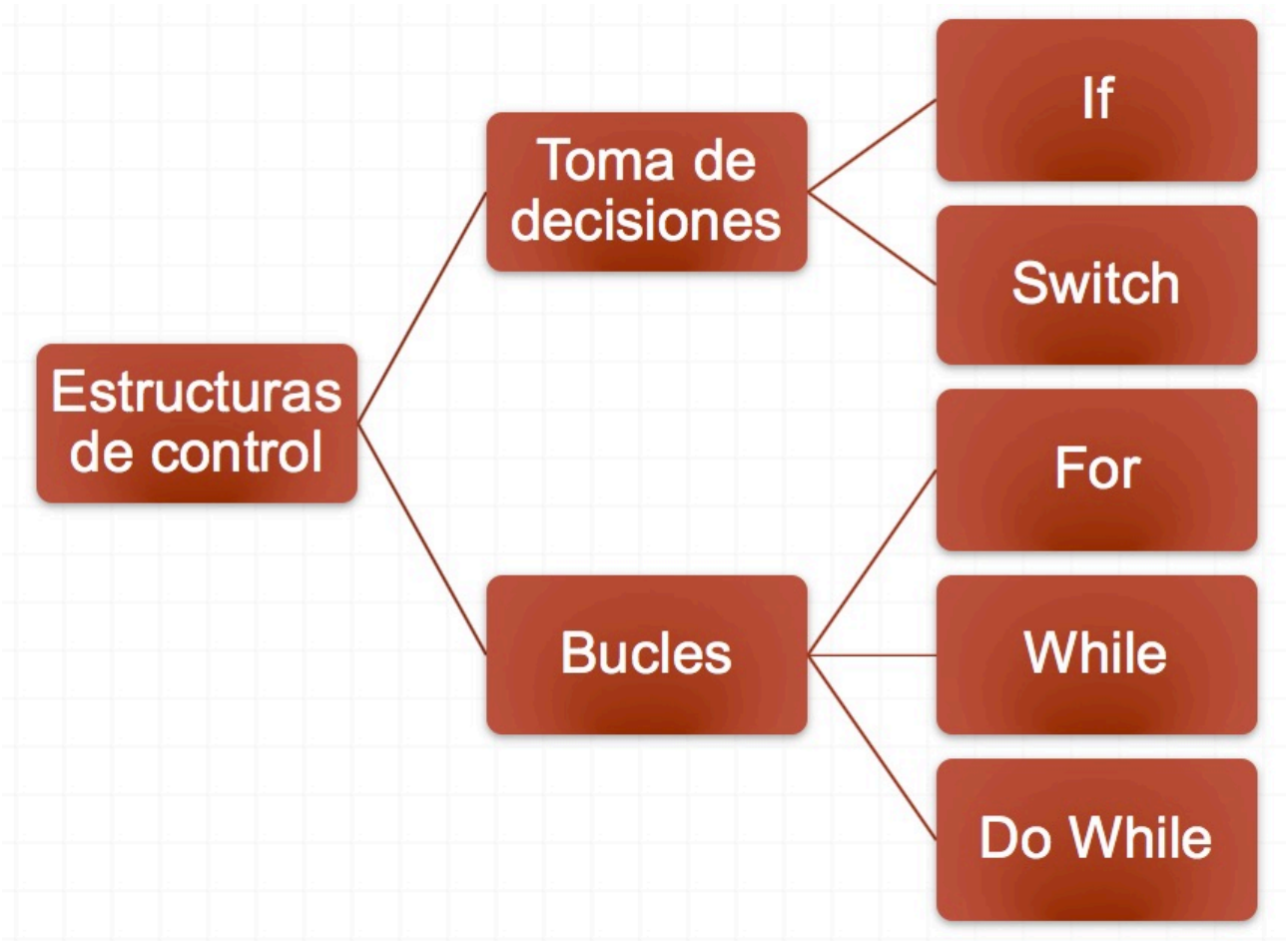
ESTRUCTURAS DE CONTROL, DOM Y JSON

Estructuras de control

Estructuras condicionales



IF

Permite tomar decisiones en función del estado de las variables.

