

iPod

**Submit buttons**

Subscribe

**Image buttons**

SUBSCRIBE

**File upload**

Upload

Browse...

# Formularios

**<form>...</form>**

Incluye todo el contenido del formulario

## Atributos

<b>method</b>	get post	Alternativas del protocolo http Lo normal es usar post, que incluye los datos en el cuerpo del mensaje, y no en la url
<b>action</b>	mailto script (PHP, ASP, Perl) CGI program	Como se procesará el formulario
<b>enctype</b>	text/plain	Especifica el formato necesario para el correo

```
<form method="post"  
action="mailto:edward@contoso.com" enctype="text/plain">  
<form method="post"  
action="http://www.contoso.com/cgi-bin/feedback.pl">
```

# *Etiquetas y grupos*

## **<label>...</label>**

Atributos

Identifica el texto que acompaña a cada control del formulario, mejorando su accesibilidad

**for** → nombre o id del control al que está asociada la etiqueta

## **<fieldset>...</fieldset>**

Agrupa un conjunto de controles, recuadrándolos

Contact details

Email:	<input type="text"/>
Mobile:	<input type="text"/>
Telephone:	<input type="text"/>

## **<legend>...</legend>**

Se sitúa dentro del anterior, permitiendo darle un título

## Controles: entradas de texto (1)

**<input>**

**type** → text

Entrada de texto

Name:

Atributos

**name** → Nombre: todos los controles tienen que tener un nombre único

**value** → valor por defecto y devuelto por el control

**maxlength** → Longitud máxima en caracteres que puede alcanzar el texto introducido

(descontinuado)

**size**

Tamaño en pixels de la caja de texto (si el texto introducido es mayor, habrá scroll)

el valor por defecto es de 20 px.

# Controles: atributos HTML5



- autocomplete** → habilita la capacidad de autocompletado
- autofocus** → especifica el campo que recibirá el foco automáticamente al cargarse la página
- placeholder** → valor por defecto de la entrada de texto, pero que no se devuelve.
- list (datalist)** → Lista de opciones sugeridas; el usuario puede elegir una o escribir su propia entrada
- spellcheck** → habilita la capacidad de comprobar la ortografía
- required** → Impide que se envíe el formulario si no se completa; junto con Pattern y Novalidate permiten una validación básica a nivel HTLM5

## Controles: entradas de texto (2)

<input>

type → password

Entrada de contraseña

Username: ivy

Password: \*\*\*\*\*

Atributos

**name**

**value**

**maxlength**

Entrada de texto especial, que no muestra los caracteres que se escriben

**required**



# Controles: entradas de texto HTML5

**<input>**

**type → email**

Entrada de correo

**<input>**

**type → URL**

Entrada de una URL

**<input>**

**type → tel**

Entrada de teléfonos

**<input>**

**type → search**

Entrada de texto a buscar



Atributos

**name**

**value**

**maxlength**

Los distintos navegadores dan soporte a diversos mecanismos de validación

**required**

**placeholder**



# Controles HTML5: implementación

type → email

abc

Please enter a valid email address

Opera 11



type → URL

ivy

Submit

Please enter a URL



type → tel



type → search

Q-|

Recent Searches

test

banana

Clear Recent Searches

Submit

Safari 5

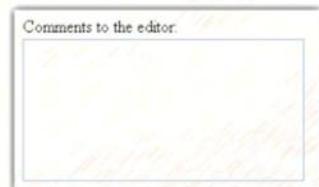
Los tipos email, tel y URL, en Safari para iOS, desencadenan un teclado en pantalla específico.

# Controles: áreas de texto

```
<textarea>  
</textarea>
```

```
<textarea>  
valor por defecto  
(opcional)  
</textarea>
```

Área de texto  
(entrada multilínea)



## Atributos

**name** → Nombre único que todos los controles tienen que tener de cara al script del lado servidor

## Descontinuados

**cols** → anchura en número de caracteres (por línea) que pueden incluirse en el área de texto sin necesidad de scrolling.  
**rows** → Número de líneas (filas) que pueden incluirse en el área de texto sin necesidad de scrolling.

```
<input>
```

**type** → **hidden**

control invisible que tiene utilidad para el programador,  
e.g. para saber desde qué página se envía un formulario

# Controles: **botones**

<input>

type → submit

Botón

<input>

type → reset

Submit Clear

### Atributos

value → valor que aparece en el botón y que será devuelto por el control al enviar el formulario.  
Por defecto vale **submit** o **reset**

```
<input type="submit" value="Enviar">  
<input type="reset" value="Borrar">
```

size

→ Tamaño en *pixels*. Por defecto se adapta al tamaño de value

Submit

ejecuta la aplicación por defecto correspondiente al atributo action del formulario (un enlace mailto o la URL de un servidor dinámico)

Clear

elimina toda la información introducida en el formulario

# Controles: botones de imagen

<input>

type → image

Atributos

src

→ URL de la imagen utilizada

Descontinuados

width

height

dimensiones, e.g.  
en píxeles

Formnovalidate  
Formaction  
Formmethod  
Formtarget  
Formenctype

En los input submit e image, se pueden definir los valores relativos al servidor, que normalmente se indicarían en el form

SUBSCRIBE

# Botones genéricos

```
<button>  
</button>
```

type → submit

Botón

type → reset

Submit Clear

type → button

Permite combinar textos e imágenes  
en un mismo botón



```
<button>  
  
Add</button>
```

## Controles: selección única

**<input>**

type → checkbox

Opción lógica (S/N)

**Check box**

Subscribe

### Atributos

**name** → Nombre único que todos los controles.

**value** → valor devuelto al seleccionar el control.  
Por defecto es *on*

**checked** → si vale *checked*, el control aparece  
marcado por defecto

```
<input type="checkbox" name="Subscribe" checked="checked">
```

# Controles: selección múltiple

<input>

type → radio

Opción múltiple  
(a elegir una)

**Radio buttons**

- Gold
- Silver
- Bronze

Atributos

**name**

nombre único que todos los controles. Es **el mismo** en el caso de los *radio buttons* que forman **un grupo**, generalmente agrupados dentro de un párrafo

**value**

valor devuelto si esta es la opción seleccionada (un valor distinto para cada rb del grupo)

**checked**

si vale *checked*, el control aparece marcado por defecto

```
<p>
<input type="radio" name="category" value="gold" checked="checked">Oro<br>
<input type="radio" name="category" value="silver">Plata<br>
<input type="radio" name="category" value="bronze">Bronce</p>
```

