

INTRODUCCIÓN A JAVASCRIPT

JAVASCRIPT EN ACCIÓN

Visualización de datos. Ejemplo de diagrama Sankey

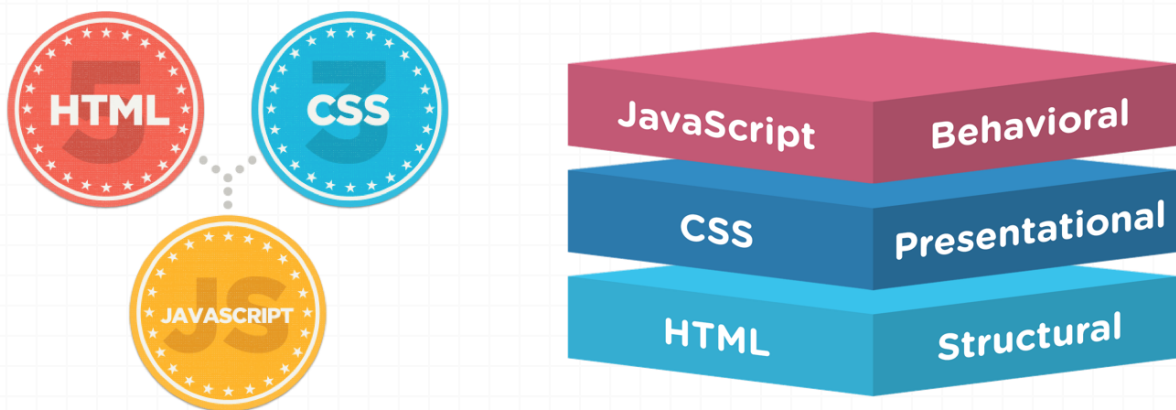
Visualización de secuencias sunburts

Visualizaciones experimentales de Google

[Ver las visualizaciones creadas con JS](#)

PRESENTANDO A JAVASCRIPT

JavaScript es el lenguaje de script nativo de la web.

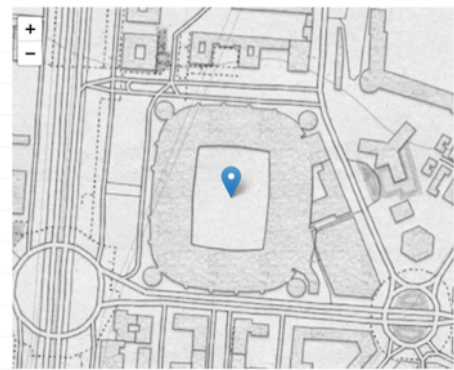
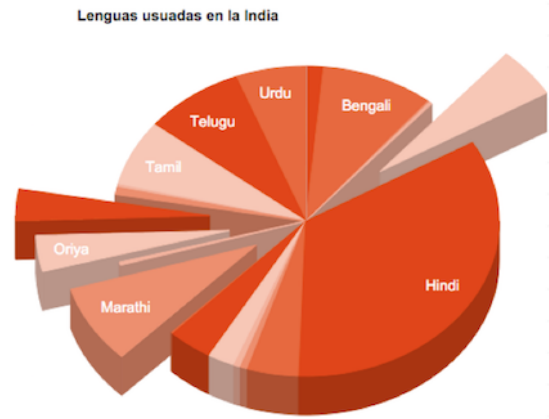
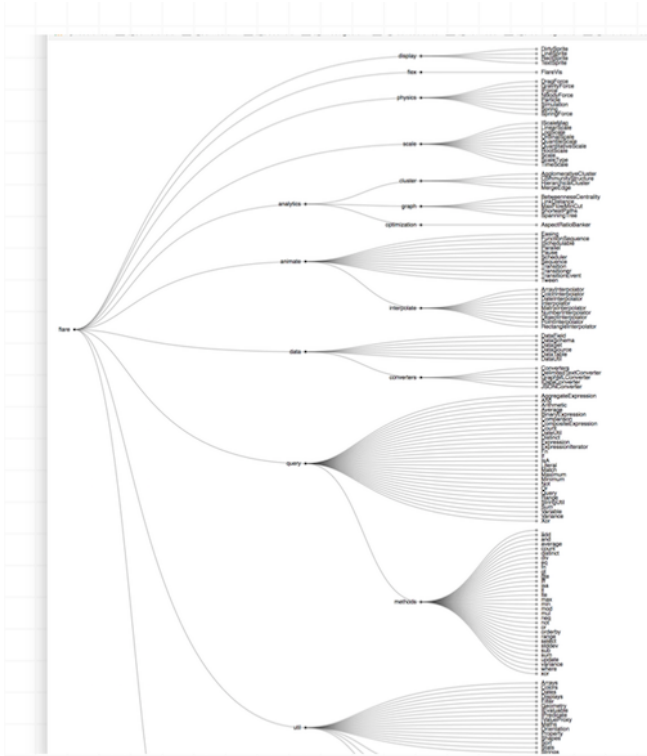


```

1 <script>
2
3   var grid_length = 75;
4   // Note that grid_length in the book is 100.
5   // I reduced it here for smooth performance on mobile devices.
6   var grid = [];
7   var temp_grid = [];
8   var beta = 0.05;
9   var gamma = 0.15;
10
11  function get_random_int(min, max) {
12    return Math.floor(Math.random() * (max - min + 1)) + min;
13  }
14
15  function init_grid() {
16    for (var i = 0; i < grid_length; i = i + 1) {
17      grid[i] = [];
18      for (var ii = 0; ii < grid_length; ii = ii + 1) {
19        grid[i][ii] = "S";
20      }
21    }
22    grid[get_random_int(0,grid_length-1)][get_random_int(0,grid_length-1)] = "I";
23  }
24
25  init_grid();
26
27  draw_grid(grid,["S","#dcdcdc","I","#c82605","R","#6fc041"]);
28

