



DESARROLLO WEB EN ENTORNO CLIENTE

UD3 – OBJETOS PREDEFINIDOS DEL LENGUAJE

CICLO FORMATIVO DE GRADO SUPERIOR EN DESARROLLO DE
APLICACIONES WEB

I.E.S. HERMENEGILDO LANZ – 2022/2023

PROFESORA: VANESA ESPÍN

vespin@ieshlanz.es



Resultados de Aprendizaje y Criterios de Evaluación

RA3. Escribe código, identificando y aplicando las funcionalidades aportadas por los objetos predefinidos del lenguaje

| CRITERIOS de evaluación del RA3 | %UD | %Curso |
|--|-----|--------|
| a) Se han identificado los objetos predefinidos del lenguaje. | 10% | 10% |
| b) Se han analizado los objetos referentes a las ventanas del navegador y los documentos Web que contienen. | 15% | |
| c) Se han escrito sentencias que utilicen los objetos predefinidos del lenguaje para cambiar el aspecto del navegador y el documento que contiene. | 15% | |
| d) Se han generado textos y etiquetas como resultado de la ejecución de código en el navegador. | 10% | |
| e) Se han escrito sentencias que utilicen los objetos predefinidos del lenguaje para interactuar con el usuario. | 15% | |
| f) Se han utilizado las características propias del lenguaje en documentos compuestos por varias ventanas y marcos. | 10% | |
| g) Se han utilizado «cookies» para almacenar información y recuperar su contenido. | 15% | |
| h) Se ha depurado y documentado el código. | 10% | |

Objetivos Didácticos de la unidad 3

1. Utilizar los objetos predefinidos del lenguaje.
2. Distinguir entre el BOM y el DOM

IMPORTANTE: no todo lo que aprenderemos está en las diapositivas, HAY QUE SEGUIR LOS ENLACES DE REFERENCIA INDICADOS Y HACER LOS EJERCICIOS



Índice

PARTE 1

1. Introducción
2. Objetos predefinidos de más alto nivel en JavaScript
 - a. windows
 - b. document
 - c. Form
 - d. Expresiones regulares

PARTE 2

3. Objetos Nativos en JavaScript
 - a. String
 - b. Math
 - c. Number
 - d. Boolean
 - e. Date
4. Las cookies



Introducción a los objetos

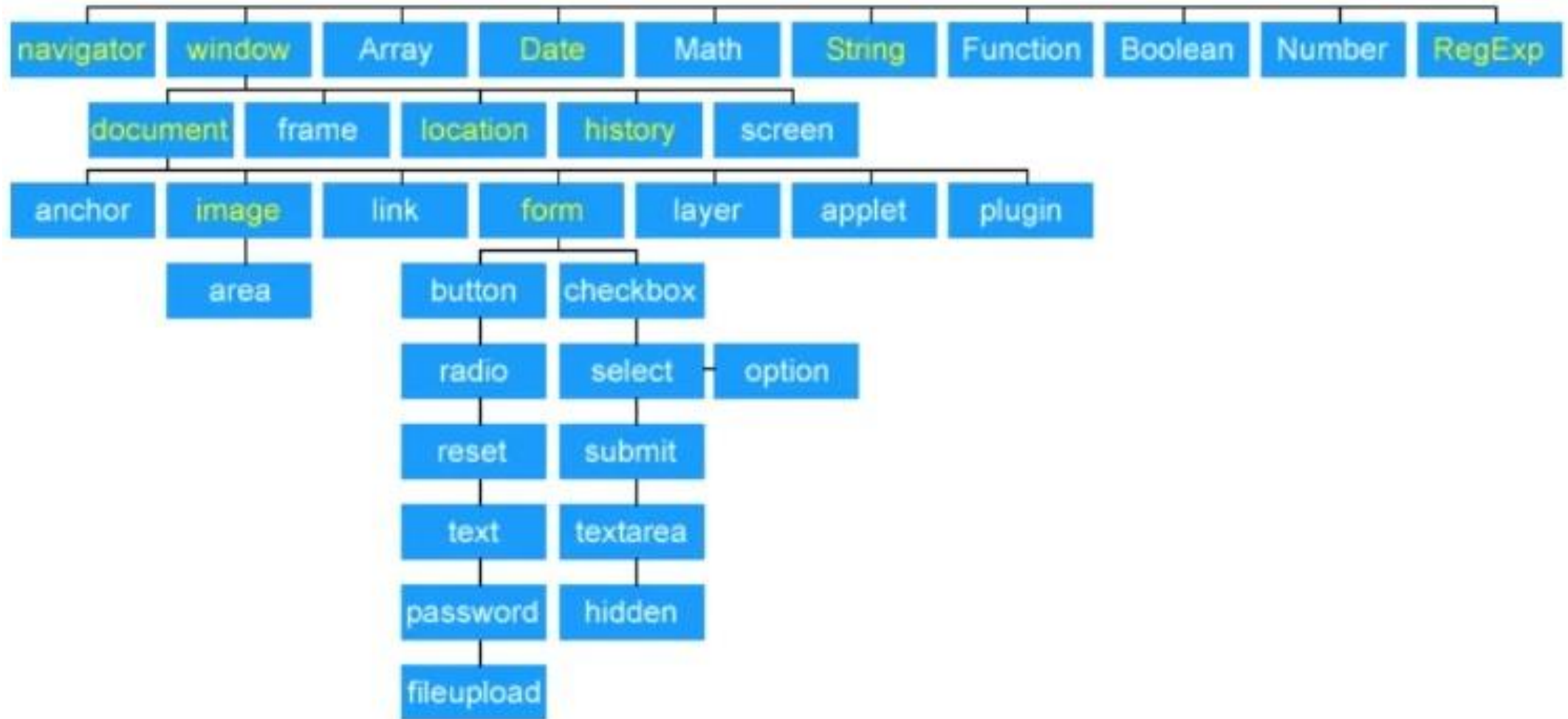
Un objeto es una colección de:

- Propiedades (valores)
- Métodos (funciones)
- Eventos (acciones)

Para acceder a una propiedad o método de un objeto se usa el punto (.)

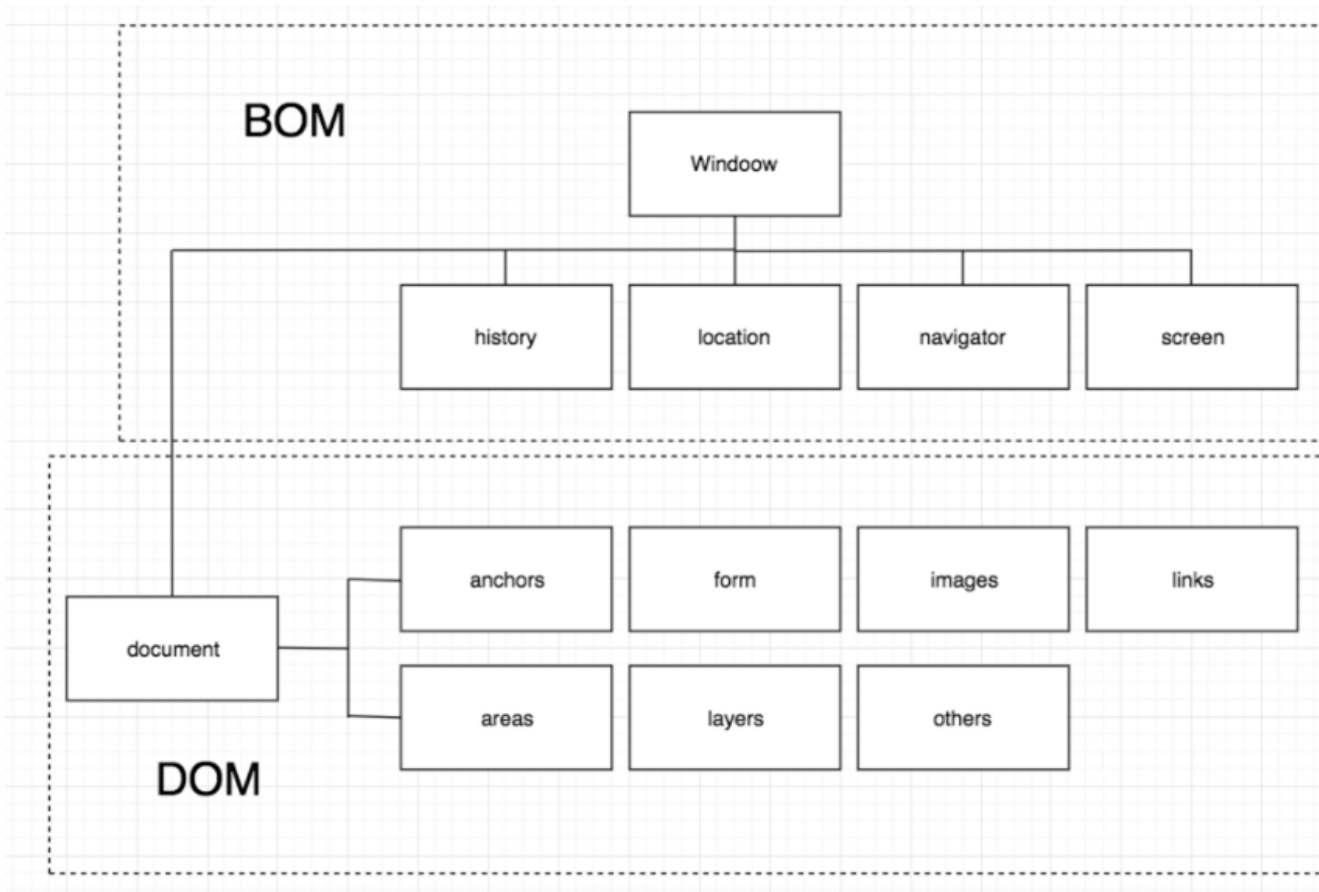
- `objeto.propiedad;`
- `objeto.metodo([argumentos]);`