# Controles: listas (1)

```
<select>  
  </select>
```

```
<option>  
  </option>
```

Atributos  
(de <select>)

**name**

```
<select>  
  <optgroup label="valor">  
    <option>valor</option>  
    <option>valor</option>  
    .....  
  </select>
```

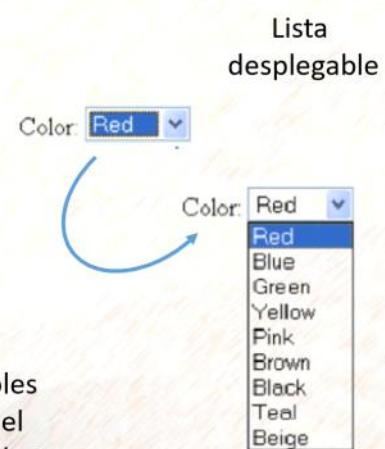
Atributos  
(de <option>)

**value**

**selected**

nombre único que todos los controles  
tienen que tener de cara al script del  
lado servidor. Se aplica a todo el *select*

valor devuelto si esta es la opción  
seleccionada. Se aplica a cada opción y por  
defecto será el texto de la opción (no suele  
usarse)  
opción seleccionada por defecto



## Controles: listas (2)

Atributos  
(de <select>)

**size** → número de opciones visibles. Se aplica a todo el *select*  
El valor 1 origina la habitual lista desplegable (*drop-down list*)

**multiple** → Si su valor es “*multiple*”, permite que se seleccione  
más de una opción simultáneamente

Atributos  
(de <optgroup>)

**label** → texto que separa “subgrupos”, sin ser una opción elegible

# Controles: listas de datos (datalist)

**<input>**

type → text

list

<nombrelista>

**<datalist>**

Atributos

**id** → Nombre de la lista

**<option>**

Cada una de las opciones

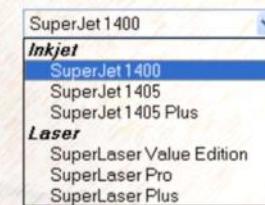
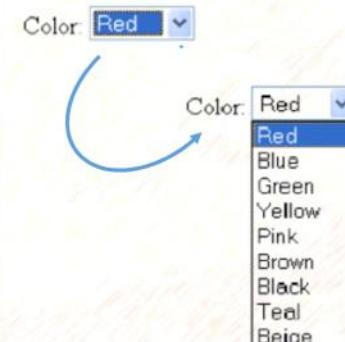
lista desplegable (*drop-down list*) de opciones que se incorpora a la caja de textos estandar

```
<input type="text" list="datalist">
<datalist id="datalist">
    <option>First Choice</option>
    <option>Second Choice</option>
    <option>Third Choice</option>
</datalist>
```

# Controles: listas (ejemplo)

```
<p>Color: <select name="colors" size="1">
    <option>Red</option>
    <option>Blue</option>
    <option>Green</option>
    <option>Yellow</option>
    <option>Pink</option>
    <option>Brown</option>
    <option>Black</option>
    <option>Teal</option>
    <option>Beige</option>
</select></p>
```

```
<p>Select your printer model:
<select name="printers" size="1">
    <optgroup label="Inkjet">
        <option>SuperJet 1400</option>
        <option>SuperJet 1405</option>
        <option>SuperJet 1405 Plus</option>
    </optgroup>
    <optgroup label="Laser">
        <option>SuperLaser Value Edition</option>
        <option>SuperLaser Pro</option>
        <option>SuperLaser Plus</option>
    </optgroup>
</select></p>
```



## Controles: ficheros

<input>

type → file

Browse...

- permite que los usuarios envíen un fichero al servidor Web
- el método del formulario debe ser post, ya que get no es capaz de enviar ficheros
- automáticamente se genera un botón asociado a la capacidad de recorrer el equipo local para localizar y seleccionar el fichero;  
Este proceso depende del sistema operativo local, y desde la página web no se controla de ninguna manera

### Atributos

accept

→ permite indicar los tipos MIME admitidos

Es importante que el formulario utilice `method="post" enctype="multipart/form-data">`

# Formularios: selecciones numéricas



Spin box

<input>

type → number

Copies: 1

<input>

type → range

Slider

Copies: 1

Atributos

name

nombre único que todos los controles tienen que tener de cara al script del lado servidor.

value

valor devuelto

min

valor mínimo

max

valor máximo

step

valor posible del incremento

```
<input type="number"
       name="copies"
       min="0" max="100" step="1"
       value="1">
<input type="range"
       name="copies"
       min="1" max="4" step="1"
       value="1">
```

# Formularios: selecciones temporales



<input>

type → date...

Date picker

type="date"

selecciona fecha

dd/mm/aaaa ▾

type="month"

selecciona meses completos

----- de ----- ▾

type="week"

selecciona semanas completas

Semana --, ---- ▾

type="datetime-local"

selecciona fecha y hora

utilizando la zona de tiempo  
del usuario

dd/mm/aaaa --:-- ▾

type="time"

selecciona sólo la hora (sin fecha)

--:-- ▾

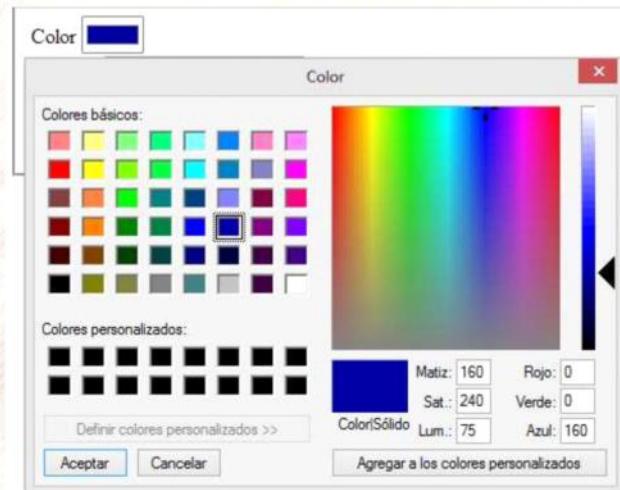
# Formularios: colores



<input> type → color

Permite seleccionar colores

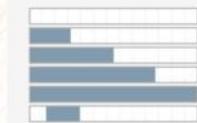
En algunos navegadores (opera, chrome, blackberry) aparece un gestor de colores



# Más HTML5, ligado a scripts



Live Demo



2048 (Grado elevado) ▾

<keygen>

Public Key / Private Key

<meter>

<progress>



<output>

```
<progress value="22" max="100"></progress>
```

```
<meter value="2" min="0" max="10">2 out of 10</meter><br>
<meter value="0.6">60%</meter>
```

## Formularios: Resumen

**<form>...</form>** Incluye todo el contenido del formulario

Atributos **method action enctype**

**<input> type** text / email / tel / URL / search

**<input> type** submit / reset

**<textarea> .... </textarea>**

**<input> type** checkbox / radio

**<select> <option>valor</option>..... </select>**

**<input> type** color / number / range / date...

Para conocer de forma actualizada el soporte en HTML5:

<http://www.wufoo.com/html5/>

Para conocer de forma actualizada el soporte en HTML5:

<http://www.wufoo.com/html5/>

Formularios: Resumen

<form>...</form> Incluye todo el contenido del formulario

Atributos method action enctype

<input> type text/ email / tel/ URL / search

<input> type submit/ reset

<textarea> .... </textarea>

<input> type checkbox/ radio

<select> <option> valor </ option>..... </select>

<input> type color / number/ range/ date...