```

Su aprendizaje supone introducirse en un lenguaje de programación, y aunque pueda parecer complicado de entrada, luego nos permitirá saber cómo modificar código ya escrito o realizar visualizaciones muy atractivas mediante el uso de librerías y API que se basan en JavaScript.



¿Para qué nos sirve JavaScript en un proyecto periodístico?

- Para tener unas nociones generales sobre la programación con scripts.
- Para saber cómo funciona, de forma global, la interactividad en la manipulación de datos y de las visualizaciones.
- Para comprender el código resultante de los sistemas de datos y visualización, y saber manipular determinados parámetros.
- Para comenzar a adentrarnos en la creación de visualizaciones basadas en librerías JS como jQuery
- Para hacer scraping de datos con Javascript (Jquery)

¿Cómo vamos a trabajar?

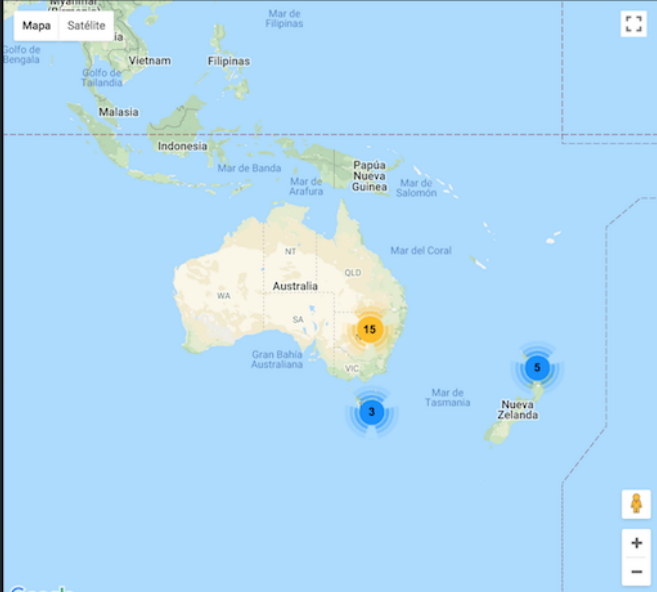
Para trabajar con JavaScript utilizaremos editores de código online como JsFiddle.net. Podrás acceder a los ejemplos y ejercicios de dos maneras:

A) A través de los enlaces que encontrarás en cada ejercicio y que te darán acceso al entorno online de edición:

```
HTML ▼
1 <div id="map"></div>
2 <!-- Habría que reemplazar el valor de la API Key por la nuestra -->
3 <script async defer
4 src="https://maps.googleapis.com/maps/api/js?
key=AizaSyCkU0d25y7hMm0yrcQ0cVlwzdm6M8s5qk&callback=initMap">
5 </script>
6 <script
7 src="https://developers.google.com/maps/documentation/javascript/examples/
markerclusterer/markerclusterer.js"></script>

CSS ▼
1 /* Es necesario indicar siempre la altura del mapa para definir el tamaño del
2 div que contiene al mapa */
3 #map {
4 height: 100%;
5 } /* Opcionalmente se puede indicar que el contenido vaya a ventana completa */
6 html, body {
7 height: 100%;
8 margin: 0;
9 padding: 0;
10 }

JavaScript + No-Library (pure JS) ▼
1 function initMap() {
2
3 var map = new google.maps.Map(document.getElementById('map'), {
4 zoom: 3,
5 center: {lat: -28.024, lng: 140.887}
6 });
7
8 // Creamos un array con los caracteres alfabéticos que usaremos para
9 etiquetar los marcadores
10 var labels = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
11
12 /* Para añadir los marcadores al mapa utilizamos una función, con dos
13 parámetros (location e i)
14 - Return devolverá la marcador la información proveniente de dos
15 arrays: 1. El anterior que hemos creado con las etiquetas. 2. El de
16 locations que creamos a continuación
17 */
18 var markers = locations.map(function(location, i) {
19 return new google.maps.Marker({
20 position: location,
21 /* La etiqueta se obtiene comenzando por el primer elemento del
22 array.
23 Obtenemos el módulo entre i y el total de elementos del array con
24 la propiedad length
25 para que vaya asignando, sucesivamente, el resto de valores del
26 array*/
27 label: labels[i % labels.length]
28 });
29 });
30
31 // Añadimos un marker clusterer (librería) para gestionar los
32 marcadores.
33 var markerCluster = new MarkerClusterer(map, markers,
34 {imagePath:
```



B) Manipular los ejemplos de manera directa a través de objetos embebidos, en el que podrás trabajar en las diferentes partes del código accediendo a las diferentes pestañas disponibles.

¿QUÉ ES JAVASCRIPT?

¿Qué es?

Un lenguaje de programación **interpretado** por los navegadores en tiempo real.

COMPILADOS

Traducen a código máquina, creando un archivo traducido para una ejecución rápida.

Requieren que las instrucciones, sean traducidas, esto lo hace más eficiente.

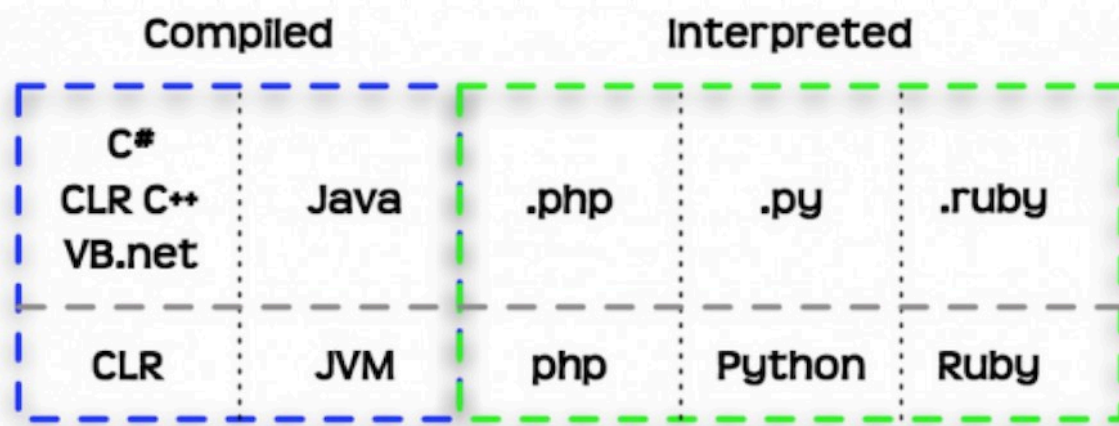
El procesador sólo entiende un lenguaje que se denomina "lenguaje máquina".