Jerarquía de objetos predefinidos JavaScript



2. Objetos de más alto nivel en Javascript

Generalmente, la web está estructurada en dos partes. Lo referente al BOM (Browser Object Model) y lo referente al DOM (Document Object Model)

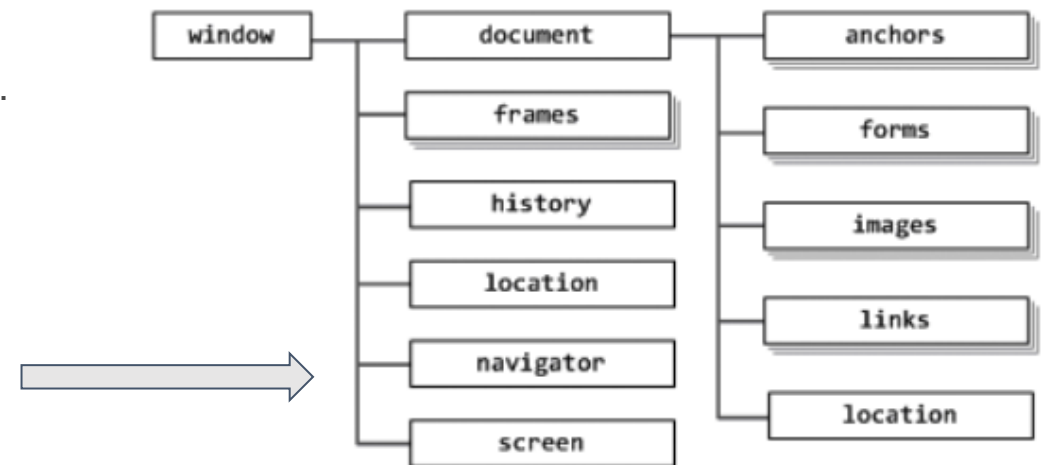


- El DOM se centraliza en el documento, el BOM incluye acceso a todas las áreas del navegador.
- El BOM es específico de cada navegador y DOM pretende ser totalmente independiente del navegador (estándar W3C).
- En BOM nos movemos en base al nombre de los objetos (*window*, *document*, *frames*, etc.) en cambio el esquema de navegación del modelo DOM se basa en el enfoque jerárquico.

https://www.w3schools.com/js/js_htmlDOM.asp

BOM

- Permite acceder y modificar las propiedades de las ventanas del propio navegador.
- Mediante BOM, es posible redimensionar y mover la ventana del navegador, modificar el texto que se muestra en la barra de estado y realizar muchas otras manipulaciones no relacionadas con el contenido de la página HTML.
- El mayor inconveniente de BOM es que, al contrario de lo que sucede con DOM, ninguna entidad se encarga de estandarizarlo o definir unos mínimos de interoperabilidad entre navegadores.
- Algunos de los elementos que forman el BOM son los siguientes:
 - Crear, mover, redimensionar y cerrar ventanas de navegador.
 - Obtener información sobre el propio navegador.
 - Propiedades de la página actual y de la pantalla del usuario.
 - Gestión de cookies.
 - Objetos ActiveX en Internet Explorer.
- El BOM está compuesto por varios objetos relacionados entre sí.
- El siguiente esquema muestra los objetos de BOM y su relación:

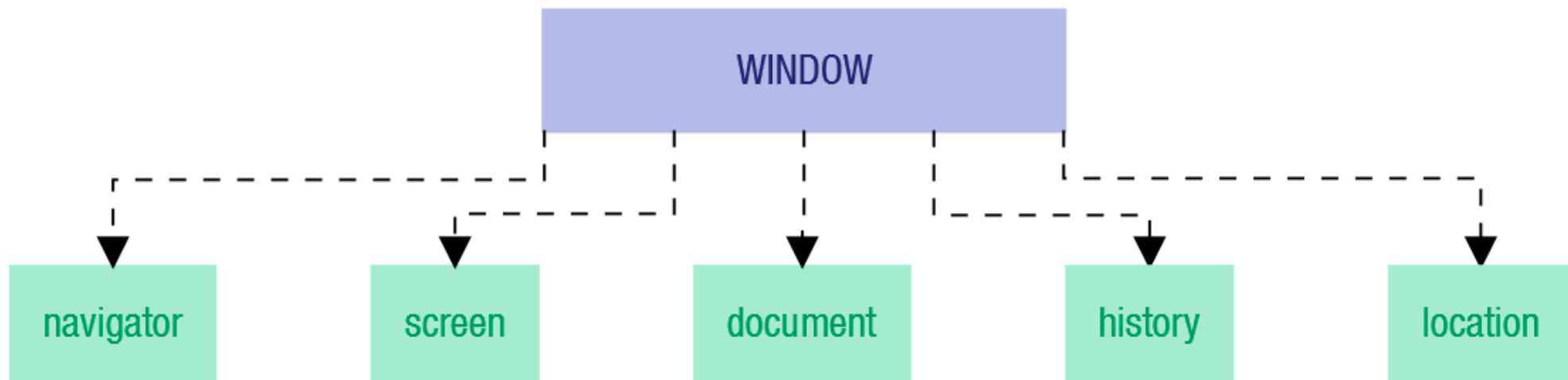


Objeto Window

- Es el **contenedor principal** de todo el contenido que se visualiza en el navegador. Tan pronto como se abre una ventana (`window`) en el navegador, incluso aunque no se cargue ningún documento en ella, este objeto `window` ya estará definido en memoria.
- Además de la sección de contenido del objeto `window`, que es justamente donde se cargarán los documentos, el campo de influencia de este objeto, abarca también las dimensiones de la ventana, así como todo lo que rodea al área de contenido: las barras de desplazamiento, barra de herramientas, barra de estado, etc.
- Como se ve en el gráfico de la **jerarquía de objetos** de la diapositiva siguiente, debajo del objeto `window` tenemos otros objetos como: `navigator`, `screen`, `history`, `location` y el objeto `document`. Este objeto `document` será el que contendrá toda la jerarquía de **objetos** que tengamos dentro de nuestra página HTML.

Gráfico del modelo de objetos de alto nivel para todos los navegadores que permitan usar JavaScript. Parte del BOM

JERARQUÍA DE OBJETOS



Acceso a propiedades y métodos

- La forma más lógica y común de realizar esa referencia, incluiría el objeto `window` tal y como se muestra en este ejemplo:

```
window.nombrePropiedad  
window.nombreMétodo( [parámetros] )
```

- Un objeto `window` también se podrá referenciar mediante `self`, cuando estamos haciendo la referencia desde el propio documento contenido en esa ventana:

```
self.nombrePropiedad  
self.nombreMétodo( [parámetros] )
```

Podremos usar cualquiera de las dos referencias anteriores, pero intentaremos dejar la palabra reservada `self`, para scripts más complejos en los que tengamos múltiples marcos y ventanas.

- Como el objeto `window` siempre estará presente al ejecutar nuestro script, podremos omitirlo, en referencias a los objetos dentro de esa ventana:

```
nombrePropiedad  
nombreMétodo( [parámetros] )
```

Gestión de Ventanas

El usuario crea la ventana principal abriendo URL en navegador. Después mediante scripts podemos crear/abrir sub-ventanas.