# CSS y formularios (1)

## Pseudoclases específicas

:required { }	:optional { }	:default { }	:not()
:valid { }	:in-range { }	:out-of-range { }	:hover { }
:invalid { }			:focus { }

## Selector de atributo

[<atributo>]

[<atributo> = "valor"]

[autofocus] { }  
[autocomplete] { }  
[list] { }  
[placeholder] { }  
[type=range] { /\*cualquier "type" \*/ }  
[multiple] { }

## **CSS y formularios (2)**

Las **implementaciones** de las novedades HTML **NO** suelen ser modificables mediante CSS

Algunas excepciones, de momento mediante **códigos específicos de navegador**

```
::-webkit-input-placeholder {  
    /* Aplica estilo al texto del placeholder */  
}  
::-moz-placeholder {  
    /* Aplica estilo al texto del placeholder */  
}  
[type=search] {  
    -webkit-appearance: none;  
}  
::-webkit-validation-bubble-message {  
    padding: 50px;  
}
```

# Validación en HTML5



**required** → impide que se envíe el formulario si no se completa

**pattern** → permite añadir una **expresión regular** que se comprobará antes de enviar

```
<input type="text" name="zipCode" value=""  
      pattern="[0-9]{5}([-][0-9]{4})?" required="required" />
```

**novalidate** → a nivel de form, desactiva la validación y sobrescribe los campos required

En la validación sin scripts se usan también las pseudoclases de CSS3

**:required** → permiten definir el estilo que se aplicará a los campos requeridos y a los que no son validos (e.g, campos requeridos vacíos al enviar)

# **Procesamiento del formulario**

CGI (Common Gateway Interface)



CGI especifica un estándar para transferir datos entre el cliente y el programa.

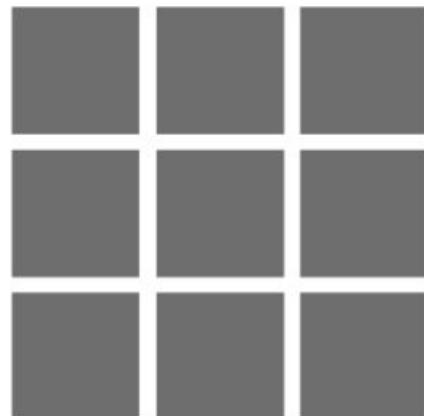
Scripts de servidor (PHP, Perl, Python, Ruby)

el lenguaje de programación del lado del servidor se incorpora directamente en el documento HTML en lugar de llamar a un archivo externo que procese los datos. El código es interpretado por un servidor web con un módulo de procesador del lenguaje en concreto

# Grid Layout

lunes, 18 de junio de 2018 17:29

<https://codepen.io/collection/nNdbqB/>



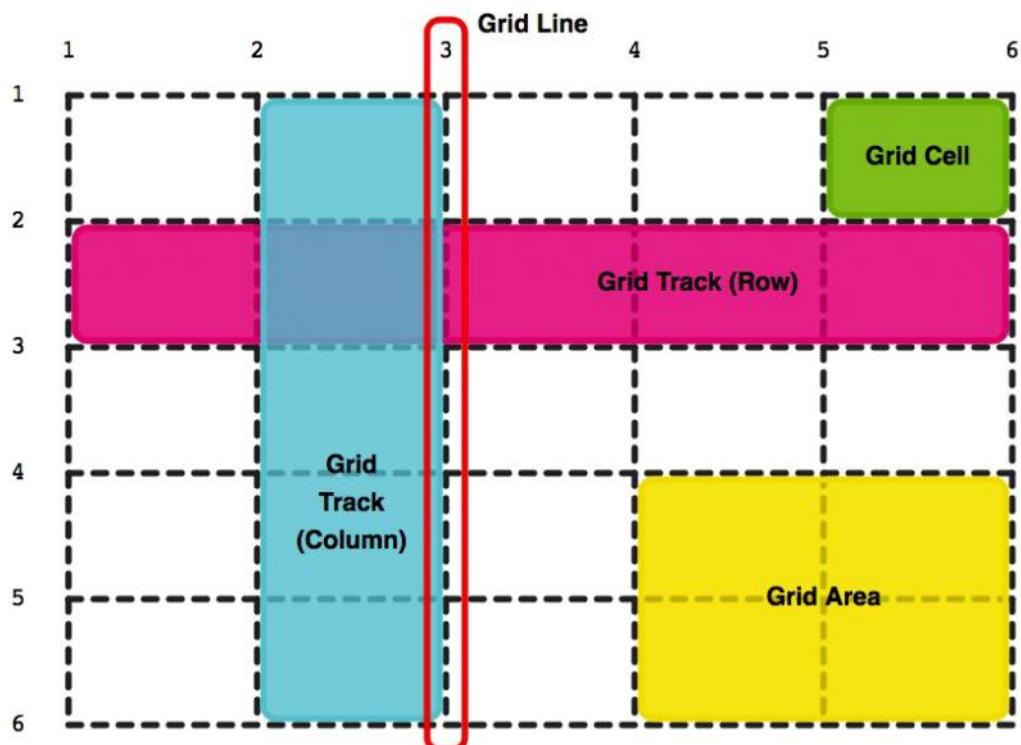
## ¿QUÉ ES GRID LAYOUT ?