INTERPRETADOS

Las instrucciones se traducen una a una, cada vez que se ejecute el programa

Son típicamente unas 10 veces más lentos que los programas compilados.

Facilidad para lograr independencia de plataformas y menor tamaño de programa.



Lenguajes compilados vs Lenguajes interpretados

JavaScript es un dialecto del estándar [ECMAScript](#) (versión 6)

Responde al modelo de [Programación orientada a objetos \(POO\)](#). Aunque sigue el paradigma de programación basado en objetos, trabaja con prototipos, aunque las versiones actuales permiten el manejo de [clases \(class\)](#). Al trabajar bajo el modelo de POO usa técnicas que le dan una importante versatilidad como, por ejemplo:

- **Modularidad.** Las aplicaciones se pueden subdividir en partes, conocidas como módulos.
- **Herencia.** Las clases pueden heredar propiedades y métodos de otra clase, lo que permite reutilizar el código.
- **Control del DOM.** Permite interactuar con la página web mediante [DOM](#)

Puede funcionar tanto:

- En el lado cliente (client-side) como parte del navegador web
- En el lado servidor (vg. [Node.js](#))

Aunque el nombre es similar a JAVA, tiene que ver poco con este otro lenguaje de programación, tanto en las semánticas como en sus propósitos, que son diferentes, así que debemos no confundirlos.

Ejercicio

1. Abre el editor de código en [JSFiddle](#) con [el ejercicio](#).

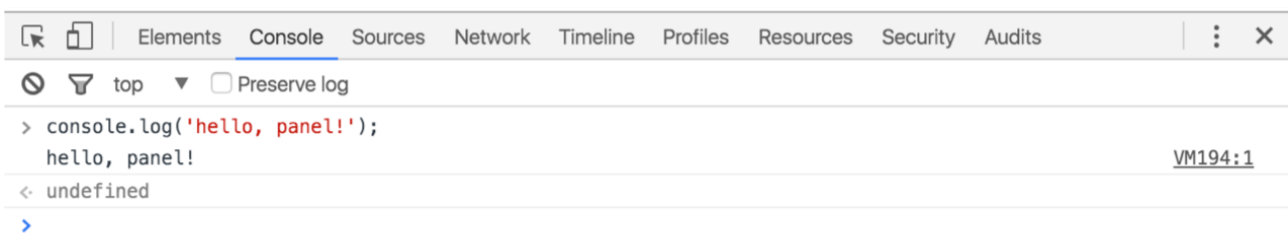
2. Revisa el código html. Fíjate que el código JS aparece en el fichero HTML. Prueba el resultado que produce el botón. ¿Cómo crees que funciona comprobando el código?

DEPURANDO. CONSOLA DEL NAVEGADOR

La **consola del navegador** nos permite revisar y depurar los errores de código.

Todos los navegadores actuales disponen de ella. Para abrirla:

- En Chrome: Ctrl+Shift+J (Windows) o Cmd+Alt+I (Mac)
- En Firefox: Control + ⇧ + J (Windows)



Consola del navegador en Chrome

Ejercicio

Vamos a comprobar, a través de la consola de Chrome, dónde aparece JavaScript.

1. Carga una página, por ejemplo de un periódico como El Mundo o El País. Abre la Consola mediante las funciones rápidas que hemos visto anteriormente. Otra opción es buscar en el menú del navegador. En el caso de Chrome: Más herramientas > Herramientas para desarrolladores
2. Fíjate en la pestaña Elements -dónde aparece las etiquetas de script. Localiza las etiquetas de script, la etiqueta **noscript**, etc.



