Alejandro Jurado Gutiérrez

HTML 5

Manual para programadores

Fundamentos de HTML y CSS

HTML

HTML son las siglas del HyperText Markup Language, lenguaje base del desarrollo de páginas y aplicaciones web. Los navegadores a través de una conexión a un servidor determinado descargan un documento HTML el cual es interpretado por el motor de dichos navegadores y representado en pantalla.

Documentos HTML

Un documento HTML consiste de un árbol de elementos (delimitados por etiquetas de inicio y fin habitualmente) y texto que se distribuye en distintas secciones, y que además puede ser enriquecido con la ejecución de scripts y por diseño gráfico basado en hojas de estilos (CSS):

```
<!DOCTYPE html>
<html>
   <head>
      <title>Título</title>
      <!-- enlaces a documentos externos, js o css -->
      <link rel="stylesheet" href="style.css">
      <!-- metadatos descriptivos (description, keywords, author), encoding
      (charset) o refresh (http-equiv="refresh" content="30") -->
      <meta name="description" content="Página ejemplo">
      <!-- scripts y estilos de la página -->
      <script>…</script>
      <style>…</style>
      <!-- base para referencias y comportamiento por defecto de links -->
      <base href="http://dominio.es/" target="_blank">
   </head>

   <body>
      <!-- comentarios -->
      Contenido
      ...
   </body>
</html>
```

Cada etiqueta de inicio puede contener una serie de atributos que determinan las propiedades del componente HTML:

```
<!-- valor puede ir sin comillas a menos que lleve espacios o ' " `´ <> = -->
<input name=nombre>
<input name='nombre completo'>
<input name="nombre completo">

<!-- atributos vacíos -->
<input name="nombre" disabled="">
<input name="nombre" disabled>

<!-- incorrecto -->
<input name="nombre'>
```

Resulta muy frecuente encontrar etiquetas anidadas. También existen etiquetas vacías (<br>).

Elementos básicos

Elemento	Etiqueta	Comentarios
Cabecera	`<h1>Texto</h1>`	Cabeceras de texto, de h1 a h6 en orden de importancia. Son empleadas por los motores
Párrafo	`<p>Texto</p>`	Delimita un párrafo de texto
Enlace	`<a href=url>Enlace</a>`	Enlace a una url, absoluta o relativa
Imagen	`<img src=url alt=texto width=dim height=px>`	Imagen, los valores de tamaños son opcionales, alt indica texto alternativo. Formatos png, jpg, gif, …
Regla horizontal	`<hr>`	Regla horizontal empleada para separar cabeceras
Salto de línea	` `	Salto de línea, HTML no interpreta los saltos de texto
Comentario	`<!-- -->`	Comentario de una o varias líneas
Comentario condicional	`<!-- [if IE 8] >…` `<![endif] -->`	Porción de código que solamente se ejecutará en el navegador determinado
Frame	`<iframe src=url height=dim weight=dim></iframe>`	Frame con las dimensiones indicadas, en % o px (por defecto)
Tabla	`<table>`	Tabla
Fila	`<tr rowspan=num>`	Fila de una tabla, rowspan para expandir en más de una fila
Columna	`<td colspan=num>`	Columna de una tabla, colspan para expandir en más de una columna
Cabecera de tabla	`<th>`	Cabecera de una tabla, por defecto negrita y centrado
Título de tabla	`<caption>`	Título de la tabla
Lista desordenada	`<ul style="list-style-type:tipo">`	Lista con viñetas según tipo (disc, circle, square, none, …)
Lista ordenada	`<ol type=tipo>`	Lista con índices según tipo (1, A, a, I, i)
Elemento de lista	`<li>`	Elemento de la lista
Lista descriptiva	`<dl>`	Lista con descripciones

Término de lista	`<dt>`	Cada uno de los elementos a describir
Descripción	`<dd>`	Descripción de cada término
Formulario	`<form action=url method=get>` …	Formulario que contiene elementos cuyos valores se envían a la url mediante petición GET o POST
Conjunto de elementos	`<fieldset>` `<legend>texto<legend>…`	Agrupación de elementos relacionados dentro de un formulario, legend indica el título
Elementos de entrada	`<input type=select name=nombre>`	Elemento de entrada, el nombre indica el parámetro que se enviará. Existen diversos tipos
Área de texto	`<textarea name=nombre rows=filas cols=columnas>`	Entrada de texto de varias líneas y columnas, ampliable en algunos navegadores
Lista de selección	`<select name=nombre multiple>…`	Lista de selección, simple o múltiple
Opción de lista	`<option value=valor selected>texto</option>`	Cada opción puede indicar un valor para enviar y un texto para mostrar. Selected indica el actual
Etiqueta	`<label for=id>Texto</label>`	Etiqueta para los componentes de entrada del formulario
Sección bloque	`<div>`	Delimita un bloque de documento, en combinación con css define la estructura
Sección línea	`<span>`	Delimita una sección en línea (no comienza en nueva línea), con css define formato de texto

Caracteres especiales

Algunos caracteres están reservados en HTML como son <, >, espacio, símbolos matemáticos, … o letras con símbolos que no pertenezcan a la codificación de la página. Estos caracteres se denominan entidades y se deben escribir en el código fuente HTML precedidos por un & (&nombre_entidad; o &num_entidad;). Algunos ejemplos aparecen en la siguiente tabla:

Elemento	**Nombre**	**Número**	**Resultado**
Espacio			
Menor que	&lt;	&#60;	<
Mayor que	&gt;	&#62;	>
Ampersand	&amp;	&#38;	&

Euro	&euro;	&#8364;	€
Copyright	&copy;	&#169;	©
A may acentuada	&Aacute;	&#193;	Á
a min acentuada	&aacute;	&#225;	á
Ñ may	&Ntilde;	&#209;	Ñ
ñ min	&ntilde;	&#241;	ñ

Con el soporte de codificación **UTF-8** no deben aparecer problemas con los caracteres de tipo acentuación aunque sean escritos de forma directa.

Enlaces

Además de sobre un texto los enlaces se pueden establecer sobre imágenes:

```
<!-- border 0 para que no aparezca línea alrededor de la imagen en IE -->
<a href="http://www.hubiqus.es">
  <img src="logo.png" alt="logo"
      style="width:36px;height:36px;border:0;">
</a>
```

Se puede establecer el comportamiento a la hora de abrir con el atributo target:

- _blank: abrir en nueva ventana.
- _self: comportamiento por defecto, abrir en la ventana o frame actual.
- _parent: abrir en la ventana o frame padre del actual.
- _top: abrir en ventana completa independientemente del nivel del frame.
- framename: abrir enlace en el frame indicado.

Empleando estilos se puede establecer el comportamiento del enlace:

- a:link: estado normal. Por defecto azul y subrayado.
- a:hover: al pasar por encima.
- a:visited: el enlace ha sido visitado. Por defecto púrpura y subrayado.
- a:active: al pinchar con el ratón sobre el enlace. Por defecto rojo y subrayado.

Además se pueden establecer referencias en la página actual o en una página distinta y enlazar a ellas con un href del tipo #id.

Entrada en formularios

El elemento principal de entrada en un formulario es el input, existen distintos tipos de input cuyo comportamiento se resume en la siguiente tabla:

Elemento	Tipo	Comentarios
Caja de texto	text	Caja de texto normal
Contraseña	password	Caja de texto para contraseñas, enmascara los caracteres
Enviar	submit	Botón de envío de un formulario
Restaurar	reset	Restaura a los valores por defecto
Botón	button	Botón con acción asociada (Javascript, HTML5)
Radio	radio	Botón de tipo radio, agrupa los elementos con mismo nombre
Check	checkbox	Botón de tipo check
Fichero	file	Selección de un fichero, el valor será el nombre del mismo
Imagen	image	Selección de una posición (x,y en px) de una imagen, envía el formulario
Oculto	hidden	Campo no visible de un formulario

Además del tipo, existe una serie de atributos usuales que se puede emplear en los input:

Elemento	Atributo	Comentarios
Nombre	name	Nombre del parámetro que se envía al servidor
Identificador	id	Identificador para hacer referencia a un ítem dentro del documento
Valor	value	Valor por defecto, también indica la etiqueta en submit
Deshabilitado	disabled	El elemento no se puede accionar
Sólo lectura	readonly	Habilitado pero no se puede modificar
Tamaño	size	Tamaño en caracteres
Tamaño máximo	maxlength	Número máximo de caracteres
Seleccionado	checked	Indica si el elemento (radio, checkbox) se encuentra marcado
Extensión	accept	Tipo MIME de los elementos aceptados en un file
URL	src	URL de la imagen a mostrar

Elementos adicionales

Elemento	Etiqueta	Comentarios
Texto formateado	`<pre>`	Muestra el texto con el formato definido en el código, respetando espacios y saltos
Cita	`<quotation>`	Cita textual, el navegador formatea el bloque
Cita corta	`<q>`	Cita corta, se muestra entre comillas
Abreviatura	`<abbr>`	Siglas de una entidad o abreviaturas
Dirección	`<addr>`	Dirección de contacto
Obra	`<cite>`	Título de una obra o trabajo, se muestra como destacado
Cambio dirección	`<bdo>`	Muestra el texto escrito en orden inverso
Texto entrada	`<kbd>`	Texto tipo entrada de teclado
Texto salida	`<samp>`	Texto tipo salida de aplicación
Variable	`<var>`	Texto tipo definición de variable
Código fuente	`<code>`	Formato de código fuente, combinar con <pre> para que respete el formato completo
Mapa de imágenes	`<map><area shape=forma coords=pix href=url>…`	Genera una serie de áreas seleccionables sobre una imagen, con una forma y coordenadas determinadas

Aplicación de estilos

Dentro de una página HTML se puede emplear estilo directamente mediante la utilización del atributo style, siguiendo el formato **propiedad:valor;**.