En JS se utilizan:

- **If** – El código se ejecuta si la condición es verdadera, como en este ejemplo.
- **Else** – Especifica el código que se ejecuta si la condición es falsa

Ejercicio If

Modifica la función anterior para que muestre el resultado de un año:

- Utiliza el método getFullYear() para el objeto Date()
- Haz una comparación para un año para que salga un mensaje: “Estamos en el año 2018”

Solución

else IF

else IF especifica una nueva condición si la primera es falsa.

Se evalúa una segunda y, en caso de que también sea falsa, pasa a un else final.

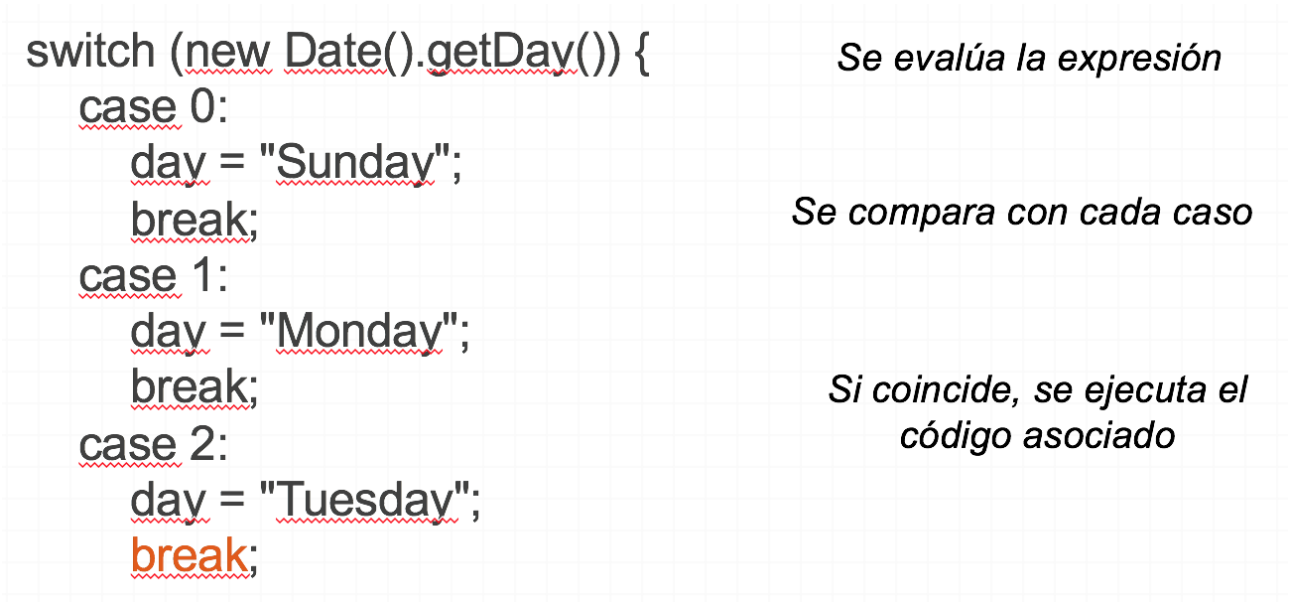
Ejercicio else If

Modifica la función anterior para que se adapte ahora al día de la semana:

- Utiliza el método getDay() para el objeto Date()
- Haz una comparación en función del día en el que estamos: 0-1 (principio de semana), 2-4 (mediados de semana), 5-6 (fin de semana)

Solución

Switch



Switch (interruptor, conmutador) se utiliza para que sucedan cosas diferentes en función de las diferentes condiciones que se den.

Break se utiliza para que, en el caso de que la sentencia sea correcta, el bucle pare la evaluación y no continúe examinando las siguientes. Si no se indica el break, podría quedarse “colgado” en bucle intentando encontrar la sentencia verdadera.

Ejercicio. Switch

Modifica la función anterior para que señale el mes en el que estamos

Ideas:

- Usa el método getMonth() para el objeto Date()
- Modifica la variable day por month y pon unos cuantos casos (mese del año)
- No hacen falta muchos (estamos cerca del principio)

Solución

Recuerda que en JS la numeración es base cero.