- Método `window.open()` :hasta tres parámetros que definen las características de la nueva ventana: la URL del documento a abrir, el nombre de esa ventana y su apariencia física (tamaño, color,etc.).
- Por ejemplo, si consideramos la siguiente instrucción que abre una nueva ventana de un tamaño determinado y con el contenido de un documento HTML:

```
var subVentana=window.open("nueva.html","nueva","height=800,width=600");
```

- si quisiéramos cerrar la nueva ventana desde nuestro script, simplemente tendríamos que hacer:

```
subVentana.close();
```

Objeto Window. Propiedades

El objeto `window` representa una ventana abierta en un navegador. Si un documento contiene marcos (`<frame>` o `<iframe>`), el navegador crea un objeto `window` para el documento HTML, y un objeto `window` adicional para para cada marco.

Propiedad

`closed`

`defaultStatus`

`document`

`frames`

`history`

`length`

`location`

`name`

`navigator`

`opener`

`parent`

`self`

`status`

Descripción

Devuelve un valor Boolean indicando cuando una ventana ha sido cerrada o no.

Ajusta o devuelve el valor por defecto de la barra de estado de una ventana.

Devuelve el objeto `document` para la ventana.

Devuelve un array de todos los marcos (incluidos iframes) de la ventana actual.

Devuelve el objeto `history` de la ventana.

Devuelve el número de frames (incluyendo iframes) que hay en dentro de una ventana.

Devuelve la Localización del objeto ventana (URL del fichero).

Ajusta o devuelve el nombre de una ventana.

Devuelve el objeto `navigator` de una ventana.

Devuelve la referencia a la ventana que abrió la ventana actual.

Devuelve la ventana padre de la ventana actual.

Devuelve la ventana actual.

Ajusta el texto de la barra de estado de una ventana.

Objeto Window. Métodos

Método

`alert()`

`blur()`

`clearInterval()`

`setInterval()`

`close()`

`confirm()`

`focus()`

`open()`

`prompt()`

Descripción

Muestra una ventana emergente de alerta y un botón de aceptar.

Elimina el foco de la ventana actual.

Resetea el cronómetro ajustado con `setInterval()`.

Llama a una función o evalúa una expresión en un intervalo (en milisegundos).

Cierra la ventana actual.

Muestra ventana emergente con un mensaje, botón de aceptar y botón de cancelar.

Coloca el foco en la ventana actual.

Abre una nueva ventana de navegación.

Muestra una ventana de diálogo para introducir datos.

Más info: https://www.w3schools.com/js/js_window.asp

Ejercicio 1

- a. Crea un documento html llamado *principal.html* con dos botones: uno para abrir una ventana y otro para cerrar esa misma ventana.
- b. Añade un botón que compruebe si la ventana se ha cerrado. Recuerda que también se puede cerrar con su icono de cerrado correspondiente.
- c. Crea una página html de nombre *auxiliar.html* que tenga un título h1 con tu nombre. Añade un botón a la página *principal.html* que abra dicha ventana y pida al usuario un nuevo nombre para ella.
- d. Añade un botón que abra, después de que el usuario confirme, en una ventana la página de Moodle Centros en tamaño 800x600 y que no permita cambiar su tamaño manualmente.
- e. Añadir a *principal.html* un enlace para cada una de las siguientes operaciones:
 - Cerrar la ventana abierta.
 - Pasar la ventana a segundo plano.
 - Mover la ventana a la posición 300,300 de la pantalla.
 - Cambiar el tamaño de la ventana a 200x200 pixeles.
 - Desplazar la barra de desplazamiento 10 pixeles hacia abajo.

DOM - El objeto Document

- Cada documento cargado en una ventana del navegador, será un objeto de tipo `document`.
- El objeto `document` proporciona a los scripts, el acceso a todos los elementos HTML dentro de una página.
- Este objeto forma parte además del objeto `window`, y puede ser accedido a través de la propiedad `window.document` o directamente `document` (ya que podemos omitir la referencia a la *ventana* actual).

Objeto Document. Propiedades y métodos

| Propiedad | Descripción |
|-----------|--|
| cookie | Devuelve todos los nombres/valores de las cookies en el documento. |
| domain | Cadena que contiene el nombre de dominio del servidor que cargó el documento. |
| referrer | Cadena que contiene la URL del documento desde el cuál llegamos al documento actual. |
| title | Devuelve o ajusta el título del documento. |
| URL | Devuelve la URL completa del documento. |

| Método | Descripción |
|---------------------|---|
| close() | Cierra el flujo abierto previamente con document.open(). |
| getElementById() | Para acceder a un elemento identificado por el id escrito entre paréntesis. |
| getElementsByName() | Para acceder a los elementos identificados por el atributo name escrito entre paréntesis. |
| getElementsByName() | Para acceder a los elementos identificados por el tag o la etiqueta escrita entre paréntesis. |
| open() | Abre el flujo de escritura para poder utilizar document.write() o document.writeln en el documento. |
| write() | Para poder escribir expresiones HTML o código de JavaScript dentro de un documento. |
| writeln() | Lo mismo que write() pero añade un salto de línea al final de cada instrucción. |

Objeto Document. Colecciones

| Colección | Descripción |
|-------------------------|--|
| anchors[] (Deprecated) | Es un array que contiene todos los hiperenlaces del documento. |
| forms[] | Es un array que contiene todos los formularios del documento. |
| images[] | Es un array que contiene todas las imágenes del documento. |
| links[] | Es un array que contiene todos los enlaces del documento. |

Enlaces:

- https://www.w3schools.com/js/js_htmlDOM_document.asp
- https://www.w3schools.com/jsref/dom_obj_document.asp

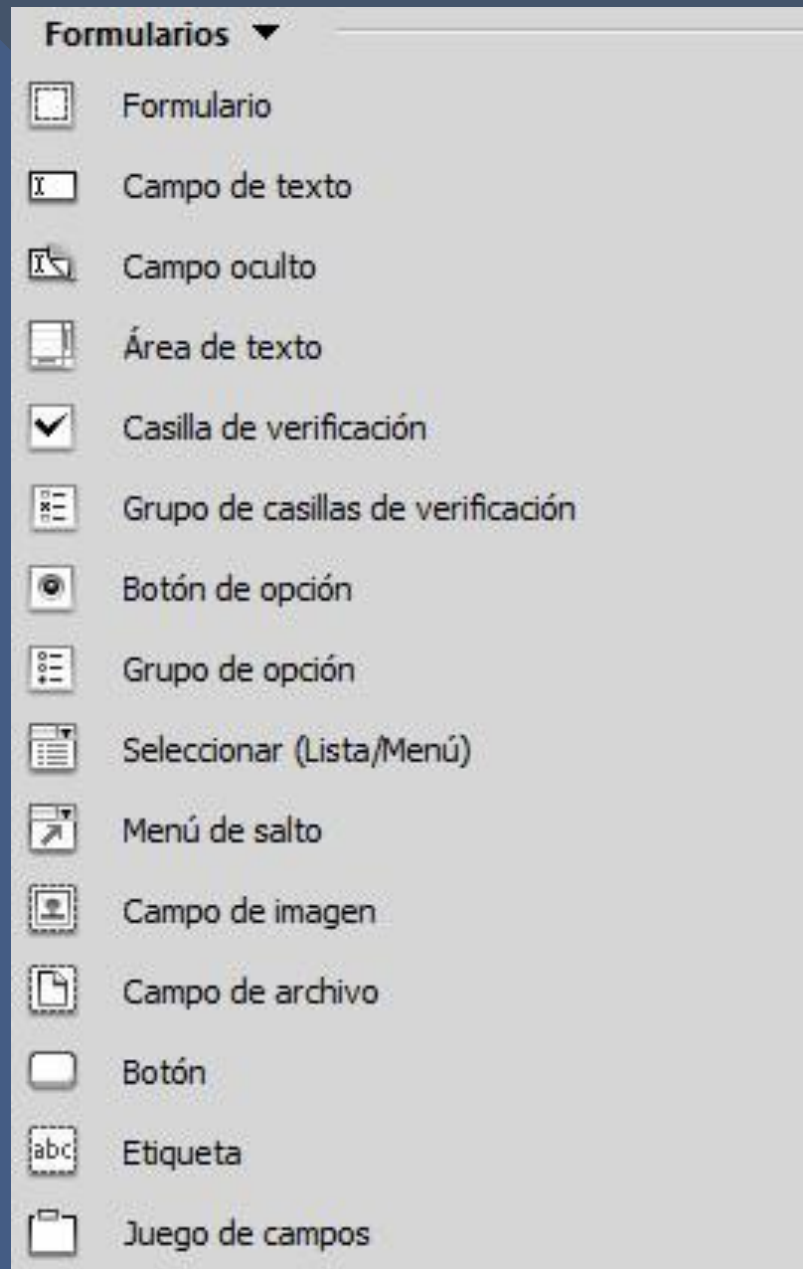
Ejercicio 2

Crea una página HTML con 5 imágenes y cinco párrafos llamada *imagenes.html* de título “Imagenes”. Ejecútala desde el live Server de Visual Code Studio. Mediante javascript y a través del objeto `document`:

- a. Modifica el título de la página a “*Página para trabajar el objeto document*”.
- b. Muestra la ruta de cada de cada imagen al final del mismo: “ruta de la imagen 1.. Ruta de la imagen 2...”. Usa un bucle.
- c. Cambia el texto del primer párrafo a “nuevo texto” mediante `innerHTML`
- d. Muestra mediante el método `writeln` la cadena que contiene el nombre de dominio del servidor que cargó el documento y mediante `write` la URL completa

El objeto form

- La mayor parte de interactividad entre una página web y el usuario tiene lugar a través de un formulario. Es ahí donde nos vamos a encontrar con los campos de texto, botones, checkboxes, listas, etc. en los que el usuario introducirá los datos, que luego se enviarán al servidor.
- En este apartado verás cómo identificar un formulario y sus objetos, cómo modificarlos, cómo examinar las entradas de usuario, enviar un formulario, validar datos, etc.
- Los formularios y sus controles, son **objetos del DOM que tienen propiedades únicas**, que otros objetos no poseen.
- JavaScript permite examinar y validar las entradas de usuario directamente, en el lado del cliente.
- El objeto de JavaScript *form*, es una propiedad del objeto *document*. Se corresponderá con la etiqueta *<form>* del HTML. Un formulario podrá ser enviado llamando al método *submit* de JavaScript, o bien haciendo click en el botón *submit* del formulario.



Ejemplo de lista de
objetos al trabajar
con formularios

Formas de selección del objeto form

```
<div id="menulateral">
  <form id="contactar" name="contactar" action="...">...</form>
</div>
```

Método 1: usando el método *getElementById()* del DOM. Ojo: *id* únicos en nuestros objetos.

Ejemplo: `var formulario=document.getElementById("contactar");`

Método 2: con el método *getElementsByTagName()* del DOM, para acceder a un objeto a través de la etiqueta HTML que queramos.

Ejemplo: `var formularios = document.getElementsByTagName("form");`
`var primerFormulario = formularios[0]; // primer formulario del documento`

o lo que es lo mismo:

`var primerFormulario = document.getElementsByTagName("form")[0];`

Formas de selección del objeto form

Otra posibilidad interesante que te permite el método 2 anterior, es la de buscar objetos con un padre determinado, por ejemplo

```
var menu=document.getElementById("menulateral");  
var formularios=menu.getElementsByTagName("form"); // formularios del menu lateral  
var primerFormulario= formularios[0];           // primer formulario del menú lateral
```

Método 3: con la colección forms[] del objeto document. Array que contiene la referencia a todos los formularios que tenemos en nuestro documento.

```
var formularios = document.forms; // la referencia a todos los formularios del documento  
var miformulario = formularios[0]; // primer formulario del documento
```

O bien

```
var miformulario = document.forms[0]; // primer formulario del documento
```

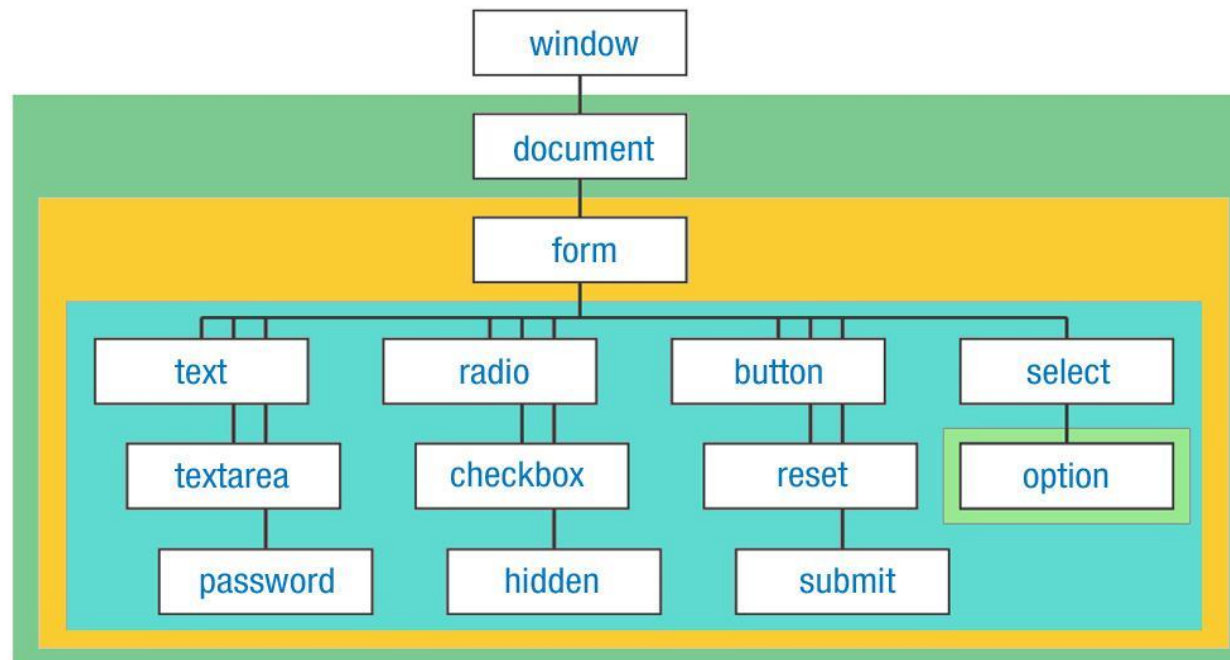
O bien

```
var miformulario = formularios["contactar"]; // referenciamos al formulario con name "contactar"
```


El formulario como objeto y contenedor

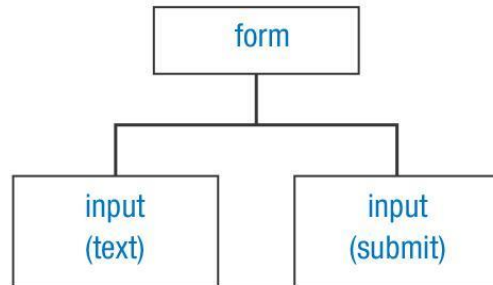
- El objeto Form está dentro de dos árboles al mismo tiempo. En las nuevas definiciones del DOM Form es el padre de todos sus nodos hijos, incluidos objetos y textos, anteriormente Form sólo era padre de sus objetos (input, select, button y elementos textarea).

Jerarquía de nivel 0 del DOM para formularios y controles:

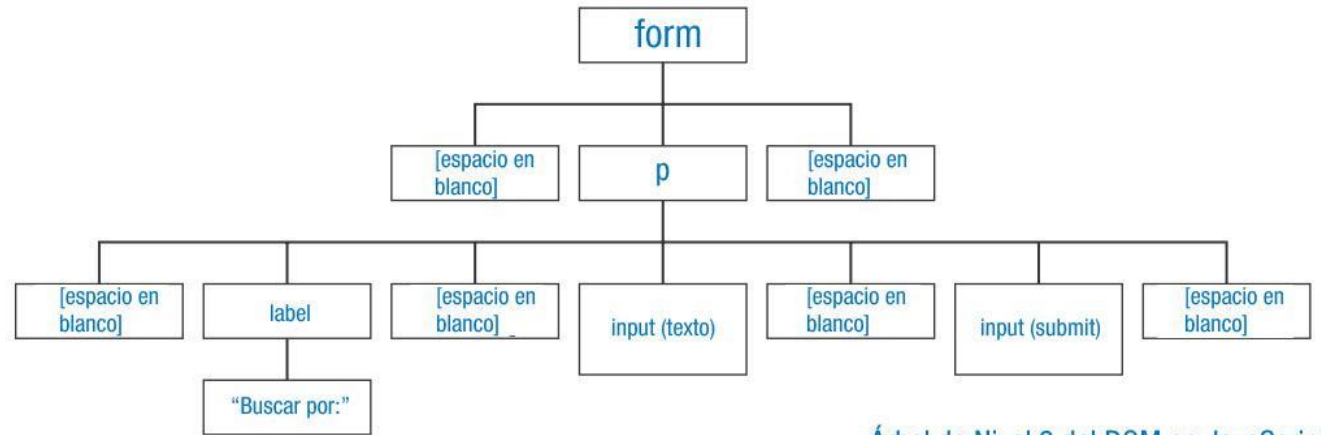


El formulario como objeto y contenedor

```
<form action="buscar.php" name="elFormulario" id="miFormulario" method="post">
  <p>
    <label for="busqueda">Buscar por:</label>
    <input id="busqueda" name="busqueda" type="text" value="">
    <input id="submit" type="submit" value="Buscar">
  </p>
</form>
```