- ▶ Sistema de rejilla en 2 dimensiones

## ¿QUÉ TIENE DIFERENTE ?

- ▶ Voy a poder colocar los ítems donde quiera...
- ▶ ... pero habrá ítems que se coloquen solos (auto-placement)
- ▶ Puedo hacer lo mismo de muchas formas...
- ▶ ... pero no todas hacen exactamente lo mismo.
- ▶ Controlo las 2 dimensiones, NO es Flex Box.
- ▶ La colocación de los ítems es muy libre, NO es una tabla.
- ▶ Tiene una extensa sintaxis.
- ▶ Nos va a volver un poco locos, pero va a cambiar el CSS para siempre.

## CONCEPTOS BÁSICOS



- Se define en base a líneas (Grid Line) que el sistema numera automáticamente
- 2 líneas consecutivas (i.e. de igual dirección) definen un track (Grid Track): filas y columnas
- el cruce de 2 tracks define una celda (Grid Cell)
- un conjunto de celdas adyacentes es un área (Grid Area)

# DISPLAY GRID

lunes, 18 de junio de 2018

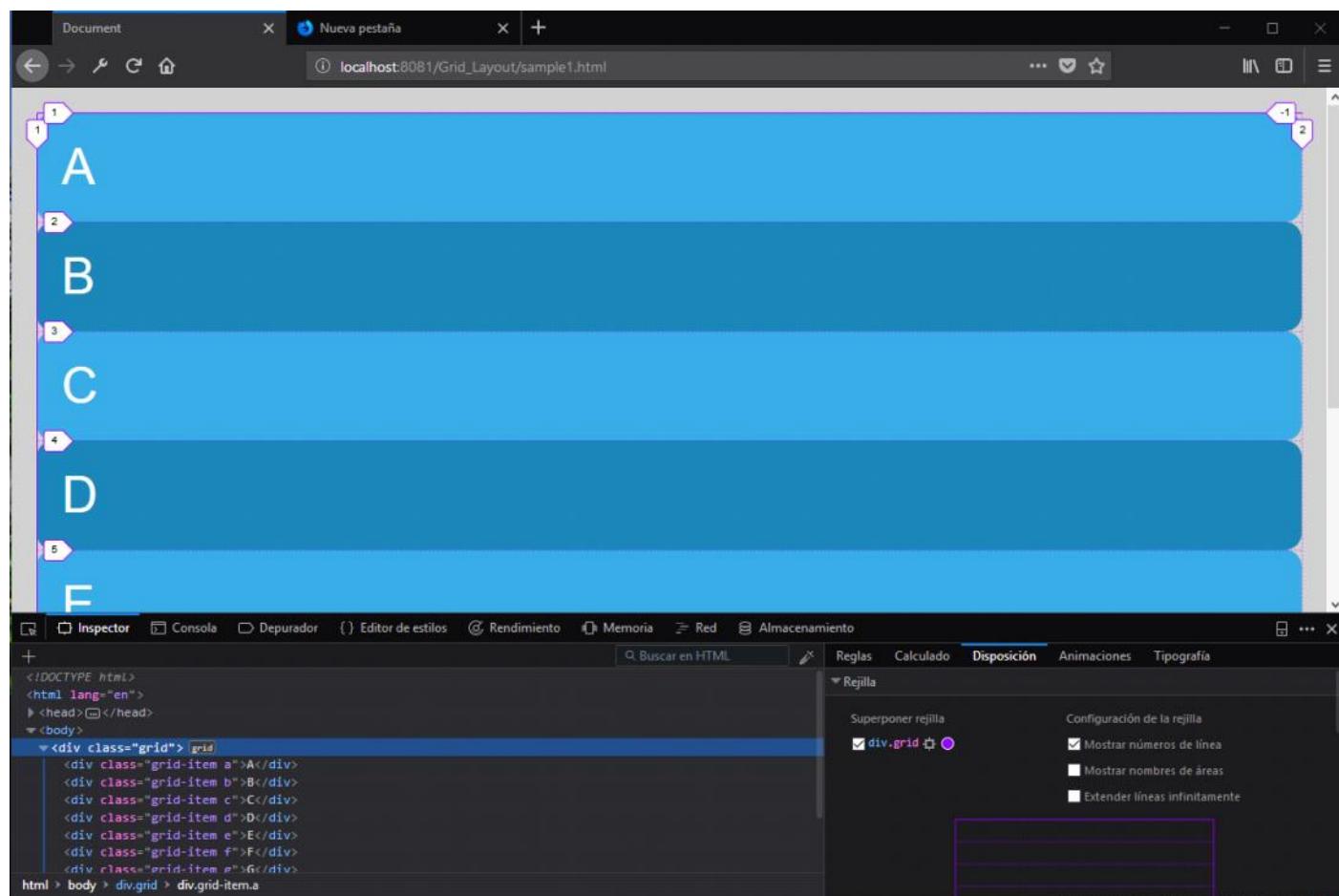
17:36

En cuanto declaro

DISPLAY: GRID o DISPLAY: INLINE-GRID

los hijos directos de ese elemento pasan a ser GRID ITEMS.

Firefox / Firefox Developer Edition



# TRACKS

Lunes, 18 de junio de 2018 17:37

Se definen cuando creamos el grid explícitamente.

Al contenedor grid, le indico cómo deben ser las filas y las columnas:

Existen múltiples sintaxis y formas de hacerlo.

DISPLAY: GRID genera una serie de tracks donde, si no indico más, los items se colocan automáticamente mediante el algoritmo de AUTO-PLACEMENT.

grid-template-columns: ancho ancho ancho  
grid-template-rows: alto alto alto alto  
grid-column-gap: ancho;  
grid-row-gap: ancho;  
grid-auto-rows: alto

- en rem
- en fr
- repeat (n, valor)

Se definen como serán las filas que aparezcan cuando haya más que las definidas en el grid explícito

## Tracks: mejor definición

- repeat(num\_tracks, ancho/anchos)
- minmax(níñimo {fijo}, máximo {fr})
- repeat(auto-fill / auto-fit, ancho/anchos)  
se define el tamaño de los track -> el sistema calcula cuantos caben. Si se combina con un minmax, se consigue un layout totalmente responsive

grid-template-columns: repeat(auto-fill, 10rem);  
grid-template-columns: repeat(auto-fill, minmax(10rem, 1fr))

# GRID ITEMS

Lunes, 18 de junio de 2018 23:29

La enorme potencia de GRID LAYOUT en parte viene dada porque una vez generada la rejilla, se puede posicionar cualquier item en cualquier posición dentro de la rejilla

```
grid-column-start: 2;
grid-column-end: 3;
grid-row-start: 2;
grid-row-end: 3; */
grid-column: 2/3;
grid-row: 2/span 2;

• indicando las líneas de inicio / final
• indicando cuántas líneas se extiende (span)
• definiendo un área completa con grid-area
• es posible colocar 2 items en la misma
  posición (capas)
```