Bucles. Loop

Los Loops se utilizan para ejecutar un código, que necesita ser repetido un número de veces, por ejemplo con un valor diferente cada vez.

Hay varios tipos de Loop:

- **for** – Repite un bloque de código un número de veces
- **for / in** – Loop en las propiedades de un objeto
- **while** – Repite en un bloque con una condición específica de verdad
- **do / while** – Repite un bloque con una condición específica de verdad

For repite una o más instrucciones varias veces.

Tiene tres partes:

1. **Inicialización**: Se ejecuta antes de que el loop empiece y suele llevar la variable que indica las veces que se ejecutará el bucle
2. **Condición**: Define la condición que se evalúa para que el loop siga funcionando
3. **Actualización**: Indica los cambios en la variable cada vez que termina una iteración del bucle

Entre llaves se ubican las sentencias que deben ejecutarse en cada iteración.

```
for (inicialización; condición; actualización) {
    //sentencias a ejecutar en cada iteración
}
```

En el siguiente ejemplo podemos ver un ejemplo de loop:

1. Declaramos la variable text y la inicializamos como string vacía
2. Declaramos la variable i
3. En los parámetros del bucle, establecemos los valores de inicialización de i, la condición, y la actualización
4. En el for, se establece una sentencia que determina que, para cada iteración, se actualiza la variable text y la variable i y se muestra el valor en el id “demo”

Ejercicio. Loop for

Modifica la función anterior para que cree un loop mediante for que:

- Comience en 200
- Imprima en pantalla “Estudiante nº: “
- Con el número correlativo descendente desde 200 hasta 1

Solución

Ejercicio. Loop para cargar elementos de un array unidimensional

Sobre la base de este código, vamos a crear un bucle que cargue los elementos de un array unidimensional.

Pistas para el for:

- Inicializa la variable (i) en 0
- La condición debe ser menor o igual al número máximo de elementos del array (food) [Recuerda que los array son base 0]
- Actualiza el incremental de 1 en 1

Solución

Loop para cargar elementos de un array multidimensional

Veamos ahora cómo hacerlo con un array multimensional. En el ejemplo vamos a ver cómo funciona dos bucles para acceder a un array multimensional y crear una tabla con los datos de este.

```
var my2Darray = [
        ["bob", "alex", "joe", "wendy"],
        ["teresa", "jodi", "sara", "peter"],
        ["jack", 44, 90],
        [true, false]
];
```

	0	1	2	3
0	bob	Alex	Joe	wendy
1	teresa	Jodi	Sara	Peter
2	Jack	44	90	
3	True	false		

Aplicaciones de for en mapas. Cargar un array GeoJSON y dibujar marcadores en un mapa

Hasta ahora hemos visto ejemplos algo abstractos del uso de los bucles, pero en el último ya nos hemos aproximado a un ejemplo práctico en el que se cargan datos para armar una tabla.

```
GeoJSON y se ubica cada marcador en cada set de coordenadas
window.eqfeed_callback = function(resultados) {
  for (var i = 0; i < resultados.features.length; i++) {
    var coords = resultados.features[i].geometry.coordinates;
    var latLng = new google.maps.LatLng(coords[1],coords[0]);
    var marker = new google.maps.Marker({
      position: latLng,
      map: map
    });
  };
  /* Explicación de la función:
  - Creamos una función con un parámetro (resultados), que es la
variable que va a recibir los valores
  - Inicializamos el bucle con 0, para que vaya al primer ítem del
array (recordamos que los array son de base cero)
  - Le indicamos como condición que siga recorriendo el array  (i++)
mientras i sea menor que el número de features del fichero GeoJSON
(i<resultados.features.lenght)
  - Mientras funcione el bucle:
     . la var coords mandará a la función el valor que encuentre, para
```



Veamos ahora cómo se usa el for para importar datos automáticamente en un mapa mediante el uso de un loop. El ejemplo siguiente hace uso de la API de Google Maps y cargar los datos formateados mediante GeoJSON, que es un formato que veremos más adelante, pero que ya avanzamos que sirve para georeferenciar la información.