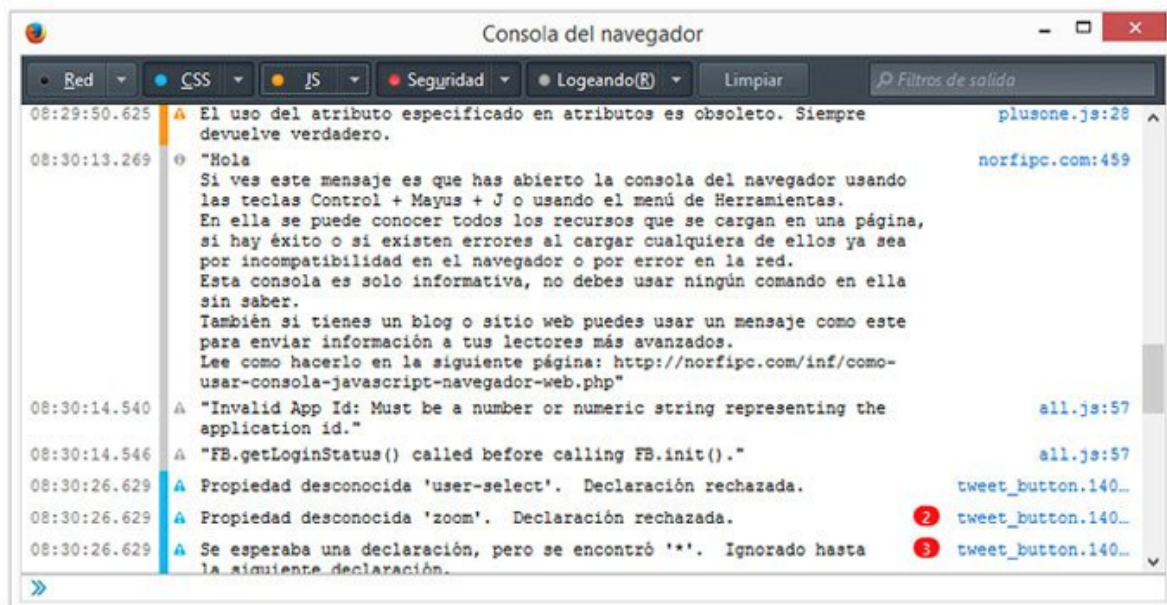
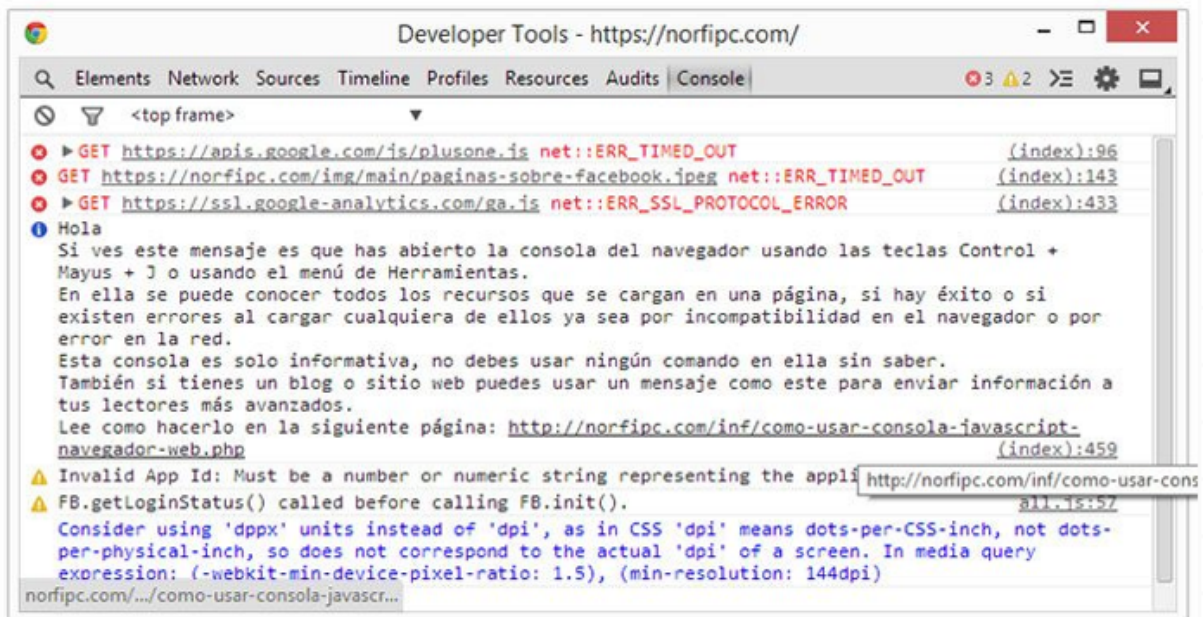
Ejemplo de código en la consola del navegador

Utilidad de la consola. Para qué sirve.