EJ.APLICACIÓN. 1 - GRID 12 COLUMNAS

<http://nthmaster.com/>

```
<div class="grid">
  <div class="grid-item item1">1</div>
  <div class="grid-item item2">2</div>
  <div class="grid-item item3">3</div>
  <div class="grid-item item4">4</div>
  <div class="grid-item item5">5</div>
  <div class="grid-item item6">6</div>
  <div class="grid-item item7">7</div>
  <div class="grid-item item8">8</div>
  <div class="grid-item item9">9</div>
  <div class="grid-item item10">10</div>
  <div class="grid-item item11">11</div>
  <div class="grid-item item12">12</div>
  <div class="grid-item item13">13</div>
  <div class="grid-item item14">14</div>
  <div class="grid-item item15">15</div>
  <div class="grid-item item16">16</div>
  <div class="grid-item item17">17</div>
  <div class="grid-item item18">18</div>
  <div class="grid-item item19">19</div>
  <div class="grid-item item20">20</div>
  <div class="grid-item item21">21</div>
  <div class="grid-item item22">22</div>
  <div class="grid-item item23">23</div>
  <div class="grid-item item24">24</div>
  <div class="grid-item item25">25</div>
  <div class="grid-item item26">26</div>
  <div class="grid-item item27">27</div>
  <div class="grid-item item28">28</div>
</div>
```

```
.grid{
  display: grid;
  grid-gap: .5rem;
  grid-template-columns: repeat(12, 1fr);
}
.grid-item:nth-child(n+13):nth-child(-n+18){
  grid-column: span 2;
}
.grid-item:nth-child(n+19):nth-child(-n+22){
  grid-column: span 3;
}
.grid-item:nth-child(n+23):nth-child(-n+25){
  grid-column: span 4;
}
.grid-item:nth-child(26),
.grid-item:nth-child(27){
  grid-column: span 6;
}
.grid-item:nth-child(28){
  grid-column: span 12;
}
```

# Ejemplo Realista

martes, 19 de junio de 2018 19:22

```
<div class="wrapper">
  <div class="grid">
    <div class="grid-item item1 green">
      <div class="content">
        <div class="bold">It's content</div>
        <div class="text">e-commerce</div>
      </div>
    </div>
    <div class="grid-item item2 white">2</div>
    <div class="grid-item item3 blue">3</div>
    <div class="grid-item item4 yellow">4</div>
    <div class="grid-item item5 white">5</div>
    <div class="grid-item item6 red">6</div>
    <div class="grid-item item7 green">7</div>
    <div class="grid-item item8 blue">8</div>
    <div class="grid-item item9 white">9</div>
  </div>
</div>
```

```
.grid{
  display: grid;
  grid-template-columns: repeat(6, 95px);
  grid-template-rows: repeat(6, 95px);
}
.item1{
  grid-column: span 3;
}
.item2{
  grid-column: 5;
}
.item3{
  grid-column: 2/5;
}
.item4{
  grid-column: 3/span 3;
}
.item5{
  grid-column: 3;
}
.item6{
  grid-column: span 3;
}
.item7{
  grid-column: span 3;
}
.item8{
  grid-column: 2/span 3;
}
.item9{
  grid-column: 6;
```



[https://codepen.io/diana\\_aceves/pen/QqMRmX](https://codepen.io/diana_aceves/pen/QqMRmX)

# Líneas Nombradas

martes, 19 de junio de 2018 19:59

## Posicionando GRID-ITEMS: LÍNEAS NOMBRADAS

- ▶ Me ayudan a recordar cómo van los tracks en layouts complejos.
- ▶ En RESPONSIVE me evitan sobreescribir la colocación de algunos items.
- ▶ Si cambia el número de tracks NO TENGO QUE REPOSICIONAR TODOS LOS ELEMENTOS.

EJ.5.1 - LÍNEAS NOMBRADAS

EJ.5.2 - LINEAS NOMBRADAS (nombres múltiples)

EJ.5.3 - LÍNEAS NOMBRADAS (variación número de tracks)

EJ.5.4 - LÍNEAS NOMBRADAS (nombres repetidos)

# Áreas nombradas

miércoles, 20 de junio de 2018 18:12

## Posicionando GRID-ITEMS : ÁREAS NOMBRADAS

Dos nuevas propiedades :

- CONTENEDOR -> GRID-TEMPLATE-AREAS
- ITEMS -> GRID-AREA

En el contenedor se utiliza la propiedad grid-template-areas  
grid-template-areas:

```
"header header ... "
"sidebar main   main"
"footer footer footer";
```



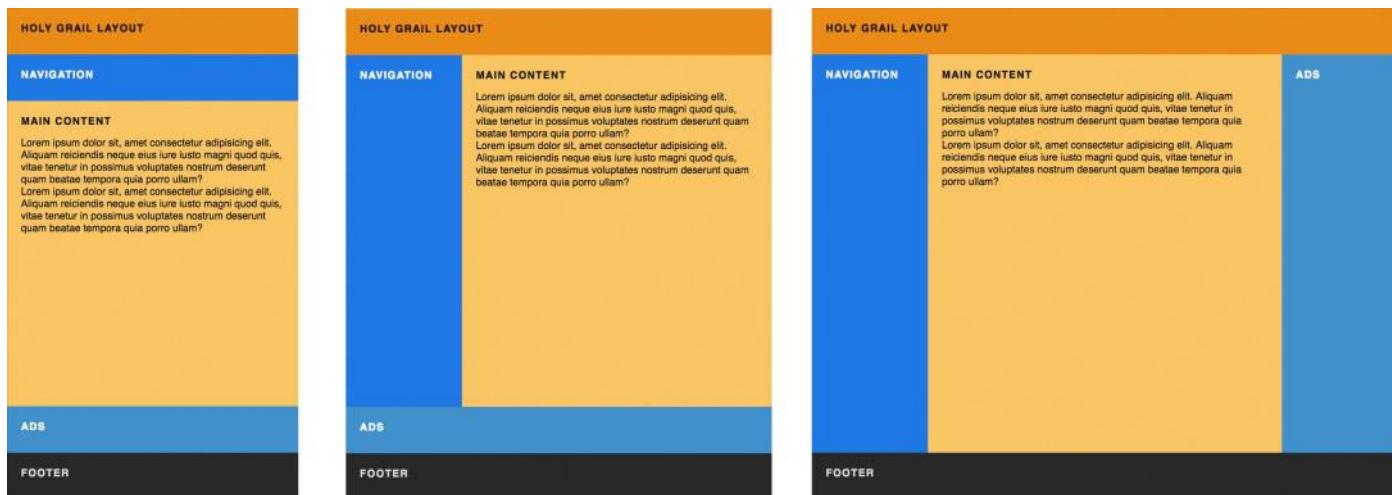
En los ítems se utiliza la propiedad grid-area

```
.header{
    grid-area: header;
}
```