Algunos valores posibles son los siguientes:

Elemento	Propiedad	Comentarios
Color de fondo	background-color	Color de fondo de un elemento, se pueden especificar colores predefinidos, RGB o hexadecimal
Color de texto	color	Color de texto (cabeceras, párrafos, …)
Fuente	font-family	Fuente de texto, la disponibilidad depende del navegador
Tamaño de fuente	font-size	Tamaño de fuente de texto en pt o %
Alineación	text-align	Tipo de alineación de texto
Ancho	width	Ancho de elemento, por defecto en px
Alto	height	Alto de elemento, por defecto en px
Borde	border	Indica las propiedades (width-style-color) de un borde

Pegar borde	border-collapse	Para que los bordes aparezcan pegados en las celdas de tablas
Espacio borde	border-spacing	Espacio entre celdas de una tabla
Flotar	float	Flotar a la izquierda o derecha sobre un párrafo
Relleno	padding	Relleno de celdas de una tabla

También existen etiquetas de formato de texto directamente sobre el HTML:

Elemento	**Etiqueta**	**Comentarios**
Negrita	<b>	Texto negrita
Importante		Texto importante, el navegador lo mostrará como negrita
Cursiva	<i>	Texto cursiva
Énfasis		Texto enfatizado, el navegador lo mostrará como cursiva
Marcado	<mark>	Texto resaltado
Pequeño	<small>	Texto más pequeño
Borrado	<del>	Texto tachado
Insertado	<ins>	Texto subrayado
Subíndice	<sub>	Texto subíndice
Superíndice	<sup>	Texto superíndice

Cualquier etiqueta de HTML puede llevar el atributo **class**, que indica un estilo definido específicamente para un elemento de una clase concreta, es decir, que para una misma clase (por ejemplo div) podemos tener estilos distintos (div.header, div.content, div.footer).

Referencias

http://www.w3.org/TR/html5/introduction.html

http://www.w3.org/TR/html5/syntax.html

http://www.w3schools.com/html/default.asp

http://www.w3schools.com/quiztest/quiztest.asp?Qtest=HTML

http://www.w3schools.com/quiztest/quiztest.asp?qtest=CSS

XHTML

XHTML son las siglas del Extensible HyperText Markup Language. Se trata de una versión más estricta del HTML donde las páginas son definidas en formato XML de forma que se garantiza que el documento se encuentre bien formado.

La principal vocación de XHTML es la de la eliminación de errores comunes en HTML y la simplificación de los motores presentes en los navegadores, al tratarse de documentos previamente validados. De la misma forma permite el procesamiento por parte de parseadores de XML para extraer información, y una base para el establecimiento de la web semántica.

Documentos XHTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
   <head>
      <title>Título</title>
   </head>

   <body>
      Contenido
      ...
   </body>
</html>
```

Las diferencias con HTML son las siguientes:

- DOCTYPE y xmlns son obligatorios.
- Todas las secciones son obligatorias (head, title y body).
- Todas las etiquetas y atributos en minúscula.
- Todas las etiquetas cerradas y bien anidadas.
- Todos los atributos con valor y entre comillas.

Los valores de DOCTYPE que definen un documento son:

Tipo	Valor	Comentarios
HTML 4.0.1 Estricto	`<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">`	Excluye elementos de presentación, deprecados y frameset
HTML 4.0.1 Transicional	`<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">`	Excluye solamente frameset

HTML 4.0.1 con frameset	`<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">`	
XHTML 1.0 Estricto	`<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">`	Excluye los elementos de presentación, deprecados y frameset
XHTML 1.0 Transicional	`<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">`	Excluye solamente frameset
XHTML 1.0 con frameset	`<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">`	
XHTML 1.1	`<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">`	Permite añadir módulos externos
HTML 5	`<!DOCTYPE html>`	No soporta los elementos de presentación, deprecados o frameset por defecto

En la siguiente web se puede realizar un chequeo de validez de un documento XHTML:

http://validator.w3.org

Referencias

http://www.w3schools.com/html/html_xhtml.asp

https://es.wikipedia.org/wiki/XHTML

CSS

CSS son las siglas de Cascading Style Sheets, lenguaje empleado para describir el estilo de una página web indicando cómo se debe mostrar cada uno de los elementos HTML que la componen.

Las propiedades de estilos que se pueden manipular con CSS son múltiples, incluyendo el diseño, la configuración de presentación (**layout**) y las diferentes vistas disponibles en función del tipo y tamaño de dispositivo. Además presenta dos grandes ventajas:

- Permite configurar el estilo de todo el conjunto de páginas que componen un portal web a través de un documento único.
- Independiza el contenido HTML de la presentación, por lo que actualizar el estilo de la página consiste en modificar la hoja de estilo de referencia (o emplear una distinta).

Documentos CSS

La definición de estilos se realiza a través de una serie de reglas:

```
body {
    background-color: steelblue;
    font-family: arial;
}

h1 {
    color: #5AA6ED;
    text-align: center;
}

p {
    font-size: 1.5em;
}
```

Cada regla del estilo está definida por un selector de elemento HTML y una serie de declaraciones de propiedades de estilo y su valor correspondiente separadas por punto y coma.

La definición de reglas de estilo habitualmente se realiza en un documento .css externo, aunque también es posible aplicar los estilos sobre el documento HTML (`<style>…</style>`) o directamente sobre el elemento en concreto al que se desea aplicar formato a través del atributo **style**. Cuando el estilo se encuentra definido externamente es necesario hacer referencia al mismo desde el documento HTML:

```
<link rel="stylesheet" type="text/css" href="grid.css">
```

Los selectores de estilo se pueden aplicar principalmente a:

Tipo	Valor	Comentarios
Elementos HTML	`nombre`	Se aplica a todos los elementos de ese tipo
Identificadores	`#id`	Se aplica únicamente al elemento cuyo atributo id coincide
Clases	`.class`	Se aplica a las clases cuyo atributo class lo contiene Un elemento puede hacer referencia a más de un class

Los selectores duplicados se pueden agrupar:

```
/* Varios selectores agrupados */
h1, h2 {
    color: #5AA6ED;
    text-align: center;
}
```

Orden de aplicación

Existe la posibilidad de emplear varias hojas de estilos o de emplear estilos dentro del documento y además de incluir estilos directamente en los elementos HTML. En caso de que aparezcan distintos estilos para un selector existe un orden de preferencia que indica qué estilo se aplicará:

1. Estilo dentro de un elemento HTML, sobrescribe a cualquier otro estilo.
2. Estilo de hoja o del documento, en orden de lectura (el último leído se aplica).
3. Estilo por defecto del navegador.

Colores

Los valores de colores se pueden especificar de varias formas, lo más habitual es realizarlo en RGB o Hexadecimal:

Tipo	Valor	Comentarios
Nombre	`DarkSeaGreen`	Existen 140 nombres de colores estándares en HTML
RGB	`rgb(255,14,20)`	Valor RGB (red-green-blue) - 0 a 255
HEX	`#8FBC8F`	Valor RGB en hexadecimal
HSL	`hsl(0,100%,100%)`	Valor HSL (hue-saturation-lightness)
RGBA	`rgba(255,14,20,0.4)`	Valor RGB con Alpha (opacidad) - 0 (transparente) a 1
HSLA	`hsl(0,100%,100%,0)`	Valor HSL con Alpha

Los colores se pueden aplicar a varios tipos de atributos:

Tipo	Atributo	Comentarios
Color	`color`	Color de fuente
Borde	`border-color`	Color del borde del elemento, necesario **border-style**
Fondo	`background-color`	Color del fondo del elemento

Fondos

Además de poder emplear un color de fondo resulta habitual utilizar una imagen o patrón. En ambos casos la configuración se realiza a través de URL:

```
body {
    background-image: url("bg.jpg");
}
```

La imagen de fondo se repite por defecto en forma de mosaico según el tamaño de la misma, se puede configurar el tipo de repetición del fondo o fijar su posición y desplazamiento:

Atributo	Valor	Comentarios
`background-repeat`	`no-repeat`	Mostrar la imagen sólo una vez
	`repeat-x`	Repetir horizontalmente
	`repeat-y`	Repetir verticalmente
	`repeat`	Por defecto
`background-position`	`right top`	Ubicación del comienzo del fondo
`background-attachment`	`fixed`	No desplazar con el scroll
	`scroll`	Por defecto

Las posiciones del fondo pueden tomar valores relativos y absolutos:

Atributo	Valor	Comentarios

Posición	`left top|center|bottom`
	`right top|center|bottom`
	`center top|center|bottom`
		center por defecto si sólo se especifica uno

Porcentaje	`0% 50%`	Posición horizontal y vertical 50% por defecto si sólo se especifica uno
Coordenadas	`30px 60px`	Posición horizontal y vertical en unidades 50% por defecto si sólo se especifica uno Se pueden combinar unidades y %

Formato abreviado:

```
/* Orden: color image repeat attachment position*/
body {
    background: url("bg.jpg") repeat-x left top;
}
```

Bordes

Los bordes de elementos en CSS pueden ser configurados de diversas maneras, existen diversos valores para el estilo a emplear, además de poder establecer color o tamaño:

Atributo	Valor	Comentarios
border-style	solid	Sólido
	dotted	Puntos
	dashed	Líneas
	double	Doble
	groove	3D hundido
	ridge	3D relieve
	inset	Hundido
	outset	Relieve
	hidden	Oculto
	none	Sin borde
border-width	thin|medium|thick	Ancho de borde, valor predefinido o unidad
border-color	red	Color del borde
border-radius	3px	Redondeo de las esquinas

El atributo border-style es imprescindible para que el resto se aplique.

Es posible definir un estilo, ancho y color distintos para cada lado, teniendo en cuenta que el orden es el siguiente:

- Si se establecen los 4 valores: **top right bottom left**.
- Si se establecen 3 valores: **top right|left bottom**.
- Si se establecen 2 valores: **top|bottom right|left**.

Formato abreviado:

```
/* Orden: width style color */
p {
    border: 2px solid black;
}
```

Además del borde de un elemento es posible configurar su espacio exterior (**outline**). El comportamiento es similar al borde solamente que se dibuja por fuera de éste.