Árbol de nivel 0 del DOM



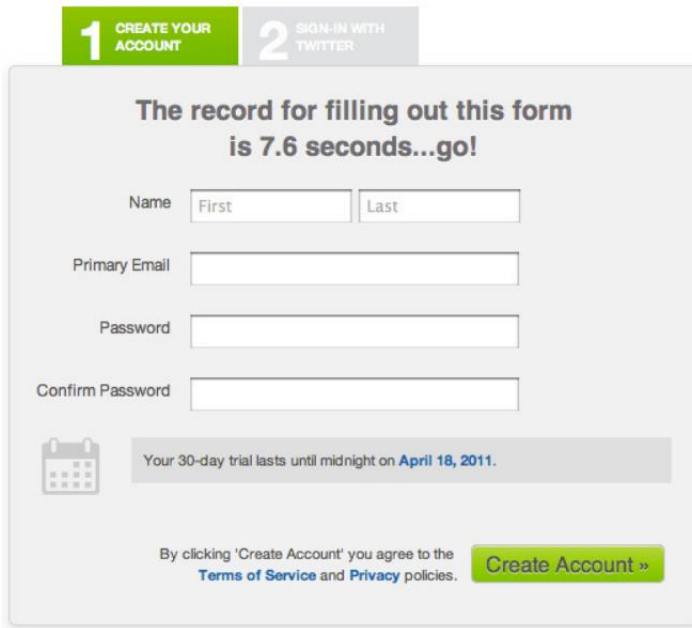
Árbol de Nivel 2 del DOM en JavaScript

El árbol del DOM de nivel 2, se puede utilizar para leer y escribir en todo el documento con un nivel muy fino de **granularidad**. El árbol del DOM de nivel 0, hace muchísimo más fácil leer y escribir los controles del formulario. Aunque utilices las técnicas del DOM 0 o DOM 2, los objetos siguen siendo los mismos. Por ejemplo

```
var elFormulario = document.getElementById("miFormulario");
var control = elFormulario.busqueda;
```

Acceso a propiedades y métodos del objeto form

Los formularios pueden ser creados a través de las etiquetas HTML, o utilizando JavaScript y métodos del DOM. Se pueden asignar atributos como name, action, target y enctype. Cada uno de estos atributos es una propiedad del objeto Form, a las que podemos acceder utilizando su nombre en minúsculas



The screenshot shows a web form with two tabs at the top: '1 CREATE YOUR ACCOUNT' (active) and '2 SIGN-IN WITH TWITTER'. The form title is 'The record for filling out this form is 7.6 seconds...go!'. It contains four input fields: 'Name' (split into 'First' and 'Last'), 'Primary Email', 'Password', and 'Confirm Password'. Below the fields is a calendar icon and a message: 'Your 30-day trial lasts until midnight on April 18, 2011.' At the bottom, there is a link to 'Terms of Service and Privacy policies' and a green 'Create Account »' button.

```
var paginaDestino = objetoFormulario.action;
```

```
objetoFormulario.action = "http://www.educacion.gob.es/recepcion.php";
```

Lo mismo usando referencias a objetos:

```
var paginaDestino = document.getElementById("id").action;  
document.forms[0].action = "http://www.educacion.gob.es/recepcion.php";
```

Propiedades y métodos del objeto Form

| Propiedad | Descripción | W3C |
|---------------|--|-----|
| acceptCharset | Ajusta o devuelve el valor del atributo accept-charset en un formulario. | Sí |
| action | Ajusta o devuelve el valor del atributo action en un formulario. | Sí |
| enctype | Ajusta o devuelve el valor del atributo enctype en un formulario. | Sí |
| length | Devuelve el número de elementos en un formulario. | Sí |
| method | Ajusta o devuelve el valor del atributo method en un formulario. | Sí |
| name | Ajusta o devuelve el valor del atributo name en un formulario. | Sí |
| target | Ajusta o devuelve el valor del atributo target en un formulario. | Sí |

| Método | Descripción | W3C |
|----------|------------------------|-----|
| reset() | Resetea un formulario. | Sí |
| submit() | Envía un formulario. | Sí |

La propiedad form.elements

- La propiedad `elements[]` de formulario es una colección, que contiene todos los objetos *input* dentro de un formulario. Es otro array, con los campos *input* en el orden del documento.
- Generalmente, es mucho más eficaz y rápido referenciar a un elemento individual usando su ID, pero a veces, los scripts necesitan recorrer cada elemento del formulario, para comprobar que se han introducido sus valores correctamente.
- Por ejemplo, empleando la propiedad `elements[]`, podemos hacer un bucle que recorra un formulario y si los campos son de tipo texto, pues que los ponga en blanco:

```
var miFormulario = document.getElementById("contactar");
// guardamos la referencia del formulario en una variable.

if (!miFormulario) return false; // Si no existe ese formulario devuelve false.

for (var i=0; i< miFormulario.elements.length; i++)
{
    if (miFormulario.elements[i].type == "text")
    {
        miFormulario.elements[i].value = "";
    }
}
```


Generar
Formulario

Contacto

Nombre

Email

Mensaje

Enviar

Ejercicio 3

En este ejercicio vamos a generar el formulario de la figura en JavaScript. Requisitos de nuestra página formulario.html:

- creamos un elemento `<section>` con `id="ContentFormulario"`
- Dentro creamos un botón "Generar Formulario" al pulsarlo, se generará el formulario, también dentro de la sección.
- La función JavaScript se llamará `GeneraFormulario()`

Usaremos la funciones: **`createElement`**, **`setAttribute`**, **`appendChild`** para trabajar con los elementos del árbol DOM

Podéis usar este enlace de ayuda:

<https://es.acervolima.com/como-crear-un-formulario-dinamicamente-con-javascript/>

Objetos relacionados con formularios

Referenciamos a los diferentes elementos a través de su id o su nombre.

```
document.getElementById("id-del-control")
```

o

```
document.nombreFormulario.name-del-control
```

Ejemplo. Escribe 5 formas de referenciar al elemento de texto del formulario siguiente:

```
<form id="formularioBusqueda" action="cgi-bin/buscar.pl">
  <p>
    <input type="text" id="entrada" name="cEntrada">
    <input type="submit" id="enviar" name="enviar" value="Buscar...">
  </p>
</form>
```