HOLY GRAIL LAYOUT

# Ejemplo realista

miércoles, 20 de junio de 2018 18:57



# Alineamiento

miércoles, 20 de junio de 2018 19:00

## BOX ALIGNMENT: ALINEACIÓN DE LOS ÍTEMES

ALINEACIÓN EN COLUMN/BLOCK AXIS:

- align-self
- align-items

ALINEACIÓN EN ROW/INLINE AXIS:

- justify-self
- justify-items

## BOX ALIGNMENT: ALINEACIÓN DE LOS TRACKS

ALINEACIÓN EN COLUMN/BLOCK AXIS:

- align-content

ALINEACIÓN EN ROW/INLINE AXIS:

- justify-content

# Auto-Placement

miércoles, 20 de junio de 2018 19:02

## ALGORITMO AUTO-PLACEMENT

- Reglas que controlan dónde se colocan los items cuando no establecemos su posición o por colocar otro, pierde su sitio natural.
- Aunque no les dé información de dónde colocarse, ellos buscarán sitio en celdas libres o crearán los tracks necesarios para hacerlo.
- Lo primero que determina cómo funciona AUTO-PLACEMENT es el valor de la propiedad: `grid-auto-flow`

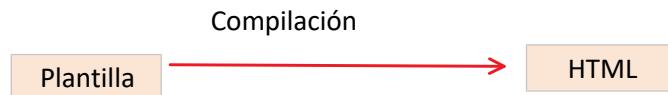
# Indice

sábado, 23 de junio de 2018 12:13

1. ¿Qué es Pug?
  - Preprocesadores
  - Instalación. Compilando Pug
2. Sintaxis básica
  - Significado del espacio en blanco
  - Doctypes
  - Etiquetas. HTML en línea Atributos Atajos (shorthands)
  - Comentarios
  - Bloque de expansión
3. Incorporando datos en plantillas (templates)
  - Sintaxis. Definiendo variables
  - Interpolación. Variables sin interpolación
  - Escapando
  - Envío de las variables al compilador
4. Lógica en plantillas
  - Agregar lógica con JavaScript
  - Operadores lógicos nativos (Built-in)
6. Mixins
  - Sintaxis y mecánica
  - El objeto arguments
7. Herencia de plantilla
  - Bloques (Blocks)
  - Extends
  - Includes
8. Organización de proyectos de Jade
  - Buenas prácticas generales

# (meta)Lenguajes de plantillas

sábado, 23 de junio de 2018 9:38



HAML created to replace ERB (the default template system included in Ruby) with a more abstract language based on indentation

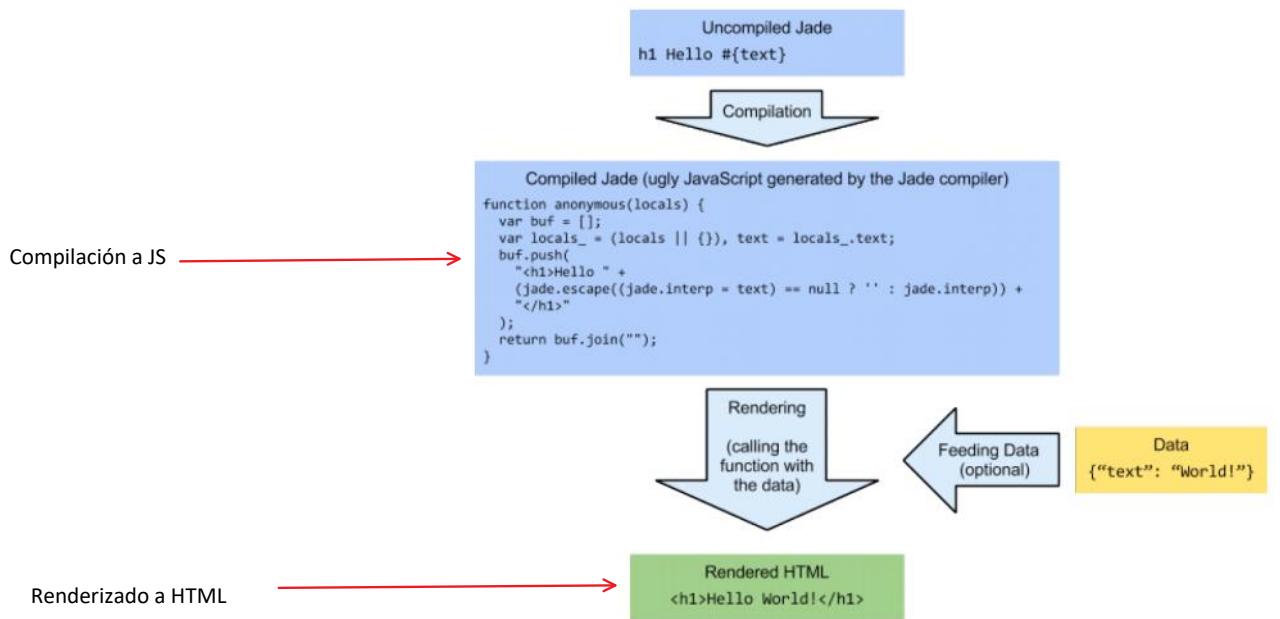
PHP, in addition to being a programming language used in servers, is itself a template language that servers render to HTML

*Mustache*, one of the most popular template languages based on JS, and which compiles to JS, so it can be executed on the server and in the client

[https://en.wikipedia.org/wiki/Comparison\\_of\\_web\\_template\\_engines](https://en.wikipedia.org/wiki/Comparison_of_web_template_engines)

## Preprocesamiento en Pug

viernes, 29 de junio de 2018 17:57



## Pug (Jade)

sábado, 23 de junio de 2018 9:40

<https://pugjs.org/api/getting-started.html>

<https://github.com/pugjs/pug>



Inicialmente denominado **Jade**, nombre que tuvo que abandonarse por ser una marca registrada, que corresponde a *Java Agent Development Framework* (<http://jade.tilab.com/>)

Pug es

- un motor de plantillas de alto rendimiento
- fuertemente influenciado por Haml
- implementado con JavaScript
- para Node.js y navegadores.