En los comentarios del código se explica el detalle para realizar la carga, pero este sería el resumen fundamental de la función:

- Creamos una función con un parámetro (resultados), que es la variable que va a recibir los valores
- Inicializamos el bucle con 0, para que vaya al primer ítem del array (recordamos que los array son de base cero)
- Le indicamos como condición que siga recorriendo el array (i++) mientras i sea menor que el número de features del fichero GeoJSON (i<resultados.features.lenght)
- Mientras funcione el bucle: a) la var coords mandará a la función el valor que encuentre, para cada item del array (features [i]) los valores de geometry.coordinates; b) la var latLng creará un nuevo valor de las coordenadas (1 y 0, Lat Long); y c) la var marker creará un nuevo marcador con los valores latLng en el mapa
- Con ello irán apareciendo en el mapa todos los marcadores

Bucles for/in

```
var person = {fname:"John", lname:"Doe", age:25};

var text = "";
var x;
for (x in person) {
    text += person[x];
}
```

For / In es un bucle dentro de las propiedades de un **objeto.**

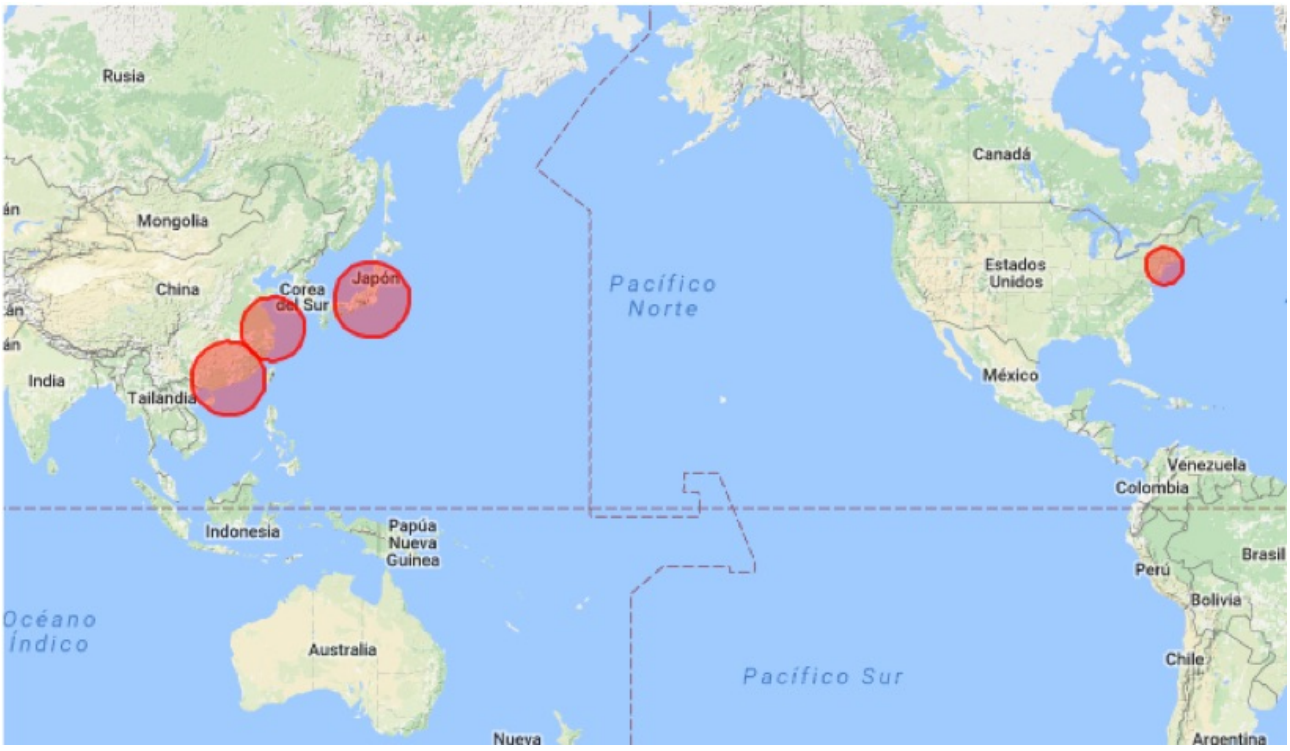
Es igual que for, pero en este caso, se utilizan las propiedades definidas para

un objeto determinado.