 solución

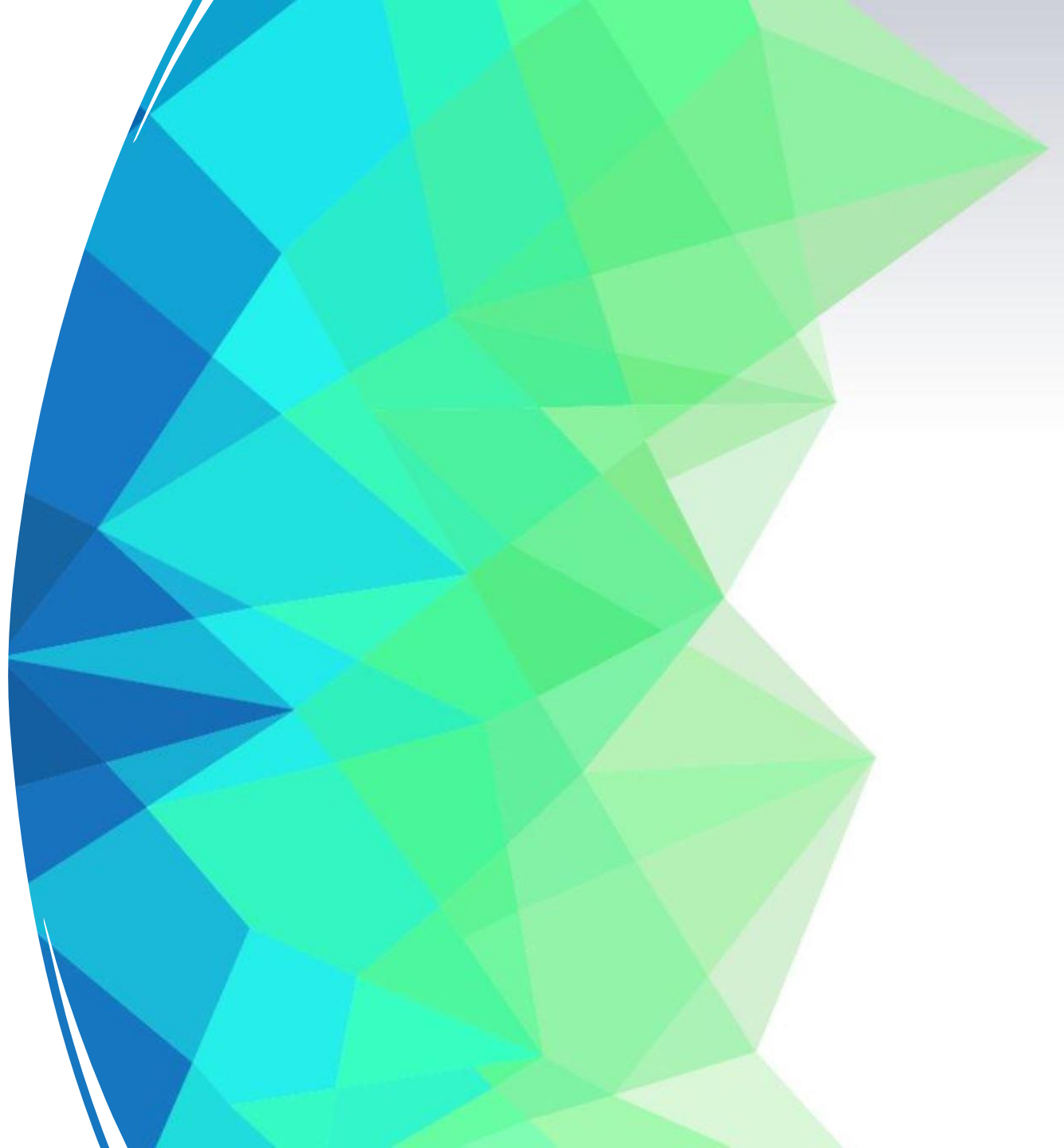
```
document.getElementById("entrada")
document.formularioBusqueda.cEntrada
document.formularioBusqueda.elements[0]
document.forms["formularioBusqueda"].elements["cEntrada"]
document.forms["formularioBusqueda"].cEntrada
```

OJO: A la hora de identificar los objetos en un formulario lo más recomendable es que el atributo id y el atributo name sean iguales y que no se repitan los id en todo el documento

Objetos relacionados con formularios

Vamos a ver:

- Objeto input de tipo texto
- Objeto input de tipo radio
- Objeto select
- Pasar objetos a las funciones usando *this*



Objeto input de tipo texto

- Cada uno de los 4 elementos de tipo texto de los formularios: `text`, `password`, `hidden` y elementos `textarea`. Todos los elementos, excepto `hidden`, se mostrarán en la página para introducir texto o seleccionar opciones.
- Al servidor se envían: `name` + `value` de cada elemento. El contenido de un `value` es siempre una cadena de texto (quizás necesitemos conversiones numéricas).

| Propiedad | Descripción |
|---------------------------|---|
| <code>defaultValue</code> | Ajusta o devuelve el valor por defecto de un campo de texto. |
| <code>form</code> | Devuelve la referencia al formulario que contiene ese campo de texto. |
| <code>maxLength</code> | Devuelve o ajusta la longitud máxima de caracteres permitidos en el campo de tipo texto |
| <code>name</code> | Ajusta o devuelve el valor del atributo <code>name</code> de un campo de texto. |
| <code>readOnly</code> | Ajusta o devuelve si un campo es de sólo lectura, o no. |
| <code>size</code> | Ajusta o devuelve el ancho de un campo de texto (en caracteres). |
| <code>type</code> | Devuelve el tipo de un campo de texto. |
| <code>value</code> | Ajusta o devuelve el contenido del atributo <code>value</code> de un campo de texto. |

| Metodo | Descripción |
|-----------------------|---|
| <code>select()</code> | Selecciona el contenido de un campo de texto. |

```

<!DOCTYPE html>
<html>
<head>
  <meta http-equiv="content-type" content="text/html; charset=utf-8">
  <title>Propiedad VALUE de un objeto de tipo texto</title>
  <script type="text/javascript">
    function convertirMayusculas() {
      /* En este ejemplo accedemos a la propiedad value de un objeto con id nombre y le asignamos su
      contenido actual pero convertido a mayúsculas con el método toUpperCase() del objeto String. */
      document.getElementById("nombre").value=document.getElementById("nombre").value.toUpperCase(); }
  </script>
</head>
<body>
  <h1>Propiedad VALUE de un objeto INPUT de tipo TEXT</h1>
  <form id="formulario" action="pagina.php">
    <p>
      <label for="nombre">Nombre y Apellidos: </label>
      <input type="text" id="nombre" name="nombre" value="" size="30" onblur="convertirMayusculas()">
    </p>
    <p> Introduce tu Nombre y Apellidos y haz click fuera del campo. </p>
  </form>
</body>
</html>

```

EJERCICIO 4

Formulario que al pinchar fuera de él modifique el campo de texto pasándolo a mayúsculas

Enlaces útiles

- Propiedades genéricas de los objetos del DOM
https://www.w3schools.com/jsref/dom_obj_all.asp
- Más información sobre el objeto input
<https://www.htmlquick.com/es/reference/tags/input.html>

Ejercicio 5

Completa la función *myFunction* para que al enviar el formulario escriba en el párrafo *demo*:

- El número de elementos del formulario
- Todos los nombres y valores de los campos de texto
- El nombre del formulario
- El target del formulario

```
<!DOCTYPE html>
<html>
<body>

<form id="frm1" action="/action_page.php">
  First name: <input type="text" name="fname" value="Donald"><br>
  Last name: <input type="text" name="lname" value="Duck"><br><br>
  <input type="submit" value="Submit">
</form>

<p>Click "Try it" to display the value of each element in the form.</p>

<button onclick="myFunction()">Try it</button>

<p id="demo"></p>

<script>
function myFunction() {

}
</script>

</body>
</html>
```

Objeto input de tipo checkbox

En los botones de un formulario la propiedad `value` nos mostrará el texto del botón, pero en un checkbox la propiedad `value` es un texto asociado al objeto. Este texto no se mostrará en la página, y sirve para asociar un valor con la opción actualmente seleccionada. Dicho valor será el que se enviará, cuando enviemos el formulario.

```
<label for="cantidad">Si desea recibir 20 Kg marque esta opción:  
    <input type="checkbox" id="cantidad" name="cantidad" value="20 Kg">  
</label>
```

Para saber si un campo de tipo checkbox está marcado, usamos la propiedad ***checked***. Esta propiedad contiene un valor booleano: *true* si el campo está marcado o *false* si no lo está.

- Si checkbox marcado: el navegador enviará el par name/value "cantidad" y "20 Kg".
- Si el checkbox no marcado: este campo no será enviado en el formulario.

Etiquetas `label` no se envían al servidor.

Ejercicio 6. Rellenar las funciones marcar y desmarcar

```
<html>
<head>
<script type="text/javascript">
function marcar()
{

}
function desmarcar()
{

}
</script>
</head>
<body>
    <form action="" method="get">
        <input type="checkbox" id="verano" name="verano" value="Si"/>¿Te gusta el verano?
        <input type="submit" />
    </form>
    <button onclick="marcar()">Marcar Checkbox</button>
    <button onclick="desmarcar()">Desmarcar Checkbox</button>
</body>
</html>
```

Propiedades y métodos del objeto input para checkbox

| Propiedad | Descripción |
|----------------|---|
| checked | Ajusta o devuelve el estado checked de un checkbox. |
| defaultChecked | Devuelve el valor por defecto del atributo checked. |
| form | Devuelve la referencia al formulario que contiene ese campo checkbox. |
| name | Ajusta o devuelve el valor del atributo name de un checkbox. |
| type | Nos indica que tipo de elemento de formulario es un checkbox. |
| value | Ajusta o devuelve el valor del atributo value de un checkbox. |

| Método | Descripción |
|----------|--|
| onChange | Evento que se produce al marcar/desmarcar un checkbox o un radio button. |

Ejercicio 7

- a. La página siguiente muestra tres checkbox. Si seleccionas uno o más y haces clic en el botón, se deberán mostrar los valores del checkbox seleccionado

Para empezar podrás guiarte por:

https://www.w3schools.com/howto/howto_js_display_checkbox_text.asp

- b. Añade la siguiente funcionalidad: cada vez que pulsemos el checkbox Red nos indicará con alert si está activado o no.