En Node.js, la librería Express, empleada en la creación de servidores Web, utiliza Pug como lenguaje de plantillas habitual

```
<!DOCTYPE html>
<html>
<head></head>
<body>
  <h1>Meet Pug</h1>
  <p>
    A simple Pug example.
    You'll learn to write
    all of this now.
  </p>
  <p>Pug FTW!</p>
</body>
</html>
```

HTML



```
doctype html
html
  head
  body
    h1 Meet Pug
    p.
      A simple Pug example.
      You'll learn to write
      all of this now.
    p PugFTW!
```

PUG

# Instalación

sábado, 23 de junio de 2018 9:56

Instalación de pug como parte de un proyecto JS

```
$ npm install pug
```

Instalación global de la línea de comandos

```
$ npm install pug-cli -g
```

Compilación / Renderizado

```
$ pug file.pug
```



file.html

Compilación

```
$ pug --client --no-debug file.pug
```



file.js

## Scripts de npm

jueves, 29 de marzo de 2018 21:05

Podemos ejecutar comandos de un paquete mediante la definición de scripts en el archivo package.json

'npm start' no necesita especificar 'run', es el script estándar para lanzar un proyecto npm

```
"scripts": {  
  "sass": "node-sass -o ./dist/ ./src/",  
  "stylus": "stylus -w ./src/style.styl -o ./dist/style.css",  
  "start": "supervisor main.js"  
}  
  
$ > npm run sass  
$ > npm run stylus  
$ > npm start
```



podemos reemplazar los gestores de tareas (*task runners*) por simples scripts NPM para automatizar tareas.

<https://www.keithcirkel.co.uk/how-to-use-npm-as-a-build-tool/>

## Tarea npm para pug

viernes, 29 de junio de 2018 18:08



Si se necesita combinar con otras tareas, e.g. otros metalenguajes como SASS, se utiliza parallelshell

- las tareas se definen como strings
- los strings se limitan con " escapadas

```
$ npm install parallelshell
```

```
{  
  "sass": "..."  
  "pug": "pug -w -P -p ./src/pug/partials -O ./src/pug/options.json ./src/pug/templates -o ./src/"  
  "watch:metlang": "parallelshell \"npm run sass\" \"npm run pug\""  
}
```

# Sintaxis básica de pug

viernes, 29 de junio de 2018 18:13

## Significado del espacio en blanco. Etiquetas

- no existen etiquetas de cierre
- no se utiliza <> para indicar las etiquetas
- la indentación refleja el anidamiento de unas etiquetas en otras

```
doctype html
html(lang="en")
  head
  body
    header
      h1 Empezando PUG
    main
      p Primer ejemplo del uso de
        b Pug
    footer
      p Alejandro Cerezo
```

### Contenidos

- En la misma línea
- En bloques
- En bloques indicados con un punto

p Texto en la misma línea

p | Texto en un bloque  
| de varias líneas

p. Atajo para utilizar  
un texto en bloque  
de varias líneas

## Inline HTML incluido en el contenido

p. En un **text** de varias líneas se pueden incluir etiquetas HTML

## Comentarios

viernes, 29 de junio de 2018 18:43

- Una sola línea
- Múltiples líneas
- Que se renderizan
- Que no (solo para desarrollo)

```
// Comentario en una sola línea
// - Comentario en una sola línea que no se renderiza
// Comentario en varias líneas
// por que las posteriores a la primera
// están indentadas
// - Comentario en varias líneas
// que no se renderiza
```

# Atributos

viernes, 29 de junio de 2018 18:41

```
html(lang="en")  
p(id="p1", class="principal") Usando varios atributos
```

() junto a la etiqueta, sin dejar espacios

## Atajos (shorthands)

- id
- class
- omitir div

```
p(id="p1", class="principal") Usando varios atributos  
p#p2.principal Usando varios atributos con atajos  
.principal Creando un div con un atajo
```

# Documento básico

viernes, 29 de junio de 2018 18:39

En VSC, el snippet html:5

```
doctype html
html(lang="en")
  head
    meta(charset="UTF-8")
    meta(name="viewport", content="width=device-width, initial-scale=1.0")
    meta(http-equiv="X-UA-Compatible", content="ie=edge")
  title Document
body
```

## Doctypes

doctype
<!DOCTYPE html>
doctype default
<!DOCTYPE html>
doctype html
<!DOCTYPE html>
doctype xml
<?xml version="1.0" encoding="utf-8" ?>
doctype transitional
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" <a href="http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"</a> >
doctype strict
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http:// <a href="http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"</a> >
doctype frameset
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" <a href="http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd"</a> >
doctype 1.1
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" " <a href="http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd</a> ">
doctype basic
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.1//EN" "http:// <a href="http://www.w3.org/TR/xhtml-basic/xhtml-basic11.dtd">www.w3.org/TR/xhtml-basic/xhtml-basic11.dtd"</a> >
doctype mobile
<!DOCTYPE html PUBLIC "-//WAPFORUM//DTD XHTML Mobile 1.2//EN" <a href="http://www.openmobilealliance.org/tech/DTD/xhtml-mobile12.dtd">"http://www.openmobilealliance.org/tech/DTD/xhtml-mobile12.dtd"</a> >

# Bloque de expansión

viernes, 29 de junio de 2018 18:43

Cuando se indentan sucesivos elementos con poco contenido pueden pasarse a una sola línea separándolos con :

# Incorporando datos en plantillas

viernes, 29 de junio de 2018 20:01

## Sintaxis. Definición y uso de variables

Al comenzar con guion (-), lo que sigue se interpreta directamente como JS

```
-var user = "Pepe"  
-let edad = 23
```

p El usuario #{user} tiene #{edad} años

Interpolación

La interpolación también admite operaciones con valores o variables

p El padre de #{user} tiene #{edad + 35} años

En los atributos, pueden usarse las variables sin interpolación (o interpolándolas)

```
-let tipo = "text"  
-let nombre = "usuario"  
input(type=tipo, name=nombre)
```

En las etiquetas también puede usarse ese mismo formato

```
-let content = "Este es un ejemplo de atajo para interpolar"  
p= content
```

# Escapando

viernes, 29 de junio de 2018 20:02

Por defecto, si una variable incluye código HTML, se interpreta como si fuera texto, escapandose los códigos HTML

Proyección contra la inyección  
de código malicioso

```
- let texto = "<script> ... código ... </script>"  
p= texto
```