Las propiedades del objeto están en formato JSON.

Caso de uso de Bucle for / in: Mostrar datos en un mapa

Vamos ahora a cargar datos de un objeto mediante for/in para crear círculos en un mapa.



Para este ejemplo utilizamos como fuente algunos datos de la Wikipedia sobre aglomeraciones urbanas y su densidad poblacional

1. Primero se crea una variable con los datos de las ciudades, incluyendo su posición y su población.
2. Luego creamos el mapa.
3. Mediante un for, construimos el círculo para cada ciudad, en función del valor de su población definido en la variable citymap, y añadimos el círculo de cada ciudad en el mapa.

Ejercicio.

Sobre la base del ejemplo anterior, vamos a probar a incluir alguna otra ciudad.

1. De la página de wikipedia sobre aglomeraciones urbanas coge el valor de Madrid.
2. Incluye en el código los datos de la ciudad de Madrid para comprobar la diferencia.
3. Cambia los valores del círculo, los colores, y el multiplicador de la

Solución

Caso de uso de Bucle for / in: Cargar datos de una tabla

Veamos ahora en este ejemplo cómo crear una **tabla html con filtro y un buscador** usando for/in.

Tras crear la tabla en html con todos los datos, el loop for/in funciona como un bucle que recorre todas las filas de la tabla y nos sirve para ocultar las que no coinciden con la consulta de búsqueda.

Ejercicio.

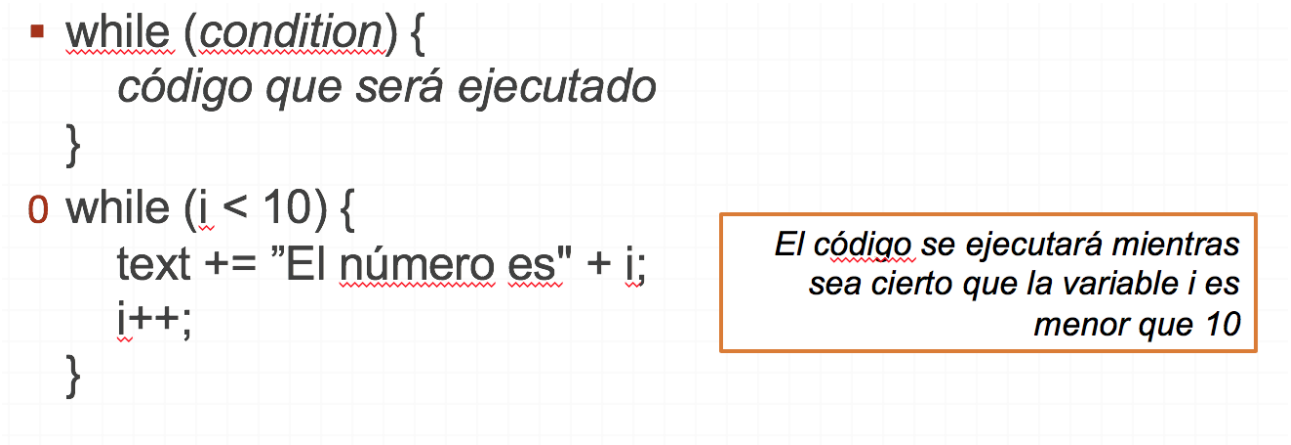
Sobre el ejemplo anterior vamos a realizar una modificación para ver cómo podríamos utilizar este caso para nuestra web, cargando otros datos.