La consola nos sirve para probar el funcionamiento de una página o una aplicación de forma que: a) podamos detectar errores de código; y b) hacer pruebas con JS

Permite:

- Revisar información sobre errores o alertas
- Utilizar el inspector de código para depurarlo
- Ejecutar expresiones o comandos de JS



¿DÓNDE APARECE?

Como sucede en CSS, el JS puede cargarse:

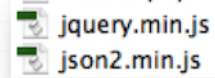
1. Como fichero independiente: [fichero.js](#)
 - Mediante el atributo `src`: `<script src="fichero.js"> </script>`
2. En el `<head>` o `<body>` de las páginas

```
<html><body>

<script type="text/javascript">

  // El código va aquí...

</script>
</body></html>
```



3. En atributos designadores de manejadores de eventos (*event handlers*) dentro del HTML:

- `<input type="button" value="clickMe" onClick='alert("gracias por hacer clic");'/>`

JavaScript puede cargarse en la página web a través de diferentes opciones.

¿PARA QUÉ LO UTILIZAMOS?

Es un lenguaje que permite **dotar de interactividad** a las páginas web. Algunas de las tareas básicas que puede realizar son:

1. Escribir en HTML
2. Reaccionar a eventos
3. Modificar elementos HTML
4. Validar entrada de datos
5. Cambiar o modificar atributos

Además, mediante la [API JS de HTML5](#) podemos acceder a recursos adicionales: cámara, almacenamiento de datos, creación de gráficos, flujo de datos con servidores...)



Permite acceder a información en internet: por ejemplo, buscar y obtener las palabras más populares en la red X de un tema, o para hacer scraping de web utilizando soluciones como [Node.js](#), [Artoo.js](#) o [pjsctrace](#))

También permite organizar y presentar datos como, por ejemplo, automatizar el trabajo de las hojas de cálculo; o la visualización de datos.

Veamos un ejemplo de cómo funciona JavaScript

Veamos cómo JS aporta interactividad a una web,

1. Accede a la web de [España en Llamas](#). Navega por el mapa, usa los filtros, etc.
2. Ahora deshabilita Javascript desde las [opciones del navegador](#) o utilizando alguna herramienta como [WebDeveloper Toolbar](#). Comprueba qué sucede. ¿Puedes navegar el mapa? ¿Compartir en redes sociales?

Y ahora cómo funciona JavaScript en un entorno de ejecución en servidor, como Node.js

En este ejemplo, usamos Node.js y dos librerías (axios y cheerio) para hacer scraping web.

1. Accede a este ejemplo [Scraping con Node.js en Replit.com](#) (Replit es una plataforma en línea para desarrollo similar a jsFiddle)
2. Vamos a ver cómo el código nos permite obtener los titulares de las últimas noticias publicadas en una web, en este caso de la URJC.

EJERCICIOS BÁSICOS DE FUNCIONAMIENTO EN NAVEGADOR

Vamos a realizar ahora algunos ejercicios básicos para conocer estas funciones de JavaScript operando en el navegador.

1. Escribir en HTML

Una de las cosas que puede hacer JS es escribir contenido en HTML. Observemos [este ejemplo](#).

Example

```
document.write("<h1>This is a heading</h1>");  
document.write("<p>This is a paragraph</p>");
```

Ejercicio

1. Sobre el ejemplo anterior, añade un enlace a la home del periódico El Mundo (www.elmundo.es)
2. Prueba a incluir otro enlace a otra web distinta.

[Solución](#)

2. Reaccionar a eventos

Example

```
<button type="button" onclick="alert('Welcome!')">Click Me!</button>
```

Otra función es reaccionar a eventos creados por el usuario (hacer clic, pasar por encima de una zona...), por el propio navegador (cargar la página), etc. como sucede en [este ejemplo](#).