Select your favorite colors:

☐ Red ☐ Green ☐ Blue

Get Selected Colors

Objeto input de tipo radio

- Recordar: deberemos asignar el mismo atributo *name* a cada uno de los botones del grupo.
- Así se crea un array con la lista de esos objetos que tienen el mismo name. El contenido del atributo name será el nombre del array.
- Algunas propiedades, se las podremos aplicar al grupo como un todo; otras en cambio, tendremos que aplicárselas a cada elemento del grupo y lo haremos a través del índice del array del grupo.
- Por ejemplo, podemos ver cuantos botones hay en un grupo radio, consultando la propiedad `length` de ese grupo:

```
objetoFormulario.nombregrupo.length
```

- Y si queremos acceder a la propiedad `checked` de un botón en particular, lo haremos accediendo a la posición del array y a la propiedad `checked`:

```
// Accedemos a la propiedad checked del primer botón del grupo:  
objetoFormulario.nombregrupo[0].checked
```

Propiedades y métodos del objeto input radio

| Propiedad | Descripción |
|-----------------|--|
| checked | Marca o comprueba si está marcado un <i>checkbox</i> o un <i>radio</i> . Es una propiedad <i>booleana</i> (que solo admite verdadero o falso) en la cual su valor será <i>"true"</i> (verdadero) si el objeto está marcado y <i>"false"</i> (falso) en caso que no lo esté. |
| name | Es el nombre que identifica a un checkbox a un grupo de radio buttons |
| disabled | Bloquea el <i>checkbox</i> o <i>radio</i> . Por lo tanto "No es enviado en el formulario" y "actúa como si no existiese". |
| length | Es la cantidad de <i>radio buttons</i> que existe en un grupo determinado con el mismo <i>name</i> . |
| index | <i>Array</i> que contiene todos los <i>radio buttons</i> que hay en un grupo con el mismo nombre. Para un grupo de 5 <i>radio buttons</i> con mismo nombre y diferente valor cada uno, para referirnos al cuarto de ellos se debe usar la sintaxis: formulario.nombre_radio[3] |
| value | Es un valor asociado a cada checkbox o radio. En el caso de los checkbox es útil asignar a un grupo de éstos con el mismo name distintos valores. Pero si se quiere acceder a las propiedades de cada checkbox separadamente mediante JavaScript, es mejor utilizar un <i>name</i> diferente para cada checkbox, ya que de lo contrario (si todos tienen el mismo <i>name</i>) la propiedad value resulta poco útil. En el caso de los radio buttons es una propiedad necesaria siempre al tener todos el mismo <i>name</i> , aunque para su acceso mediante JavaScript, esta propiedad no es muy útil. |

| Evento | Descripción |
|-----------------|---|
| onFocus | Permite realizar una acción al poner el foco en el objeto. |
| onBlur | Permite realizar una acción cuando el foco ya no se encuentra en el objeto. |
| onClick | Permite realizar una acción cuando se hace click sobre el objeto. |
| onChange | Evento que se produce al marcar/desmarcar un <i>checkbox</i> o un <i>radio button</i> . |

Ejercicio 8

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>Trabajando con objetos input de tipo radio</title>
<script type="text/javascript">
function mostrarDatos()
{
}
</script>
</head>
<body>
<h1>Trabajando con objetos input de tipo radio</h1>
<form name="formulario" action="stooges.php">
  <fieldset><legend>Selecciona tu actor favorito:</legend>
  <label for="actor-1">Willis</label>
  <input type="radio" name="actores" id="actor-1" value="Walter Bruce Willis - 19 de Marzo de 1955" checked>
  <label for="actor-2">Carrey</label>
  <input type="radio" name="actores" id="actor-2" value="James Eugene Jim Carrey - 17 de Enero de 1962">
  <label for="actor-3">Tosar</label>
  <input type="radio" name="actores" id="actor-3" value="Luis Tosar - 13 de Octubre de 1971">
  <input type="button" id="consultar" name="consultar" value="Consultar Más Datos" onclick="mostrarDatos()">
  </fieldset></form>
</body>
</html>
```

Rellena la función mostrarDatos para que al enviar el formulario vaya mostrando un alert con el nombre y fecha de nacimiento de cada actor seleccionado

Objeto Select

Un objeto **select** está compuesto de un array de objetos **option**. El objeto select se suele mostrar como una lista desplegable en la que seleccionas una de las opciones, aunque también tienes la opción de selecciones múltiples, según definas el objeto en tu documento. Vamos a ver cómo gestionar una lista que permita solamente selecciones sencillas.

La propiedad más importante es *selectedIndex*, a la que puedes acceder de las siguientes formas:

```
//forma 1
objetoFormulario.nombreCampoSelect.selectedIndex
//forma 2
document.getElementById("objetoSelect").selectedIndex
```

El valor devuelto por esta propiedad, es el **índice** de la opción actualmente seleccionada (recuerda que comienzan en la posición 0).

Las opciones tienen dos propiedades accesibles que son **text** y **value**, y que te permitirán acceder al texto visible en la selección y a su valor interno para esa opción (ejemplo: `<option value="OU">Ourense</option>`). Veamos las formas de acceso a esas propiedades:

```
//forma 1
objetoFormulario.nombreCampoSelect.options[n].text
objetoFormulario.nombreCampoSelect.options[n].value

//forma 2
document.getElementById("objetoSelect").options[n].text
document.getElementById("objetoSelect").options[n].value
```

Mas info:

https://www.w3schools.com/jsref/dom_obj_select.asp

ACTIVIDAD

Reescribe la función consultar con la otra forma aprendida

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8">
<title>DWE05 - Trabajando con un objeto Select</title>
<script type="text/javascript">
function consultar()
{
    var objProvincias=document.getElementById("provincias");
    var texto=objProvincias.options[objProvincias.selectedIndex].text;
    var valor=objProvincias.options[objProvincias.selectedIndex].value;
    alert("Datos de la opción seleccionada:\n\nTexto: "+texto+"\nValor: "+valor);
}
</script>
</head>
<body>
<h1>Trabajando con un objeto Select</h1>
<form id="formulario" action="pagina.php">
<p>
    <label for="provincias">Seleccione provincia: </label>
    <select name="provincias" id="provincias">
        <option value="C">La Coruña</option>
        <option value="LU">Lugo</option>
        <option value="OU">Ourense</option>
        <option value="PO">Pontevedra</option>
    </select>
</p>
<p>Selecciona una opción y pulsa el botón.</p>
<input type="button" name="boton" value="Consultar información de la opción" onclick="consultar()"/>
</form>
</body>
</html>
```


Ejercicio 9

- Crea un programa donde se pregunte al usuario un número (con prompt) y deba seleccionar otro número (del 1 al 5) de una lista desplegable para multiplicarlos y dar la solución en un párrafo con identificador “solucion”.

Pasando objetos a las funciones usando *this*

- En JavaScript existe un método para llamar a una función, pasándole directamente la referencia del objeto, sin tener que usar variables globales o referenciar al objeto al comienzo de cada función.
- Para conseguir hacerlo necesitamos usar la palabra reservada *this*, que hace referencia siempre al objeto que contiene el código de JavaScript en donde usamos dicha palabra reservada.
- Por ejemplo, si programamos una función para un botón, que al hacer click haga algo, si dentro de esa función usamos la palabra *this*, entonces estaremos haciendo referencia al objeto en el cuál hemos hecho click, que en este caso será el botón. El uso de *this* nos permite evitar usar variables globales, y el programar scripts más genéricos.
- En **el ejemplo siguiente**, cada vez que hagamos click en alguno de los objetos, llamaremos a la función *identificar()* y a esa función le pasaremos como parámetro *this*, que en este caso será la referencia al objeto en el cuál hemos hecho click. La función *identificar()* recibe ese parámetro, y lo almacena en la variable *objeto*, la cuál le permite imprimir todas las referencias al *name*, *id*, *value* y *type*.


```

<!DOCTYPE html>
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8">
<title>Uso de la palabra reservada this</title>
<script type="text/javascript">
function identificar(objeto)
{
    var atrName=objeto.name;
    var atrId=objeto.id;
    var atrValue=objeto.value;
    var atrType=objeto.type;
    alert("Datos del campo pulsado:\n\nName: "+atrName+"\nID: "+atrId+"\nValue: "+atrValue+"\nType: "+atrType);
}
</script>
</head>
<body>
    <h1>Trabajando con this</h1>
    <form id="formulario" action="pagina.php">
        <label for="nombre">Nombre: </label>
        <input type="text" name="nombre" id="nombre" value="Constantino" onclick="identificar(this)"/>
        <label for="edad">Edad: </label>
        <input type="password" name="edad" id="edad" value="55" onclick="identificar(this)"/>
        <label for="pais">País: </label>
        España <input type="radio" name="pais" id="pais1" value="ES" onclick="identificar(this)"/>
        Francia <input type="radio" name="pais" id="pais2" value="FR" onclick="identificar(this)"/>
        <p>Haga click en cada uno de los campos para ver más información. </p>
    </form>
</body>
</html>

```


Envío y validación de formularios

- Validación: comprobar que los datos del formulario son correctos (por ejemplo, email con formato válido).
- Puede hacerse en cliente (más rápida) o en servidor (más segura)
- Categorías típicas de validación:
 1. **Existencia**: comprueba cuando existe o no un valor.
 2. **Numérica**: que la información contenga solamente valores numéricos.
 3. **Patrones**: comprueba que los datos sigan un determinado patrón, como el formato de un e-mail, una fecha, un número de teléfono, un número de la seguridad social, etc.