1. Abre el ejemplo anterior para crear una copia.
2. Descarga de Datos.gob.es el set de datos de Monumentos de Cáceres en formato CSV
3. Convierte el CSV con el servicio Convertcsv en una tabla HTML, seleccionando solo la columna 8 [Puedes limitar el número de elementos que se cargan. Por ejemplo: a) En Choose input options limita la carga a 30 filas; b) En Choose outpuy options limita el display a las columnas que nos interesan (en el ejemplo, dejamos sólo 2: 5,6)]
4. Copia el resultado html de la tabla de <tr> a </tr> y sustitúyelo por los datos que tenía la tabla anterior.
5. En el ejemplo anterior la búsqueda se realizaba solo en la primera columna. Ahora puedes probar para que realice la búsqueda en las dos columnas que hemos dejado. Para ello, en el script, dentro del for, ponemos que la búsqueda la haga en las dos columnas, idncando [0, 1] en la línea 13

Solución

While

While ejecuta el loop hasta que la condición especificada es verdadera.



For lo usamos normalmente cuando tenemos una previsión del número de iteraciones que necesitamos, como en lo casos que hemos visto de extraer datos de un arreglo.

While se usa cuando desconocemos el número de ciclos que necesitamos y usamos una variable “centinela” (está pendiente) hasta que se cumpla la condición del bucle.

Ejercicio. While

Sobre el ejercicio anterior, vamos a crear un bucle con while que en este caso haga un conteo descendente desde 1000 hasta 0, incluido, de 100 en 100

Ideas:

- Debe inicializar la variable (i) en 1000
- Establecer la condición para (i) mayor o igual a 0

Solución

Evaluación de condiciones con while

Otro uso de while es ejecuta el loop hasta que la condición especificada como verdadera sea resuelta por un evento del usuario.

Por ejemplo, cuando tenemos un formulario en el que solicitamos al usuario que introduzca el valor. Hasta que este no introduce el valor correcto, el bucle **sigue evaluando la condición**.

Veamos este ejemplo en el que se nos pide que introduzcamos un valor concreto a través de un cuadro de diálogo que utiliza la función .

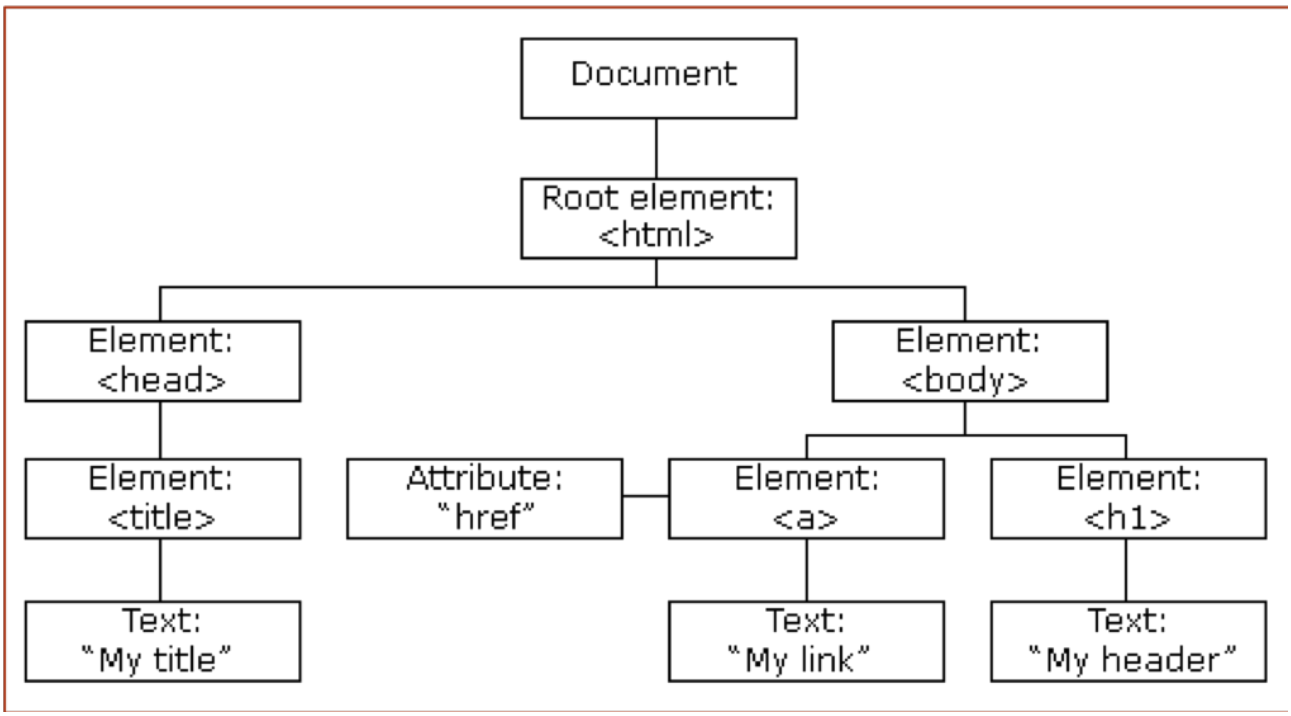
1. Declaramos la variable color y la inicializamos como string vacío
2. En el while indicamos la condición: Mientras el texto del string sea diferente (!=) a rojo, sigue el bucle evaluando
3. Lanza con la función prompt la ventana emergente para introducir un valor
4. En el momento que indique rojo, que es el valor de la condición, se cierra la ventana