Ejercicio

1. Modifica el ejemplo anterior para que el mensaje de alerta muestre tu nombre.

[Solución](#)

3. Modificar contenido en HTML

Otra función es modificar contenido que ya [está cargado en el html](#).

Example

```
x=document.getElementById("demo") //Find the element  
x.innerHTML="Hello JavaScript"; //Change the content
```

Ejercicio

En este ejemplo hay algo que no funciona. Revísalo y ajusta lo que esté mal para que funcione el cambio de contenido.

[Solución](#)

El id no es correcto. Deben de tener el mismo nombre.

4. Validar la entrada de datos

JavaScript es muy útil para validar si la entrada de datos en los campos de un formulario son los adecuados, aquellos que por formato o tipo se esperan recibir.

[En este ejemplo](#), podemos comprobar si los datos que se introducen están comprendidos dentro de un rango. En caso contrario, saltará una alerta indicando que no son correctos.

Example

```
if isNaN(x) {alert("Not Numeric");}
```

isNaN (x) Es un objeto global de Javascript que evalúa un argumento para determinar si es un número

Ejercicio

Modifica el ejemplo anterior para que la validación sea para números comprendidos entre 10 y 20.

Solución

Ejemplo de validación de contraseña con condicionales

En este [ejemplo podemos ver un caso más complejo](#) en el que validamos una contraseña en un formulario para que cumpla tres condiciones.

5. Cambiar o modificar atributos

En [este ejemplo](#) generamos un efecto de sustitución modificando el atributo src de una imagen.

Otra posibilidad es cambiar determinados atributos de los elementos html. Aquí las posibilidades son muy amplias, dado que podemos modificar los valores de cualquier atributo, desde el color de un texto, el tipo de fuente, etc.

```
function changeImage()
{
element=document.getElementById('myimage')
if (element.src.match("bulbon"))
{
element.src="pic_bulboff.gif";
}
else
{
element.src="pic_bulbon.gif";
}
}
</script>
```

Ejercicio

Modifica el valor del atributo src con estas imágenes que realizan un efecto de sustitución similar

- <http://comunicaciondigital.es/wp-content/master/lighton.png>
- <http://comunicaciondigital.es/wp-content/master/lightoff.png>