```
/* Orden: width style color */
p {
    outline: 2px solid black;
    outline-offset: 5px;
}
```

El outline no admite radius, por contra se puede configurar un **offset** con la distancia hasta el borde.

Margen y relleno

El margen indica el espacio alrededor de un elemento, mientras que el relleno se refiere al espacio entre el borde y su contenido interior.

Atributo	Valor	Comentarios
margin-top| margin-right| margin-bottom| margin-left	100px	Valor
	10 %	Porcentaje
	auto	Centra horizontalmente en el contenedor
	inherit	Heredado del elemento padre
padding-top| padding-right| padding-bottom| padding-left	100px	Valor
	10 %	Porcentaje
	inherit	Heredado del elemento padre

Formato abreviado:

```
/* Orden: top right bottom left */
p {
    margin: 2px 0 3px 0;
    padding: 1px 1px 1px 1px;
}
```

Cuando el margen inferior de un elemento y el superior del siguiente se establecen ocurre un colapso, que implica que sólo se aplica el mayor de ambos, no la suma.

Con respecto al ancho de los elementos, éste se calcula en base al relleno, borde y el contenido, a menos que se establezca la propiedad box-sizing:

```
div {
    width: 300px;
    padding: 100px;
    box-sizing: border-box;
}
```

Los atributos **width** y **height** se emplean para establecer ancho y alto respectivamente (en unidades o porcentajes). En caso de que el espacio establecido no quepa en la pantalla del navegador se introducen scrolls. Para evitar la aparición del scrolling se pueden emplear **max-width|max-height** en lugar de las anteriores.

También se encuentran disponibles las propiedades **min-width|min-height** que en combinación con las anteriores indican el mínimo espacio a ocupar.

El atributo **overflow** indica qué ocurre cuando el contenido es mayor al contenedor en el que se encuentra:

- **visible**: por defecto, el contenido que sale del contenedor es visible.
- **hidden**: oculta el contenido excedente.
- **scroll**: añade barras de scroll al contenedor.
- **auto**: similar al anterior, pero solamente añade el scroll cuando es necesario.

Es posible establecer el overflow para ancho y alto de forma independiente:

```
div {
    overflow-x: auto;
    overflow-y: hidden;
}
```

Formato de texto

A continuación se enumeran algunas de las propiedades que permiten formatear el contenido de elementos textuales HTML:

Atributo	Valor	Comentarios
font-family	Arial	Nombre de fuente
	"Times New Roman"	Nombre de fuente (más de una palabra)
	serif|sans-serif|monospace	Familia genérica
	Arial, Helvetica, sans-serif	Combinación

font-style	normal	Por defecto
	italic	Cursiva
font-size	16px|1em	Tamaño en px o em (px/16)
font-weight	normal	Por defecto
	bold	Negrita
font-variant	normal	Por defecto
	small-caps	Todo mayúsculas (tamaño mayor para may)
text-align	left	Izquierda, por defecto
	right	Derecha
	center	Centrado
	justify	Justificado
text-indent	16px	Indentación de la primera línea
text-decoration	none	Por defecto
	underline	Subrayado
	overline	Sobrerayado
	line-through	Tachado
text-transform	none	Por defecto
	uppercase	Mayúscula
	lowercase	Minúscula
	capitalize	Mayúscula en el inicio de cada palabra
text-shadow	2px 2px gray	Sombra
direction	ltr	Por defecto
	rtl	Derecha a izquierda
line-height	1.5	Espaciado entre líneas
letter-spacing	16px	Espacio entre letras
word-spacing	16px	Espacio entre palabras

Enlaces

Los enlaces pueden configurar distintos estilos para los distintos estados existentes:

- link: por defecto.
- visited: enlace que ya ha sido pulsado.
- hover: ratón encima.
- active: pulsación.

El estilo aplicado enlace se realiza de forma similar a la ya comentada para textos. Destacar que el orden al establecer el estilo de los estados en el documento debe ser como el anterior (active después de hover, hover después de link y visited)

```
a:link {
    background-color: blue;
}

a:visited {
    background-color: pink;
}

a:hover, a:active {
    background-color: cyan;
}
```

También es posible configurar el tipo de cursor sobre el enlace o cualquier texto:

```
a:hover {
    cursor: auto|crosshair|default|help|move|pointer|text|wait…;
}
```

Listas

Estos son los atributos para formato simple de listas:

Atributo	Valor	Comentarios
list-style-type	disc	Por defecto
	circle	Círculo
	square	Cuadrado
	decimal	Número
	lower-alpha	Letra (igual a lower-latin)
	lower-roman	Número romano
	upper-alpha	Letra mayúscula
	none	Ninguno
	…	
list-style-image	url('sq.gif')	Imagen desde URL
list-style-position	outside	Por defecto
	inside	Marcador dentro del contenido

Para determinar un estilo completo de lista es posible modificar las propiedades del espacio que ocupa la lista y cada uno de sus componentes:

```
ul {
    background-color: red;
    padding: 10px;
}

ul li.a {
   padding: 5px;
    margin: 5px 35px;
}

ul li.a {
    background-color: pink;
}

ul li.b {
    background-color: salmon;
}
```

Tablas

En tablas principalmente será necesario configurar los bordes, y el formato del texto que contienen, incluyendo algún efecto más específico que comentaremos:

Atributo	Valor	Comentarios
border	1px solid black	Borde de tabla, tr o th
border-collapse	separate	Por defecto
	collapse	Borde simple
border-spacing	2px	Distancia entre líneas de borde
vertical-align	top|bottom|middle	Alineación vertical de celda
tr:hover		Efecto de fila
tr:nth-child(even|odd)		Estilo de fila (par|impar)

Visualización y posición

Los elementos de un documento HTML poseen unas propiedades de visualización por defecto, que afectan a cómo se muestran en pantalla y además al espacio que ocupan. Existen principalmente dos tipos de visualización para elementos:

- **block**: el elemento ocupa una única línea, cualquier elemento a continuación se mostrará en la línea siguiente. Ocurre con los form, table, ul, p, h1, h2, …
- **inline**: el resto de elementos se muestra en la línea actual a continuación del texto o componentes de la página, como ocurre con los a, img, span, …
- **inline-block**: los elementos van ocupando una línea hasta que el ancho no lo permite, entonces el comportamiento es block. Útil para crear **grids** de elementos.

Se puede modificar el comportamiento de un elemento a través de su estilo:

```
li {
    display: inline;
}

a {
    display: block;
}
```

Se incluyen además valores que indican que un elemento se encuentra oculto:

- **none**: el elemento no se muestra ni ocupa espacio, es como si no estuviera.
- **hidden**: el elemento es invisible para la página, aunque su espacio se mantiene.

```
/* hidden es un valor del atributo visibility, no de display */
.invisible {
    visibility: hidden;
}
```

Respecto a la posición de los elementos:

- **static**: es el valor por defecto de todo elemento.
- **relative**: relativa a su posición por defecto. Se modifica empleando los atributos left|right|bottom|top.
- **fixed**: posición fija aún cuando se realice scroll de la página. No ocupa espacio.
- **absolute**: posición fija respecto al ancestro posicionado (no static) más cercano. Si el contenedor no está posicionado es similar a fixed (fija respecto a body).

Los elementos con posición fija pueden aparecer sobre otros componentes de la página. El atributo **z-index** ordena las capas:

```
div.fixed {
    position: fixed;
    top: 0;
    right: 0;
    width: 200px;
    z-index: -1;
}
```

A través del atributo float se puede establecer dónde se colocan los objetos. Los valores posibles son **left**, **right**, **none** (por defecto), **inherit** (heredado).

De esta forma los elementos son posicionados estableciendo un **layout**, aunque en su utilización más simple, float sirve para establecer cómo un texto se ubica con respecto a una imagen.

El atributo **clear** se emplea para aquellos elementos que no queremos que floten como los anteriores. Los valores posibles son **left**, **right**, **both, none** (por defecto), **inherit** (heredado). En el caso de left por ejemplo, indica que el elemento no flote a la izquierda como sus antecesores.

Imágenes e Iconos

Para las imágenes es posible establecer sus dimensiones a través de width y height, un **clip** permite recortar la imagen a un tamaño definido por un rectángulo:

```
/* Orden de clip: top right bottom left */
img {
    position: absolute;
    clip: rect(0, 20px, 20px, 0);
}
```

Una imagen permite definir el nivel de transparencia:

```
img {
    opacity: 0.5;
}
```

Además de imágenes desde recursos y URLs se pueden cargar iconos de recursos web, como por ejemplo Google, añadiendo la referencia de estilo:

```
<link rel="stylesheet"
      href="https://fonts.googleapis.com/icon?family=Material+Icons">
```

Los iconos se emplean en HTML con la etiqueta i y admiten modificadores de tamaño y color:

```
<i class="material-icons" style="font-size:48px;color:gray;">save</i>
```

Combinaciones, Atributos y Pseudo-elementos

Las combinaciones especifican relación entre elementos:

Combinación	Ejemplo	Comentarios
(espacio)	tr td	Afecta a los elementos dentro del principal

>	tr > td	Afecta a los hijos inmediatos del principal
+	tr + tr	Afecta a los elementos inmediatamente adyacentes
~	tr ~ tr	Afecta a todos los elementos adyacentes

Cuando hablamos de enlaces vimos algunas pseudo-clases que se podían aplicar, como hover, link, visited y active. En el caso de hover, se puede emplear para otro tipo de elementos (div por ejemplo).

Existe otra serie de pseudo-clases para aplicar estilo a elementos en función de su estado:

Valor	Ejemplo	Comentarios
first-child	table tr:first-child	Afecta al primer hijo del principal
last-child	table tr:last-child	Afecta al último hijo del principal
nth-child	table tr:nth-child(2)	Afecta a la posición indicada (even/odd para pares o impares)
nth-last-child	table tr:nth-last-child(2)	Afecta al hijo de la posición indicada empezando por el final
empty	div:empty	Afecta a los elementos vacíos
only-child	div:only-child	Afecta elementos que son hijos únicos de un padre
only-of-type	div:only-of-type	Afecta a los elementos que son el único de un tipo concreto
checked	input:checked	Afecta a los elementos con valor checked
disabled	input:disabled	Afecta a los elementos con valor disabled
enabled	input:enabled	Afecta a los elementos que no están deshabitados
focus	input:focus	Afecta al elemento con el foco
in-range	input:in-range	Afecta a los elementos dentro de un rango
out-of-range	input:out-of-range	Afecta a los elementos fuera de un rango
invalid	input:invalid	Afecta a los elementos con valor no válido
valid	input:valid	Afecta a los elementos con valor válido
optional	input:optional	Afecta a los elementos no requeridos
required	input:required	Afecta a los elementos requeridos

read-only	input:read-only	Afecta a los elementos con valor read-only
read-write	input:read-write	Afecta a los elementos que no son read-only
…		

Existen además pseudo-elementos que también afectan a la presentación:

Valor	Ejemplo	Comentarios
after	p::after	Estilo después del elemento
before	p::before	Estilo antes del elemento
first-letter	p::first-letter	Estilo para la primera letra
first-line	p::first-line	Estilo para la primera línea
selection	p::selection	Afecta al contenido seleccionado por el usuario

El caso de after se suele emplear para realizar un clearfix en caso de disponer de contenedores con elementos flotantes que puedan ocasionar un overflow:

```
.clearfix::after {
    content: "";
    clear: both;
    display: table;
}
```

Por último, los selectores de atributos sirven para indicar comportamiento del estilo en función de valores concretos de éstos:

Valor	Ejemplo	Comentarios
atributo	img[alt]	Elementos que contienen un atributo
atributo="valor"	img[alt="logo"]	Elementos cuyo valor coincide
atributo~="valor"	img[alt~="logo"]	Elementos cuyo valor contiene la palabra completa (o separada por espacios)
atributo|="valor"	img[alt|="logo"]	Elementos cuyo valor comienza por palabra completa
atributo^="valor"	img[alt^="logo"]	Elementos cuyo valor comienza por la cadena
atributo$="valor"	img[alt$="logo"]	Elementos cuyo valor termina por la cadena
atributo*="valor"	img[alt*="logo"]	Elementos cuyo valor contiene la cadena

Los selectores se suelen emplear para el estilo de componentes de formularios:

```
input[type="text"]:invalid {
    border: 5px solid red;
}
```

Referencias

https://www.w3schools.com/css/default.asp

https://www.w3schools.com/colors/colors_names.asp

https://www.w3schools.com/cssref/css_units.asp

https://www.w3schools.com/cssref/default.asp

https://www.w3schools.com/css/css_align.asp

https://www.w3schools.com/css/css_form.asp

https://www.w3schools.com/css/tryit.asp?filename=trycss_form_responsive

https://www.w3schools.com/css/tryit.asp?filename=trycss_website_layout_blog

HTML5

Introducción

HTML5 es la quinta revisión del lenguaje de marcado de la web donde se recogen la sintaxis y las APIs para poder emplear HTML y XHTML en la definición y programación de páginas webs avanzadas. Como nuevos elementos más destacados se encuentran los siguientes:

- Elementos semánticos
- Controles de formularios
- Elementos gráficos y multimedia

También incorpora una serie de nuevas APIs de programación entre las que destacan:

Paquete	Comentarios
Geolocalización	Obtener de forma sencilla la localización actual. Requiere permisos de usuario.
Arrastrar y Soltar	Permite que cualquier elemento sea arrastable.
Almacenamiento local	Guardar hasta 5Mb de información local al navegador. Nunca se transmite al servidor.
Caché	Posibilita crear una versión offline de la aplicación web.
Web Workers	Ejecución de procesos de la web en segundo plano
SSE	Obtener actualizaciones automáticas desde un servidor

La peor noticia de HTML5 es que actualmente no dispone de compatibilidad completa con los navegadores existentes. En la siguiente web se puede realizar un chequeo de compatibilidad de los distintos elementos de HTML5 con los navegadores disponibles:

https://html5test.com

Documentos HTML5

Un documento HTML5 es en esencia un documento HTML convencional que puede disponer de elementos incorporados en esta versión y que en su configuración más básica posee el siguiente aspecto:

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="UTF-8">
      <title>Título</title>
   </head>

   <body>
      Contenido
      ...
   </body>
</html>
```

Composición en HTML5

Existe una serie de etiquetas introducidas en HTML5 que permiten estructurar el documento. La idea es emplear etiquetas que por sí mismas determinen las distintas secciones de la página de forma equivalente a los div.

Elemento	Etiqueta	Comentarios
Cabecera	<header>	Delimita la cabecera (logos, accesos directos, …)
Grupo de cabeceras	<hgroup>	Agrupación de las distintas cabeceras para no afectar al SEO
Navegación	<nav>	Navegación principal de la página, normalmente delimitado con
Barra lateral	<aside>	Información no principal o complementos
Sección	<section>	Secciones principales de la página
Artículo	<article>	Cada uno de los artículos que componen una sección
Pie	<footer>	Pie de página (copyrights, sitemap, …)

Para que estas etiquetas tengan efecto es necesario que se combinen con un css, no disponen de significado estructural por sí mismas, simplemente organizativo.

Referencias

http://www.w3.org/TR/html5/

http://www.w3schools.com/html/html5_intro.asp

https://es.wikipedia.org/wiki/HTML5

Formularios

Una de las principales características de HTML5 es la presencia de nuevos elementos en los formularios y nuevos atributos para éstos. Los nuevos elementos de HTML5 son los siguientes:

Elemento	Etiqueta	Comentarios
Lista de datos	<input list=lista> <datalist id=lista>…	Valores predefinidos para un input, list indica el id del datalist. No funciona en Safari
Opción de lista	<option value=valor label=texto>	Cada una de las opciones de un datalist, label es una etiqueta explicativa
Generación de clave	<keygen name=nombre>	Genera un par de claves, la privada se guarda en local y la pública se envía. No funciona en IE
Elemento de salida	<form oninput=accion> <output name=out for=ids>	Realiza el cálculo definido por la acción al modificar los valores indicados. No funciona en IE

Nuevos tipos de elementos de entrada, en caso de no ser soportado por el navegador aparece como de tipo text:

Elemento	Tipo	Comentarios
Número	number	Selector de un número, habitualmente en un rango (min/max)
Rango	range	Selector de rango, entre un min y un max según el incremento (step) indicado
Fecha	date	Selector de fecha
Hora	time	Selector de hora
Mes	month	Selector de mes
Semana	week	Selector de semana
Correo	email	Entrada de texto tipo email
Teléfono	tel	Entrada de texto tipo teléfono
URL	url	Entrada de texto tipo URL
Color	color	Diálogo de selección de color, devuelve el valor hexadecimal
Búsqueda	search	Entrada de palabras para realizar una búsqueda, agrega control eliminar

Además se han añadido atributos para los input:

Elemento	Atributo	Comentarios
Autocompletar	autocomplete	Habilitar (on/off) el autocompletar de los input, también sobre form

Foco automático	autofocus	Establece el foco en un input al cargar la página
Requerido	required	Indica que un campo es requerido
Ayuda	placeholder	Ayuda para completar un campo (hint)
Dirección texto	dirname	Nombre del parámetro que indica la dirección del texto de un input
Formulario	form	Formulario al que pertenece un campo que se encuentra fuera
Acción	formaction	Acción distinta del action del form que se invoca al hacer submit
Encriptación	formenctype	Cambia tipo de encriptación del formulario al enviar
Método	formmethod	Cambia método de envío del formulario
No validar	formnovalidate	No valida campos del formulario al enviar
Destino	formtarget	Tipo de destino del formulario (como en los links)
Ancho	height	Ancho para input de tipo imagen
Alto	width	Alto para input de tipo imagen
Mínimo	min	Valor mínimo en campos numéricos, de fecha y de rango
Máximo	max	Valor máximo en campos numéricos, de fecha y de rango
Incremento	step	Valor del incremento en campos numéricos y de rango
Múltiple	multiple	El usuario puede indicar varios valores, por ejemplo en un file
Lista de valores	list	Establece la lista de valores predefinidos para un input
Patrón	pattern	Patrón de entrada, con title se indica la ayuda para completar

La mayoría de los elementos y atributos nuevos no van a funcionar en Internet Explorer (nunca en versiones antiguas) ni en Safari. La mejor apuesta en estos momentos es Google Chrome.

Referencias

http://www.w3schools.com/html/html_form_input_types.asp

http://www.w3schools.com/tags/tag_input.asp

Geolocalización

Localizar la posición actual del usuario a través del navegador, empleando técnicas de localización por red (Wifi) o GPS en caso de dispositivos móviles, de manera que la información que se presenta al usuario pueda basarse en sus preferencias locales.

Requiere siempre de permisos de usuario, en el caso de Chrome requiere además de que el sitio web sea https y se encuentre habilitado por el usuario en las preferencias de localización.