 <p>&lt;script&gt; ... código ... &lt;/script&gt;</p>

Asumiendo el riesgo, se  
puede evitar el escapado

p!= texto

 <p><script> ... código ... </script></p>

# Envío de las variables al compilador

viernes, 29 de junio de 2018 20:02

Se pueden enviar variables como argumento al compilador, en forma de objeto JSON

```
$ pug file.pug --obj '{"user":"Jose"}'
```

Lo habitual es agrupar las variables en un fichero JSON con todas ellas.  
El parámetro -O permite indicarle al compilador que utilice dicho fichero

```
$ pug file.pug -O 'options.json'
```

# Lógica en plantillas

viernes, 29 de junio de 2018 20:42

Existe dos posibilidades

- Agregar lógica con JavaScript
- Operadores lógicos nativos (Built-in)

```
// - name = "Pepe"
- name = "Juan"
- if (name == "Pepe") {
h1 Hola Pepe
- } else {
h1 Hola #{name}, ¿sabes algo de Pepe?
- }
```

# Condicionales

sábado, 30 de junio de 2018 9:20

## If / else if / else

```
// - name = "Pepe"
- name = "Juan"
if (name == "Pepe")
    h1 Hola Pepe
else
    h1 Hola #{name}, ¿sabes algo de Pepe?
```

## Unless (= if not)

```
- name = "Pepe"
- name = "Juan"
unless (name == "Pepe")
    h1 Hola #{name}, ¿sabes algo de Pepe?
else
    h1 Hola Pepe
```

## Case

```
- name = "Pepe"
- name = "Alicia"
- name = "Juan"
case name
    when "Pepe"
        p Hola Pepe, ¿como estás?
    when "Alicia"
        p Que tal, Alicia
    default
        p Hola #{name}!
```

# Iteraciones

sábado, 30 de junio de 2018 9:35

## each VAL[, KEY] in OBJ

```
- let aSecciones = ['Inicio', 'Porfolio', 'Clientess', 'About']
nav
  ul
    each seccion in aSecciones
      li: a(href="#" + seccion.toLowerCase())= seccion
```

Key toma el valor del índice del array o del nombre de la propiedad, si se itera sobre un objeto

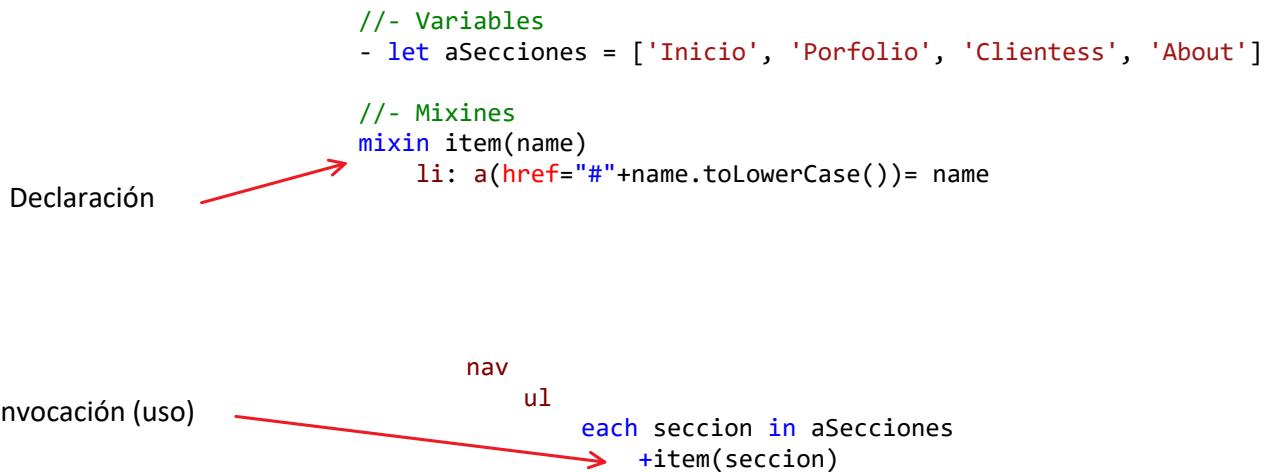
## while

```
nav
  ul
    - i = 0
    while i < aSecciones.length
      li: a(href="#" + aSecciones[i].toLowerCase())= aSecciones[i]
      - i++
```

# Mixins

sábado, 30 de junio de 2018 9:54

## Sintaxis y mecánica



## Use de bloques (block)



# Argumentos indefinidos

sábado, 30 de junio de 2018 9:54

Un mixin puede definir un número indefinido de argumentos y recogerlos en el objeto arguments, como en cualquier función JS, al que accede mediante el prototipo de Array: Array.prototype.slice.call(arguments)

```
mixin argumentos()
- var args = Array.prototype.slice.call(arguments)
  each val in args
    li= val
```

Usando ES6, hacemos esto mismo mediante el *spread operator* (...)

```
mixin argumentos(...args)
  each val in args
    li= val
```

En ambos casos, invocamos el mixin pasándole cualquier número de parámetros

```
ul
+argumentos("uno", "dos", "tres", "cuatro")
```

# Herencia y combinación de plantillas

sábado, 30 de junio de 2018 11:34

La herencia de plantillas se consigue con la palabra **Extends**



Una nueva plantilla extiende / hereda de otra



Para modificar la segunda plantilla, se definen en la plantilla padre bloques (**Block**) que pueden ser modificados (append y prepend) o reemplazados (block )

La combinación de varias plantillas se consigue con la palabra reservada **Include**

Los includes pueden ser

- Ficheros de texto/html
- Ficheros pug



El contenido de un include ¿no puede ser modificado?

# Herencia

sábado, 30 de junio de 2018 11:49

"Plantilla" padre,  
fichero layout.pug

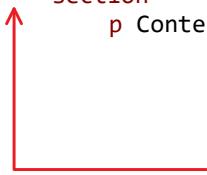
```
<!DOCTYPE html>
html(lang="en")
head
  meta(charset="UTF-8")
  meta(name="viewport", content="width=device-width, initial-
scale=1.0")
  meta(http-equiv="X-UA-Compatible", content="ie=edge")
  title Document
body
header
  h1 Herencia y combinación de plantillas
main
  block main
footer
  p Alejandro Cerezo
```

Zona que podrá ser modificada  
en las plantillas hijas

Herencia:  
fichero inicio.pug

```
extends ./layout.pug ← Referencia a la plantilla padre
```

```
block main
  section
    p Contenido específico de la página
```



Bloque de contenido para la zona predefinida  
previamente en la plantilla padre

append main → Añadiría a continuación del  
bloque, sin modificarlo

prepend main → Añadiría antes del bloque,  
sin modificarlo