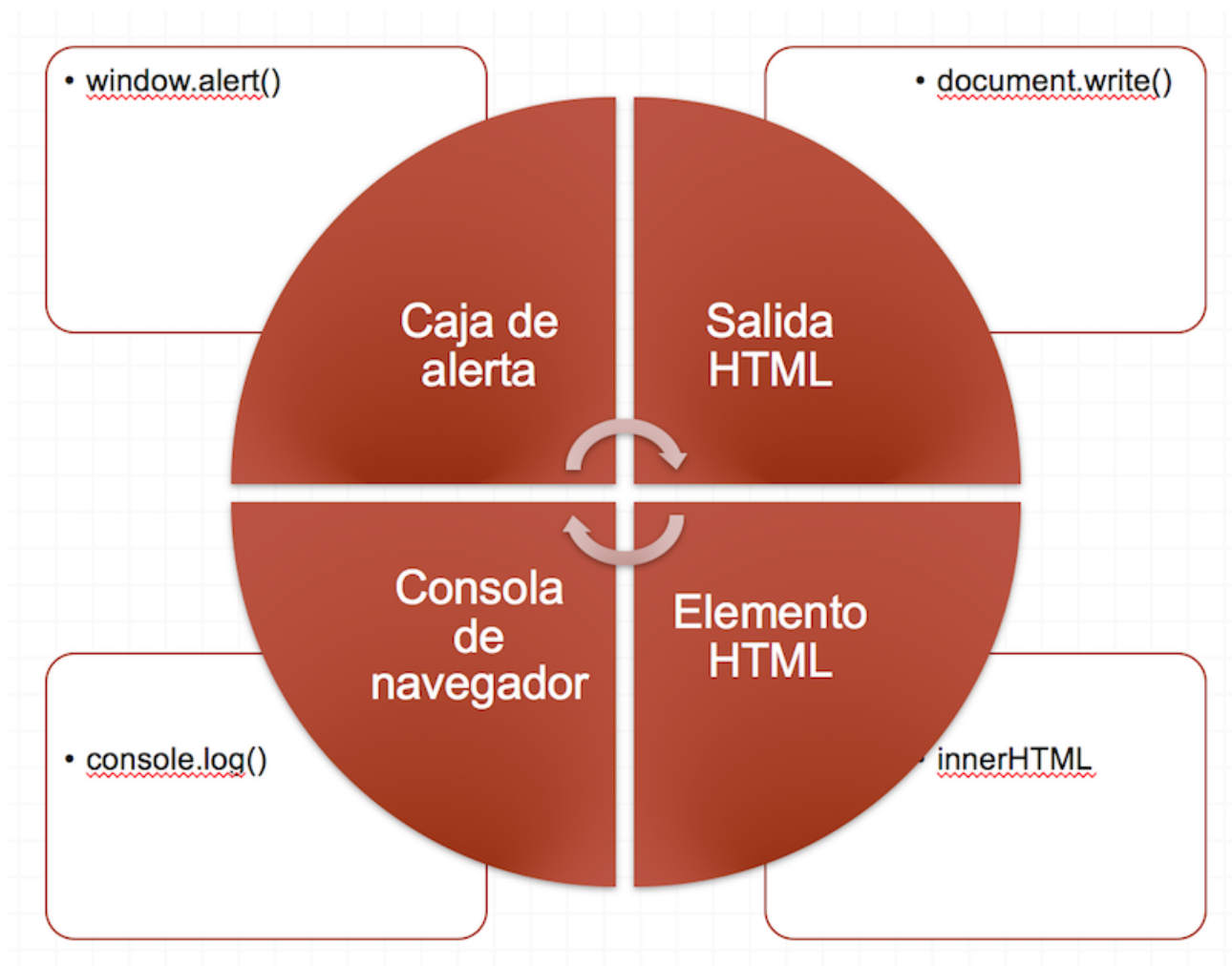
Solución

Enciende o apaga la luz (la primera imagen es el primer botón), la segunda es para img y el segundo botón

¿QUÉ RESULTADOS PRODUCE?

Javascript utiliza varios procedimientos o displays de escritura:

- Caja de alerta – `window.alert()` [[Ver ejemplo](#)]
- Salida html – `document.write()` [[Ver ejemplo](#)]
- Elemento html – `inner.html()` [[Ver ejemplo](#)]
- Consola de navegador – `console.log()` [[Ver ejemplo](#)]



RECURSOS PARA PROFUNDIZAR

[Web de Sarah Drasner](#)

Web de Sarah Drasner, desarrolladora frontend y divulgadora que escribe artículos y libros sobre temas como JavaScript o animaciones SVG.

[Web de Seb Lee](#)

Web de Seb Lee, artista digital de Reino Unido que practica lo que se llama «creative coding», es decir, la fusión de la programación con el arte.

[Web de Lea Verou](#)

Web de Lea Verou, divulgadora y desarrolladora frontend griega que es investigadora en el MIT, y lleva años realizando proyectos alrededor de CSS y JavaScript.

[Web de Remy Sharp](#)

Web de Remy Sharp, desarrollador frontend de Reino Unido especializado en JavaScript.

[Web de Jenn Schiffer](#)

Web de Jenn Schiffer, desarrolladora de web apps e ilustradora de pixel art que trabaja en Glitch.com, una comunidad de desarrollo de aplicaciones web.

[Web de Sacha Grief](#)

Web de Sacha Grief, diseñador y desarrollador nacido en París, que vive en Osaka y que escribe artículos y graba podcasts de desarrollo.

EJEMPLOS

[Ejemplo de visualización](#)

[See the Pen](#)

Vue Time Comparison by Sarah Drasner ([@sdras](#))

on [CodePen](#).

[Grupo Ciberimaginario](#) | Manuel Gertrudix - Alejandro Carbonell |
2024/2025 | Esta obra está bajo una Licencia Creative Commons Atribución 4.0
Internacional. Los contenidos citados se ajustan a lo regulado en el art. 32 del TRLPI de
España