Es necesario incluir un método Javascript para obtener la posición:

```
function localizacion() {
   if (navigator.geolocation) {
      navigator.geolocation.getCurrentPosition(function (position) {
         var latitud = position.coords.latitude;
         var longitud = position.coords.longitude;
         alert("(" + latitud + "," + longitud + ")");
      });
   }else{
         alert("Tu navegador no permite la localización");
   }
}
```

Elementos adicionales

La variable position posee información adicional a latitud y longitud siempre que el dispositivo lo permita:

Elemento	Atributo	Comentarios
Precisión	accuracy	Precisión de la posición, siempre lo devuelve
Altitud	altitude	Metros sobre el nivel del mar
Precisión altitud	altitudeAccuracy	Precisión de la altitud
Rumbo	heading	Grados de diferencia con el norte (dirección agujas del reloj)
Velocidad	speed	Velocidad actual en m/s
Fecha/hora	timestamp	Sello temporal de la respuesta

Para permitir examinar la posición en tiempo real de forma similar a los navegadores existe un método **watchPosition()** que devolverá la nueva posición cada vez que se produzca un cambio. Su utilización es igual a la de **getCurrentPosition()**. Para terminar de realizar la navegación se emplea **clearWatch()**.

Referencias

http://www.w3schools.com/html/html5_geolocation.asp

JavaScript

Fundamentos de JavaScript

Se trata de un lenguaje de programación que se interpreta del lado del cliente, para permitir una mayor funcionalidad y mejorar la experiencia de usuario.

La inclusión de Java en su nombre se debe a que la sintaxis es similar a la de este lenguaje, aunque NO es parte de la especificación Java de Oracle.

Cada navegador ejecuta JavaScript con un motor propio, de forma que análogamente a los problemas de compatibilidad o interpretación que ocurren con HTML o CSS existen algunos matices que pueden resultar problemáticos en función del navegador seleccionado. Además existe la posibilidad de desactivar el complemento de Javascript en los navegadores, en tal caso la página debería mantener su funcionalidad o hacerlo de forma complementaria.

Introducción

Una de las características principales de JavaScript es la capacidad de modificar los atributos y el contenido HTML de los elementos de una página:

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="UTF-8">
      <title>JavaScript</title>
      <script>
         function cambiar(){
            document.getElementById("demo").innerHTML = "Nuevo
            contenido";
         }
      </script>
   </head>

   <body>
      <p id="demo">Contenido original</p>

      <input type="button" onclick="cambiar()" value="Cambiar">
      <script>
            document.getElementById("demo").style.fontSize = "24px";
      </script>
   </body>
</html>
```

También es posible ocultar o mostrar la información:

```
document.getElementById("demo").style.display = "none";
document.getElementById("demo").style.display = "block";
```

Además de de ubicar código JavaScript en head o body, es habitual emplear archivos JavaScript que recogen las funciones que se van a emplear en una o varias páginas, en tal caso habrá que realizar una referencia a dicho archivo desde la página:

```
<script src="funciones.js"></script>
```

Sintaxis

La sintaxis de JavaScript es similar a Java (espacios, líneas, terminación con ;, comentarios, literales, …). Las variables no tienen tipo definido, se declaran con la palabra clave **var** o **let**:

```
var saludo = "HOLA";

//Las variables tienen comportamiento dinámico
saludo = 5;

if (hora > 20) {
   //Variable local al bloque
   let despedida = 'ADIOS';
}
//La variable despedida no accesible tras el bloque
saludo = "BUENAS";
```

En el caso de constantes se emplea la palabra clave **const**.

Una variable sin inicializar toma el valor y tipo **undefined**. También existe el valor null en JavaScript pero se considera un objeto. Ambos sirven para vaciar una variable.

```
typeof undefined              // undefined
typeof null                   // object
null === undefined            // false
null == undefined             // true
```

Los operadores son los habituales, así como las reglas de precedencia:

Tipo	Operadores
Matemático	+ - * / % ++ --
Asignación	+= -= *= /= %= =
Comparación	> < >= <= == === != !== ?
Tipo	instanceof typeof
Lógicos	&& || ! & | ~ ^ << >>

Los bloques que se pueden emplear en código JavaScript son los siguientes. Se permite la utilización de break y continue:

Bloque	Sintaxis

if	`if (condicion1) {`
	`  …`
	`} else if (condicion2) {`
	`  …`
	`} else {`
	`  …`
	`}`

switch	`switch(expresion) {`
	`  case n1: … break;`
	`  case n2: … break;`
	`  default: …`
	`}`

for	`for (inicializacion; incremento; condicion) {`
	`  …`
	`}`

for in	`for (item in items) {`
	`  …`
	`}`

while	`while (condicion) {`
	`  …`
	`}`

do while	`do {`
	`  …`
	`} while (condicion);`

try	`try {`
	`  …`
	`} catch(err) {`
	`  …`
	`} finally {`
	`  …`
	`}`

Funciones

Las funciones en JavaScript representan la mejor forma de escribir código que vaya a ser compartido o reutilizado en varias partes de la página y con distintos argumentos.

```
function funcion(param1, param2) {
   …
   return resultado;
}
```

Cualquier función definida en una página o en un archivo importado por la misma puede ser llamada desde código script, o desde un evento de página.

Hay que tener en cuenta que en JavaScript se pueden usar las funciones a su vez como valores, en asignaciones o argumentos para otras funciones.

Eventos

La ocurrencia de eventos HTML puede provocar la ejecución de scripts, en la siguiente tabla aparecen los principales eventos de componentes HTML que pueden ejecutar código directamente o en el mejor caso invocar a una función:

Evento	Descripción
onload	La página se ha cargado, similar a colocar código script en body
onclick	Al pulsar un elemento como botón o enlace
onchange	Al modificar el valor de un elemento
onmouseover	El ratón pasa sobre un elemento
onmouseout	El ratón sale del elemento
onkeydown	Al pulsar una tecla

Objetos

La definición de objetos en JavaScript se realiza de forma implícita al inicializar las variables (formato JSON):

```
var persona = {nombre:"Alejandro", apellidos:"Jurado", edad:28};
```

En la definición del objeto también se pueden incluir métodos:

```
var persona = {nombre:"Alejandro", apellidos:"Jurado", edad:28,
     saluda:function(persona){
             return this.nombre + " dice Hola a " + persona.nombre;
     }
};
```

De esta forma ya tenemos definida una clase y la instancia de un objeto de ésta. Con esta forma de definición y creación de objetos no es necesario reservar memoria, aunque se pueden definir constructores y emplearlos a través de **new**:

```
function Persona(nombre, apellidos, edad) {
    this.nombre = nombre;
    this.apellidos = apellidos;
    this.edad = edad;
}
var p = new Persona("Juan", "Suárez", 50);
```

Para clases definidas en JavaScript también es posible utilizar la llamada al constructor (**new** `String("HOLA")`) pero se desaconseja por cuestiones de rendimiento y efectos no deseados.

Los atributos y métodos de un objeto son accedidos con la notación punto o notación tipo array:

```
var edad = persona.edad;
var nombre = persona["nombre"];
```

La definición de objetos es dinámica en JavaScript:

```
p.ciudad = "Málaga";
Persona.ciudad = "Málaga";
```

Arrays

La definición de arrays en JavaScript se realiza de forma similar a los objetos. Los arrays pueden tener varias dimensiones y sus valores pueden ser heterogéneos:

```
var personas = ["Alejandro", "Juan", "Fran"];
```

Los valores de un array se acceden a través de su índice:

```
var nombre = personas[0];
```

Los arrays son dinámicos en JavaScript:

```
var personas = ["Alejandro", "Juan", "Fran"];
personas.push("Javier");
personas[personas.length] = "Ana";
```

Los métodos principales son los siguientes:

Método	Descripción
toString()	Devuelve los elementos separados por comas
valueOf()	Devuelve el array como valor primitivo (igual anterior)
join(sep)	Devuelve los elementos separados por el separador
pop()	Elimina y devuelve el último elemento
push(item)	Agregar al final, devuelve la longitud del array

shift()	Elimina y devuelve el primer elemento
unshift(item)	Agregar al inicio, devuelve la longitud del array
concat(arr1,…,arrN)	Concatenar con otros arrays
splice(index, n, i1,…iN)	Inserta a partir del índice eliminando n elementos los ítems que se indiquen. Si no se indica ningún elemento y n=1 elimina el elemento del índice
slice(index)	Devuelve un nuevo array a partir del índice
sort(f)	Ordena el array, si no se especifica función se aplica orden alfabético
reverse()	Orden alfabético inverso

Cadenas

Las cadenas de caracteres se pueden escribir con comillas dobles o simples:

```
var nombre = "Alejandro";
var apellidos = 'Jurado';
var alias = "El 'profe'";
var length = nombre.length + apellidos.length + alias.length;
```

Los métodos principales son los siguientes:

Método	**Descripción**
indexOf(str)	Buscar primera ocurrencia de una subcadena dentro de una cadena
lastIndexOf(str)	Buscar última ocurrencia de una subcadena dentro de una cadena
search(str)	Similar a indexOf pero con expresiones regulares
substring(start, end)	Devuelve subcadena desde el índice que marca start a end - 1
slice(start, end)	Similar a substring pero acepta índices negativos, desde el final de la cadena
replace(str1, str2)	Reemplazar la primera cadena por la segunda una vez, o más si str1 es una er
toUpperCase()	Pasar a mayúsculas
toLowerCase()	Pasar a minúsculas
concat(str1, …, strN)	Concatenar con una o más cadenas
charAt(index)	Carácter en una posición determinada
split(char)	Devuelve un array con los tokens. Si char es "" devuelve un array de la cadena

Números

Todos los números en JavaScript son punto flotante de 64 bits, se pueden escribir con o sin decimales, en formato exponencial, o hexadecimal (0x):

```
var edad = 36;
var salario = 1750.85;
var retencion = 12e-2;   // 0.12
```

Existen números con valores especiales:

Valor	Constantes	Descripción
Infinity	Number.POSITIVE_INFINITY Number.NEGATIVE_INFINITY	Resultado de división por cero o de un número más allá de los límites (Number.MIN_VALUE y Number.MAX_VALUE)
NaN	Number.NaN	Resultado de operar con algo que no sea un número, o con NaN

Las cadenas de caracteres que contienen números son convertidas implícitamente a números en las operaciones:

```
var x = 36 / "3";   // 12
```

Los métodos principales son los siguientes:

Clase	Método	Descripción
Number	toString(base)	Pasar a cadena de caracteres en base 2, 8, 16 (o ninguna)
Number	toFixed(num)	Fijar a número de decimales
Number	toPrecision(num)	Fijar a número de cifras
	Number(x)	Convertir a número (cadenas, booleanos, fechas). Si no es posible la conversión devuelve NaN
	parseInt(x)	Convertir a entero. Si no es posible la conversión devuelve NaN, se permiten espacios aplicando conversión al primer token
	parseFloat(x)	Convertir a número. Si no es posible la conversión devuelve NaN, se permiten espacios aplicando conversión al primer token
Math	random()	Devuelve un número aleatorio entre [0, 1)
Math	min(x1, …, xN)	Mínimo de varios números
Math	max(x1, …, xN)	Máximo de varios números
Math	round(x)	Redondear al entero más cercano

Math	ceil(x)	Redondear al primer entero mayor
Math	floor()	Redondear al primer entero menor
Math	abs(x)	Valor absoluto
Math	sqrt(x)	Raíz cuadrada
Math	pow(x, y)	Valor de x elevado a y

Fechas

Las fechas se representan internamente como el número de milisegundos desde el 1 de Enero de 1970.

Los métodos principales son los siguientes:

Método	**Descripción**
Date()	Fecha actual, zona horaria del navegador
Date(ms)	Fecha en milisegundos desde el 01/01/1970
Date(str)	Fecha desde cadena en diversos formatos
Date(y, m, d, h, m, s, ms)	Fecha especificando todos los campos
toUTCString()	Muestra la fecha en formato UTC
toDateString()	Muestra la fecha formateada
getTime()	Milisegundos desde el 01/01/1970
getMilliseconds()	Milisegundos actuales
getSeconds()	Segundos actuales
getMinutes()	Minutos actuales
getHours()	Horas actuales (0-23)
getDate()	Día actual
getDay()	Día de la semana actual (0-6)
getMonth()	Mes actual (0-11)
getFullYear()	Año actual

Depuración

Una forma de realizar la depuración en JavaScript es lanzar mensajes a la consola, cada navegador dispone de un mecanismo para mostrar la consola:

```
console.log(x);
```

Los navegadores también disponen de depuración paso por paso, permitiendo establecer puntos de interrupción en el código de forma manual. También es posible forzar un punto de interrupción a través de código JavaScript escribiendo la palabra clave **debugger**.

Validación

La validación de formularios es una práctica muy habitual en JavaScript:

```
<form action="resultado.xhtml" onsubmit="return validateForm()"
method="post">
```

El método validateForm devolverá false en caso de error de validación, lo cual provoca que no se realice el submit:

```
function validateForm() {
    var x = document.forms[0]["nombre"].value;
    if (x == null || x == "") {
        alert("Nombre es obligatorio");
        return false;
    }
}
```

Document Object Model

Es la representación del árbol de elementos que componen un documento HTML. Todos los elementos son accesibles y modificables, y se pueden eliminar o agregar nuevos elementos sobre ellos. La estructura habitual es:

document

elemento raíz (html)

elemento head

elemento title

texto

elemento body

elemento n

atributo

texto

Los métodos principales de document son los siguientes:

Método	Descripción
getElementById(id)	Buscar elemento por id
getElementsByTagName(name)	Buscar elementos (array) por etiqueta
getElementsByClassName(name)	Buscar elementos (array) por clase
createElement(value)	Crear un elemento
createTextNode(value)	Crear nodo de texto
appendChild(element)	Añadir un elemento
removeChild(element)	Eliminar un elemento
replaceChild(element)	Reemplazar elemento
write(text)	Escribir valor al documento, con la página ya cargada la sobrescribe

Existen una serie de colecciones asociadas a document que permiten acceder a los elementos disponibles:

Propiedad	Descripción
head	Cabecera de la página
title	Título de la página
body	Cuerpo de la página
forms	Formularios
links	Enlaces
anchors	Referencias a otros elementos de la página
images	Imágenes

Las propiedades que se pueden acceder y/o modificar sobre los elementos son las siguientes:

Propiedad	Descripción
innerHTML	Cuerpo del elemento
atributo	Nombre de cualquiera de los atributos HTML disponibles
evento	Asignar una función al evento. Ver addEventListener y removeEventListener
style.propiedad	Nombre de cualquiera de las propiedades de un estilo
parentNode	Nodo padre, en cada nodo se puede acceder a nodeName, nodeValue y nodeType
childNodes	Nodos hijos
firstChild	Primer nodo hijo

lastChild	Último nodo hijo
previousSibling	Nodo hermano anterior
nextSibling	Nodo hermano siguiente

Browser Object Model

Se trata de una serie de elementos para interactuar con el navegador:

Elemento	Propiedad	Descripción
window.location	href	Obtener o modificar la página actual
window.location	hostname	Nombre del servidor
window.location	pathname	Ruta de la URL
window.location	protocol	Protocolo empleado (http)
window.history	back()	Anterior del navegador
window.history	forward()	Siguiente del navegador
window	open(url, tgt, car, rep)	Abrir url en ventana definida, devuelve objeto window
window	close()	Cerrar ventana
window	moveTo(x, y)	Mover a posición x,y
window	focus()	Poner el foco en la ventana
window	resizeTo(w, h)	Redimensionar a tamaño concreto
window	alert(msg)	Mensaje de alerta
window	confirm(msg)	Mensaje de confirmación, devuelve boolean
window	prompt(msg)	Pantalla de entrada, devuelve storing
navigator	cookieEnabled	Comprueba si las cookies están activadas
navigator	userAgent	Información del navegador
navigator	appVersion	Similar anterior
navigator	appName	Nombre del navegador
window	setTimeout(f, ms)	Ejecutar la función pasado el tiempo, devuelve un timer
window	clearTimeout(timer)	Detener el timeout del timer
window	setInterval(f, ms)	Ejecutar una función cada intervalo, devuelve un timer
window	clearInterval(timer)	Detener el intervalo del timer

También es posible acceder a los parámetros de pantalla:

Elemento	Propiedad	Descripción

window	innerHeight	Ancho del frame o pantalla del navegador en pixels
document.documentElement	clientHeight	Igual anterior, funciona en IE
document.body	clientHeight	Igual anterior, funciona en IE
window	innerWidth	Alto del frame o pantalla del navegador en pixels
document.documentElement	clientWidth	Igual anterior, funciona en IE
document.body	clientWidth	Igual anterior, funciona en IE
screen	height	Alto de pantalla
screen	width	Ancho de pantalla
screen	availHeight	Alto real disponible (eliminando barras, …)
screen	availWidth	Ancho real disponible (eliminando barras, …)
screen	colorDepth	Número de bits por color
screen	pixelDepth	Profundidad de píxel de pantalla

Referencias

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

http://www.w3schools.com/js/

http://www.w3schools.com/jsref/default.asp

http://www.w3schools.com/js/js_examples.asp

http://www.w3schools.com/quiztest/quiztest.asp?qtest=JavaScript

jQuery

Se trata de una API que permite la gestión de eventos, animaciones o interacciones AJAX de forma sencilla, simplificando además la programación JavaScript.

Para usar jQuery es necesario incluir la referencia, ya sea mediante descarga directa o incluyendo una referencia web desde un CDN:

```
<script src="/js/jquery-3.2.1.js"></script>

<!-- Versión de producción -->
<script src="http://ajax.googleapis.com/ajax/libs/jquery/3.2.1/
jquery.min.js"></script>
```

Sintaxis

La sintaxis de jQuery es sencilla, siempre se realiza una acción sobre un selector. Los selectores pueden hacer distintos tipos de referencias:

Selector	Ejemplo	Comentarios
this	$(this).hide()	Elemento actual
elemento	$('div').hide()	Elemento de tipo concreto
.class	$('.main').hide()	Elemento de una clase concreta
.id	$('#mail').hide()	Elemento con id concreto

Los selectores de tipo class siguen la notación css, podemos por tanto encontrar referencias como las siguientes:

```
$("[href]").hide();
$("tr:even").css("background-color", "#BDBDBD");
$("tr td:first-child").hide();
$("*").show();
```

Eventos

El concepto de evento es el mismo que ya vimos para JavaScript. La forma de definir un evento es a través de una función:

```
$(document).ready(function() {
            …
});

//Versión compacta de ready
$(function() {
            …
});
```

La mayoría de eventos del DOM poseen equivalencia en jQuery, a continuación se presentan algunos de éstos:

Evento	Comentarios
ready	Evento de carga del documento actual
click	Click simple
dblclick	Click doble
mouseenter/mouseleave	Entrada o salida con el ratón de un elemento
mousedown/mouseup	Eventos de click del ratón
keydown/keypress/keyup	Eventos de teclado
focus/blur	Obtener o perder el foco un elemento
submit	Enviar el formulario
hover	Al pasar por encima de un elemento
scroll	Al hacer scroll en un elemento o la ventana
…	

Todos estos eventos son en realidad una versión compacta de la asignación habitual de una función a un evento a través de la acción on:

```
$("div").on("click", function() {
            …
});

//Versión compacta de click
$("div").click(function() {
            …
});

//Varias asignaciones a través de on
$("div").on({
      click: function() {
            …
      },
      dblclick: function() {
            …
      },
            …
});
```

También es posible asignar una función de callback que se ejecuta una vez el evento ha concluido por completo:

```
$("#btnHide").on("click", function() {
      $("#div").hide(1000, function() {
            alert("div oculto");
      });
});
```

Animaciones

Existen distintas acciones para la implementación de animaciones simples:

Acción	Parámetros	Comentarios
hide/show/toggle	[speed], [callback] speed: "slow"/"fast"/ms	Ocultar/mostrar elementos speed: "slow"/"fast"/ms
fadeIn/fadeOut/ fadeToggle	[speed], [callback]	Animación de entrada o salida
fadeTo	speed, opacity, [callback]	Animación a nivel de opacidad
slideDown/slideUp/ slideToggle	[speed], [callback]	Animación deslizante
animate	{params}, [speed], [callback]	Animación personalizada params: propiedades a animar
stop	[stopAll], [goToEnd]	Detención de la animación stopAll: detener toda la cola goToEnd: terminar actual

Contenido y atributos

Existen llamadas jQuery para acceder a los contenidos y atributos HTML y CSS.

Método	Ejemplo	Comentarios
html	$('p').html('hola')	Obtener/modificar el contenido de un elemento
text	$('div').text()	Obtener/modificar el texto de un elemento (sin etiquetas HTML)
val	$('#name').val()	Obtener/modificar el valor de un componente de formulario
attr	$('#logo'). attr('src', 'logo.png')	Obtener/modificar el valor de un atributo
append	$('div').append('fin')	Añadir contenido al final
prepend	$('div').prepend('inicio')	Insertar contenido al inicio
before	$('div').before('antes')	Insertar contenido antes del elemento
after	$('div').after('después')	Añadir contenido después del elemento
empty	$('div').empty()	Limpiar los hijos de un elemento

remove	`$('#logo').remove()`	Eliminar un elemento. classes: parámetro opcional
replaceWith	`$('div').` `  replaceWith('<h1></h1>')`	Reemplazar por completo
css	`$('div').` `  css('color', 'red')`	Obtener/modificar el valor de propiedades css

Componentes

Existe una serie de **widgets** definidos por jQuery que se pueden emplear como componentes en la construcción de interfaces web ricos. Para poder emplear los componentes es necesario emplear el js de jQuery UI, desde CDN o personalizando el tema y descargando los archivos correspondientes.

```
<script src="https://code.jquery.com/ui/1.11.4/jquery-ui.min.js"></
script>
<link rel="stylesheet" href="https://code.jquery.com/ui/1.12.0/
themes/smoothness/jquery-ui.css" type="text/css" media="screen" />
```

Ejemplo de uso de calendario sobre un input o un div (inline):

```
$( function() {
      $( "#fecha" ).datepicker({
            dateFormat: "dd/mm/yy", //formato de salida
            onSelect: function(dateText) {
                  //ejecutado al seleccionar
            }
      });
});
```

Ejemplo de uso de un acordeón, que oculta o muestra paneles con contenido:

```
$( function() {
      $( "#accordion" ).accordion({
            heightStyle: "content" //ajustar al contenido
      });
});

<div id="accordion">
  <h3>Sección 1</h3>
  <div>
    <p>Cuerpo 1</p>
  </div>
  <h3>Sección 2</h3>
  <div>
    <p>Cuerpo 2</p>
  </div>
</div>
```

Mostrar un diálogo:

```
$( function() {
      $( "#dialog" ).dialog();
});

<div id="dialog" title="Título">
      Cuerpo
</div>
```

Autocompletar un input:

```
$( function() {
      var jefes = ["Alejandro", "Juan", "Fran"];
      $( "#jefe" ).autocomplete({
            source: jefes //origen de datos
      });
});
```

AJAX

La API de jQuery facilita las operaciones AJAX para la interacción con las tareas del servidor.

Existen distintos tipos de llamadas para esta funcionalidad, destacando las existentes para permitir llamadas GET y POST:

```
$.ajax(
       {
              url: "../ajax/method?userid=" + id,
              method: "GET",
              contentType : "application/json",
              timeout : 20000,
              success: function(result){
                     …
              },
              error: function(result){
                     …
              }
       }
);
```

En el caso de los POST será necesario codificar o serializar el valor de los campos del formulario para enviarlo al servidor. Admite tanto campos simples como de tipo binario (imágenes), siempre que el servidor sea capaz de resolver contenido de tipo **multipart/form-data**.

```
var form = $('form[name="user-form"]');
var formdata = false;
if (window.FormData){ //Objeto HTML5, si no existe serializa el form
      formdata = new FormData(form[0]);
}

$.ajax(
       {
              url: "../ajax/method",
              method: "POST",
              contentType: false,
              processData: false,
              timeout: 20000,
              data: formdata ? formdata : form.serialize(),
              success: function(result){
                     …
              },
              error: function(result){
                     …
              }
       }
);
```

Una alternativa posible es enviar un objeto JSON:

```
var producto = {id:5, nombre:"test"};

$.ajax(
       {
              url: "../ajax/method",
              method: "POST",
              contentType: "application/json",
              timeout: 20000,
              dataType: "json",
              data: JSON.stringify(producto),
              success: function(result){
                     …
              },
              error: function(result){
                     …
              }
       }
);
```

Referencias

https://www.w3schools.com/jquery/default.asp

https://www.w3schools.com/jquery/jquery_selectors.asp

http://www.w3schools.com/quiztest/quiztest.asp?qtest=jQuery

https://jquery.com

https://jqueryui.com

Bootstrap

Bootstrap es un framework que facilita el uso de HTML, CSS y JavaScript.

Incluye **templates** predefinidos que se ajustan a distintas necesidades de la web de manera **responsiva**, incorporando además la posibilidad de adquirir temas personalizados.

También se puede emplear a nivel básico para mejorar el formato de nuestro portal simplemente por el uso de sus clases de CSS.

```
<head>
 <meta name="viewport" content="width=device-width, initial-scale=1">
 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/
bootstrap/4.1.0/css/bootstrap.min.css">
 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/
jquery.min.js"></script>
 <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/
1.14.0/umd/popper.min.js"></script>
 <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.1.0/js/
bootstrap.min.js"></script>
```

Existen clases propias para los elementos HTML, permitiendo por ejemplo la definición de estilos de texto, alineaciones, colores, …

Cabe recordar que varias clases se pueden combinar en un mismo elemento.

```
<p class="font-weight-bold">Negrita</p>
<p class="font-weight-bold font-italic">Negrita y cursiva</p>
<p class="text-primary">Color principal</p>
```

Aunque Bootstrap incluye opciones de personalización más avanzadas, es posible personalizar el comportamiento de una determinada clase sobrescribiéndola desde una hoja de estilo propia:

```
.font-weight-bold{
  color: #2023fa;
}
```

En este caso es necesario tener en cuenta dos aspectos:

- El enlace a nuestra hoja personalizada debe aparecer bajo el de Bootstrap:

```
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/
bootstrap/4.1.0/css/bootstrap.min.css">
<link rel="stylesheet" href="css/custom.css">
```

- Muchas de las propiedades css de Bootstrap están marcadas como **!important**, con lo cual tomarán prioridad aunque sean redefinidas en nuestra hoja.

Tablas y Grids

También se encuentran disponibles clases para el formato automático de tablas:

```
<!-- Por defecto table width:100% -->
<table class="table table-bordered table-hover">
	<thead class="thead-dark">
		<tr>
			<th>Nombre</th><th>Edad</th>
		</tr>
	</thead>
	<tbody>
		<tr>
			<td>Juan</td><td>23</td>
		</tr>
	</tbody>
</table>
```

Para la distribución del contenido en la página, el sistema de grids de Bootstrap permite dar formato al documento empleando elementos de tipo div organizados en filas y columnas, de forma similar a como se realiza en las tablas.

Se puede considerar que una pantalla está conformada por 12 columnas, la distribución de la anchura de columnas se puede realizar de forma uniforme si no se especifica valor alguno, o indicando el ancho de columna específico.

```
<div class="row text-center">
	<div class="col">1</div>
	<div class="col">2</div>
	<div class="col">3</div>
</div>
<div class="row text-center">
	<div class="col-6">1</div>
</div>
<div class="row text-center">
	<div class="col-2">1</div>
	<div class="col-10">2</div>
</div>
```

Los grids se pueden comportar responsivamente empleando clases específicas que tienen en cuenta el tamaño de pantalla para ajustarlo en caso necesario horizontalmente:

Tipo	Tamaño	Descripción
.col-	XS	<576px
.col-sm-	S	>=576px
.col-md-	M	>=768px
.col-lg-	L	>=992px
.col-xl-	XL	>=1200px

Información destacada

Es posible emplear un contenedor para mostrar información de interés:

```
<div class="jumbotron">
      <h1>Jumbotron</h1>
      <p>Contenido destacado</p>
</div>
```

El empleo de bordes para mostrar párrafos es muy flexible, permitiendo a través de las clases mostrar colores y formas adicionales:

```
<span class="border rounded-circle border-warning"></span>
```

Los colores empleados en los bordes corresponden con estándares de Bootstrap que se utilizan para formato de texto, alertas, …

Color	Código
primary	#007bff
secondary	#6c757d
muted	#6c757d
dark	#343a40
body	#212529
light	#f8f9fa
white	#fff
success	#28a745
info	#17a2b8
warning	#ffc107
danger	#dc3545

Componentes HTML

A continuación se presentan algunas clases que se pueden emplear para elementos habituales de páginas HTML:

HTML	Clase	Ejemplo	Comentarios

`img`	`rounded`
	`rounded-circle`
	`img-thumbnail`
			Define el tipo de aspecto

`img`	`img-fluid`	`<img class="img-fluid" src="img/image1.jpg">`	Comportamiento responsivo

form	form-group	 <label for="nombre"> Nombre:</label> <input type="text" class="form-control" name="nombre"> 	Agrupa elementos del form. Un form por defecto toma width:100%
input	form-control form-control-sm form-control-lg		Necesario para width y padding apropiados Aplicable a text y a select Tamaños sm y lg
input	form-check form-check-label form-check-input	 <label class="form-check-label"> <input class="form-check-input" type="checkbox"> Recordar</label> 	Checkbox aspecto block Necesario form-check y form-check-label para padding apropiado Aplicable a check y radio
input	form-check-inline		Checkbox aspecto inline
input	form-control-range		Aplicable a range
input	form-control-file	<input type="file" class="form-control-file border">	Aplicable a file
ul	list-group		Listas
li	list-group-item	<li class="list-group-item list-group-item-info">Item	Elementos de listas Admite modificador de color
li	active		Elemento activo Aplicable a li, a
a	list-group-item-action	 Item 	Elementos con links Se aplica a enlaces dentro de un div
a	disabled		Elemento deshabilitado
button	btn btn-sm btn-lg	<button type="button" class="btn btn-primary btn-outline-secondary">Primary</button>	Formato de botón Aplicable a input, a, button Admite modificador de color, para relleno y borde Tamaños sm y lg
button	btn-block		Botón aspecto block
…			

Componentes propios

Existe una serie de componentes propios que se pueden emplear de forma sencilla para enriquecer el interfaz de la página a través de la propia hoja de estilos:

Componente	Clase	Ejemplo	Comentarios

Alert	alert	 <button type="button" class="close" data-dismiss="alert">×</button> Ole 	Mostrar alertas
Badge	badge badge-pill	<button type="button" class="btn btn-primary"> Alertas 3 </button>	Globo informativo Admite modificador de color
Progress	progress progress-bar	 	Barra de progreso El progreso se indica con width (sobre width 100%) Admite modificador de color
Card	card card-header card-body card-footer	 Cabecera Contenido Pie 	Cajas con contenido Por defecto width 100%
Collapse	collapse	<button type="button" class="btn btn-primary" data-toggle="collapse" data-target="#myCol"> Abrir </button> HOLA	Texto desplegable
Modal	modal modal-dialog modal-content modal-body	<button type="button" class="btn btn-primary" data-toggle="modal" data-target="#myModal"> Abrir</button> HOLA 	Ventana modal
…			

Referencias

https://www.w3schools.com/bootstrap/

https://www.w3schools.com/quiztest/quiztest.asp?qtest=Bootstrap

http://getbootstrap.com/docs/4.1/examples/

http://getbootstrap.com/docs/4.1/examples/starter-template/

AngularJS

Se trata de un framework para facilitar la construcción de aplicaciones web empleando como base fundamental JavaScript. AngularJS extiende los atributos HTML a través de expresiones que permiten modificar y mostrar la información de forma que se pueden desarrollar aplicaciones en una única página (SPAs).

Se puede utilizar AngularJS a través de un CDN:

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.10/
angular.min.js"></script>
</head>
<body>
      <div ng-app="">
        <input type="text" ng-model="name">
        <h1>Hola {{name}}</h1>
      </div>
</body>
</html>
```

Angular

Angular es el nombre que adquiere la plataforma para las versiones actuales de AngularJS, con la inclusión de herramientas y prácticas habituales en frameworks avanzados y una optimización bastante alta de rendimiento respecto a su predecesor.

Aunque permite variedad de lenguajes, se suele escribir en TypeScript, el cual se trata de un superconjunto de JavaScript que añade tipos adicionales del lenguaje, empleado en las nuevas tecnologías de scripting como Node.js, y enriquecido a través de la inclusión de módulos y clases JS.

Para las versiones actuales de Angular no existe un CDN como ocurre con la versión 1, con lo cual es necesario trabajar en conjunto con node y **npm.**

Directivas y expresiones

Se trata de extensión de atributos HTML para inicialización y asignación de información:

Directiva	Ejemplo	Comentarios
`ng-app`	`<div ng-app="">`	Inicializar app Solamente una por página
`ng-init`	`<div ng-app=""` `ng-init="user='Jane';age=24">`	Inicializar información de la app Solamente una por página
`ng-model`	`<input type="text"` `ng-model="user">`	Vincular información de la app a un componente
`ng-controller`	`<div` `ng-controller="controller">`	Inicializar controlador

ng-repeat	<li ng-repeat="p in items"> {{ p }}	Repetir elementos HTML
ng-disabled	<button ng-disabled="chk"> Send</button> <input type="checkbox" ng-model="chk"/>Disable	Componente deshabilitado o no
ng-show	<button ng-show="chk"> Send</button> <input type="checkbox" ng-model="chk"/>Show	Componente visible o no
ng-hide	<button ng-hide="chk"> Send</button> <input type="checkbox" ng-model="chk"/>Hide	Componente oculto o no
ng-click …	<th ng-click="ordBy('name')">	Directivas asociadas a eventos

Los valores de la información inicializada en la app normalmente se emplean desde expresiones de tipo Angular (no permiten todo el código JavaScript como bucles, condicionales o excepciones):

```
<p>Legal: {{ age > 21 ? 'yes' : 'no' }}</p>
<p>{{ name.length }}</p>
```

La directiva ng-model permite realizar un enlace en dos sentidos: desde un valor a un componente HTML, y desde el componente a una expresión:

```
<div ng-app="" ng-init="col='#FFA55A'">
<input type="text" style="background-color:{{col}}" ng-model="col">
</div>
```

Modelos

Los modelos se suelen emplear para la representación de información, desde variables a componentes de tipo input, y de ahí a expresiones:

```
<input type="text" ng-model="name">
<h1>Hola {{name}}</h1>
```

Los componentes del modelo permiten la obtención del estado de campos del formulario, así como la validación de los mismos:

```
<form name="userForm">
      <p>{{userForm.mail.$valid}} indica si es válido</p>
      <p>{{userForm.mail.$dirty}} si se ha modificado</p>
      <p>{{userForm.mail.$touched}} si ha tomado el foco</p>
      <p>{{userForm.mail.$error}} indica los errores</p>
      <input type="email" name="mail" ng-model="mail" required>
      <span ng-show="userForm.mail.$error.email">No válido</span>
</form>
```

Además es posible controlar la presentación de los campos a través de clases que se introducen al realizar las correspondientes validaciones (ng-valid, ng-dirty, ng-touched, ng-empty…):

```
input.ng-invalid {
    border: 2px solid red;
}
```

Existe la posibilidad de aplicar una serie de filtros a la información del modelo:

Filtro	Ejemplo	Comentarios
lowercase uppercase	{{ name | uppercase }}	Pasar a mayúscula o minúscula
currency	{{ cost | currency }}	Formato de moneda
date	{{ today | date: "dd/MM/y" }}	Formato de fecha
number	{{ age | number }}	Pasar número a cadena
json	{{ person | json }}	Mostrar como JSON
filter	ng-repeat="p in people | filter : 'j'"	Filtrado de array por un criterio
limitTo	ng-repeat="p in people | limitTo : 1"	Limitación de elementos de array
orderBy	ng-repeat="p in people | orderBy : p"	Ordenación por expresión

Controladores

Un controlador permite la definición de la información inicial de la app, así como de funciones necesarias para manipularla.

Se aconseja colocar las definiciones de script en la cabecera o al inicio del cuerpo del documento HTML, o preferentemente en un documento **js** independiente:

```
<script>
//Referencia a la app
var app = angular.module('app', []);
//Al iniciar la app, rootScope representa el ámbito de la app
app.run(function($rootScope) {
    $rootScope.title = 'HOLA';
});
//Definición de controlador: scope representa el ámbito del
controlador
app.controller('controller', function($scope) {
      //Valor
      $scope.title = 'Datos de la persona'
    //Objeto
      $scope.person = {name:'Juan',age:23};
    //Función
    $scope.greet = function(greet) {
            return greet + " " + $scope.person.name;
    }
});
</script>
```

Cada controlador trabaja en un ámbito concreto delimitado por un elemento con directiva ng-controller:

```
<body ng-app="app">
      <h1>{{ title }}</h1>

      <div ng-controller="controller">
            <h2>{{ title }}</h2>
            Name: <input type="text" ng-model="person.name"><br>
            Age: <input type="text" ng-model="person.age"><br>
            <p>Data: {{person.name + " " + person.age}}</p>
            <p>{{ greet('hello') }}</p>
      </div>
</body>
```

Servicios

Un servicio es una función asociada a la app.

```
app.service('parentCtrl', function() {
    this.legal = function (person) {
        return person.age > 21;
    }
});
app.controller('controller', function($scope, parentCtrl) {
    $scope.person = {name:'Juan',age:23};
    $scope.check = function(person) {
            $scope.legal = parentCtrl.legal(person);
    }
});
```

En AngularJS existen servicios que se pueden emplear para diversos tipos de acciones relacionadas con el documento (similar a JavaScript) como $location, $window, …

Uno de los servicios más empleados es HTTP, que produce una llamada AJAX, lo cual permitirá conexión con un web service para la obtención de información del modelo:

```
app.controller('controller', function($scope, $http) {
      $http({
               method : "GET",
               url : "../api/producto"
           }).then(function mySuccess(response) {
               $scope.data = response.data;
           }, function myError(response) {
               $scope.data = response.statusText;
           });
```

SPAs

Para conseguir implementar una app en una sola página, es necesario poder realizar cambios de vistas de forma automática. Una posibilidad es realizar inclusiones, pero la práctica habitual es el **routing**.

Esta técnica permite realizar cambios de vistas sin necesidad de realizar ninguna redirección, simplemente cargando cada nueva página en un contenedor.

```
<div ng-view></div>
<script>
var app = angular.module("app", ["ngRoute"]);
app.config(function($routeProvider) {
    $routeProvider
    .when("/", {
        templateUrl : "main.htm"
    })
    .when("/list", {
        templateUrl : "list.htm"
    })
    .when("/form", {
        templateUrl : "form.htm"
    });
});
</script>
```

Referencias

https://www.w3schools.com/angular/

https://www.w3schools.com/angular/angular_model.asp

https://www.w3schools.com/angular/angular_controllers.asp

https://www.w3schools.com/angular/angular_validation.asp

https://angularjs.org

https://angular.io