La validación de un formulario en el lado del cliente puede ahorrar algunas idas y vueltas a la hora de enviar los datos, pero aún así, **tendrás que realizar la validación de datos en el servidor**, puesto que es allí realmente donde se van a almacenar esos datos y el origen de los mismos puede venir por cauces que no hemos programado.

Ver ejemplo de
validación subido en
Moodle



Expresiones regulares

- Las expresiones regulares son **patrones de búsqueda**, que se pueden utilizar **para encontrar texto** que coincida con el patrón especificado.
- Cuando buscamos cadenas que cumplen un patrón en lugar de una cadena exacta, necesitaremos usar expresiones regulares. Podrías intentar hacerlo con funciones de *String*, pero al final, es más sencillo hacerlo con expresiones regulares, aunque su sintaxis es un poco extraña y no muy amigable.

Matrícula Coche:

- Las expresiones regulares se gestionan a través del objeto RegExp. **Sintaxis:**

```
var expresion = /expresión regular/modificador;
```

Ejemplo: expresión regular para encontrar la palabra Aloe Vera

```
var expresion = /Aloe\s+Vera/; \s+ = uno o varios espacios en blanco
```

Expresiones regulares: modificadores y corchetes

| Modifier | Description |
|-------------------|--|
| g | Perform a global match (find all matches rather than stopping after the first match) |
| i | Perform case-insensitive matching |
| m | Perform multiline matching |

Corchetes: para encontrar un RANGO de caracteres (recordar: no cadenas)

| Expression | Description |
|------------------------|---|
| [abc] | Find any character between the brackets |
| [^abc] | Find any character NOT between the brackets |
| [0-9] | Find any character between the brackets (any digit) |
| [^0-9] | Find any character NOT between the brackets (any non-digit) |
| (x y) | Find any of the alternatives specified |

Importante consultar:

https://www.w3schools.com/jsref/jsref_obj_regexp.asp

Caracteres especiales más usados para expresiones regulares (sigue)

| Carácter | Coincidencias | Patrón | Ejemplo de cadena |
|----------|--|-----------|--|
| ^ | Al inicio de una cadena | /^Esto/ | Coincidencia en "Esto es..." |
| \$ | Al final de la cadena | /final\$/ | Coincidencia en "Esto es el final". |
| * | Coincide 0 o más veces | /se*/ | Que la "e" aparezca 0 o más veces: "seeee" y también "se". |
| ? | Coincide 0 o 1 vez | /ap?/ | Que la p aparezca 0 o 1 vez: "apple" y "and". |
| + | Coincide 1 o más veces | /ap+/ | Que la "p" aparezca 1 o más veces: "apple" pero no "and". |
| {n} | Coincide exactamente n veces | /ap{2}/ | Que la "p" aparezca exactamente 2 veces: "apple" pero no "apabullante". |
| {n,} | Coincide n o más veces | /ap{2,}/ | Que la "p" aparezca 2 o más veces: "apple" y "appple" pero no en "apabullante". |
| {n,m} | Coincide al menos n, y máximo m veces | /ap{2,4}/ | Que la "p" aparezca al menos 2 veces y como máximo 4 veces: "apppppple" (encontrará 4 "p"). |
| . | Cualquier carácter excepto nueva línea | /a.e/ | Que aparezca cualquier carácter, excepto nueva línea entre la a y la e: "ape" y "axe". |
| [...] | Cualquier carácter entre corchetes | /a[px]e/ | Que aparezca alguno de los caracteres "p" o "x" entre la a y la e: "ape", "axe", pero no "ale". |
| [^...] | Cualquier carácter excepto los que están entre corchetes | /a[^px]/ | Que aparezca cualquier carácter excepto la "p" o la "x" después de la letra a: "ale", pero no "axe" o "ape". |
| \b | Coincide con el inicio de una palabra | /\bno/ | Que "no" esté al comienzo de una palabra: "novedad". |

Caracteres especiales más usados para expresiones regulares (cont)

| Carácter | Coincidencias | Patrón | Ejemplo de cadena |
|----------|---|---------------|--|
| \B | Coincide al final de una palabra | /\Bno/ | Que "no" esté al final de una palabra: "este invierno" ("no" de "invierno"). |
| \d | Dígitos del 0 al 9 | /\d{3}/ | Que aparezcan exactamente 3 dígitos: "Ahora en 456". |
| \D | Cualquier carácter que no sea un dígito | /\D{2,4}/ | Que aparezcan mínimo 2 y máximo 4 caracteres que no sean dígitos: encontrará la cadena "Ahor" en "Ahora en 456". |
| \w | Coincide con caracteres del tipo (letras, dígitos, subrayados) | /\w/ | Que aparezca un carácter (letra, dígito o subrayado): "J" en "JavaScript". |
| \W | Coincide con caracteres que no sean (letras, dígitos, subrayados) | /\W/ | Que aparezca un carácter (que no sea letra, dígito o subrayado): "%" en "100%". |
| \n | Coincide con una nueva línea | | |
| \s | Coincide con un espacio en blanco | | |
| \S | Coincide con un carácter que no es un espacio en blanco | | |
| \t | Un tabulador | | |
| (x) | Capturando paréntesis | | Recuerda los caracteres. |
| \r | Un retorno de carro | | |
| ?=n | Cualquier cadena que está seguida por la cadena n indicada después del igual. | /la(?= mundo) | Hola mundo mundial. |

El objeto RegExp

```
let re = /ab+c/;
```

```
let re = new RegExp('ab+c');
```

- Es tanto un literal como un objeto de JavaScript, por lo que también se podrá crear usando un constructor: `var expresionregular = new RegExp("Texto Expresión Regular");`
- Usar el literal cuando sabemos que la expresión no cambiará. Una versión compilada es mucho más eficiente.
- Usaremos el objeto cuando sabemos que la expresión regular va a cambiar o cuando vamos a proporcionarla en tiempo de ejecución.

| Propiedad | Descripción |
|------------|--|
| global | Especifica que sea utilizado el modificador "g". |
| ignoreCase | Especifica que sea utilizado el modificador "i". |
| lastIndex | El índice donde comenzar la siguiente búsqueda. |
| multiline | Especifica si el modificador "m" es utilizado. |
| source | El texto de la expresión regular RegExp. |

| Método | Descripción |
|-----------|---|
| compile() | Compila una expresión regular. |
| exec() | Busca la coincidencia en una cadena. Devolverá la primera coincidencia. |
| test() | Busca la coincidencia en una cadena. Devolverá true o false. |

Ejercicio 10 – Expresiones regulares

- a. Escribe una expresión regular para identificar cadenas que contengan "Blog" cualquier cadena y a continuación "Goog". Comprobar su funcionamiento devolviendo verdadero o falso si se cumple con cadenas recogidas desde teclado. Repetir hasta que el usuario escriba un 0.
- b. Validación de un número de Seguridad Social Americano: consiste en 8 dígitos, **generalmente** escritos en tres campos separados por guiones: AAA-GG-SSS. Los tres primeros dígitos se denominan el "número de área". Los dos dígitos centrales son el "número de grupo" y los 3 finales son el "número de serie". Validar que un número facilitado cumpla el formato indicado con o sin guiones ya que son opcionales.

Para cada uno, hacer una versión con el literal y otra con el objeto RegExp.

Más ayuda en:

https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Regular_Expressions

The background of the slide is a dark, artistic composition. On the left, a large, silver-colored film reel is partially visible, with its circular frames and the edges of film strips. On the right, a black clapperboard is positioned diagonally. It features white text and horizontal lines for recording information. The text on the clapperboard includes 'PRODUCTION', 'DIRECTOR', 'CAMERA', 'SCENE', and 'TAKE'. The overall lighting is moody, with highlights on the metallic surfaces of the reel and the white text on the clapperboard.

Fin de la parte 1

Continuará...

Recuerda que el tema está aun incompleto, vamos a por la parte 2