DOM

¿Qué es el DOM?

DOM (**D**ocument **O**bject **M**odel) es la representación del árbol de objetos que componen una página web HTML

Todos los elementos son considerados como un nodo: *window y document, elementos y atributos html, texto y contenido de los nodos…*



El DOM es una **Interfaz de Programación** para los documentos HTML y XML

Es una representación estructurada del documento que permite que los programas puedan acceder y modificar:



Javascript usa el DOM para acceder al documento y sus elementos, y poder manipularlos

Funcionamiento del DOM

Cada **objeto** puede activar varias interfaces (*formas de manipular dicho objeto*) mediante métodos o propiedades

El listado de métodos es muy amplio, y puede afectar al Objeto documento, al Objeto Elemento o al Objeto Atributo.

Objeto documento

Property / Method
document.activeElement
document.addEventListener()
document.adoptNode()
document.anchors
document.applets
document.baseURI
document.body
document.close()
document.cookie

Objeto Elemento

Property / Method
element.accessKey
element.addEventListener()
element.appendChild()
element.attributes
element.blur()
element.childElementCount
element.childNodes
element.children

Objeto Atributo

Property / Method
attr.isId
attr.name
attr.value
attr.specified
nodemap.getNamedItem()
nodemap.item()
nodemap.length
nodemap.removeNamedItem()
nodemap.setNamedItem()



Accediendo al árbol DOM, JavaScript puede realizar **cualquier cambio** dentro del documento HTML.

Todos los elementos HTML son considerados objetos:

- Con sus propiedades
- Con sus métodos
- Sus contenidos

En este primer ejemplo accedemos al atributo title de la página para coger su contenido y mostrarlo en la página.

En este otro ejemplo accedemos a un atributo css para modificar el background color de la página en función del lugar sobre el que pasa el ratón (el fondo coge el color del cuadro sobre el que está activo el foco).

Eventos en el DOM



Las acciones que realizamos sobre los elementos del DOM pueden, como en otros casos, activarse a través de diferentes eventos.

Veamos aquí un ejemplo en la API de Google Maps, con el método *addListener* y el evento .

En este caso se muestra un marcador centrado en el Congreso de los Diputados. Cuando hacemos clic sobre el marcador, se abre una ventana *popup*.

Ejercicio. Eventos DOM

Sobre el ejemplo anterior, sustituye en el infowindow el texto por un vídeo.

Utiliza este código para insertar el vídeo: *<iframe width=»560″ height=»315″ src=»https://www.youtube.com/embed/p0efuzumOCw» frameborder=»0″ allowfullscreen></iframe>*

Solución

JSON y GeoJSON

¿Qué es JSON y para qué sirve?

JSON es el acrónimo de **J**avaScript **O**bject **N**otation.

Es un lenguaje independiente con una sintaxis basada en Javascript para *almacenamiento e intercambio de datos.*

Se utiliza en aplicaciones AJAX y es una alternativa a XML **más sencilla** de usar.

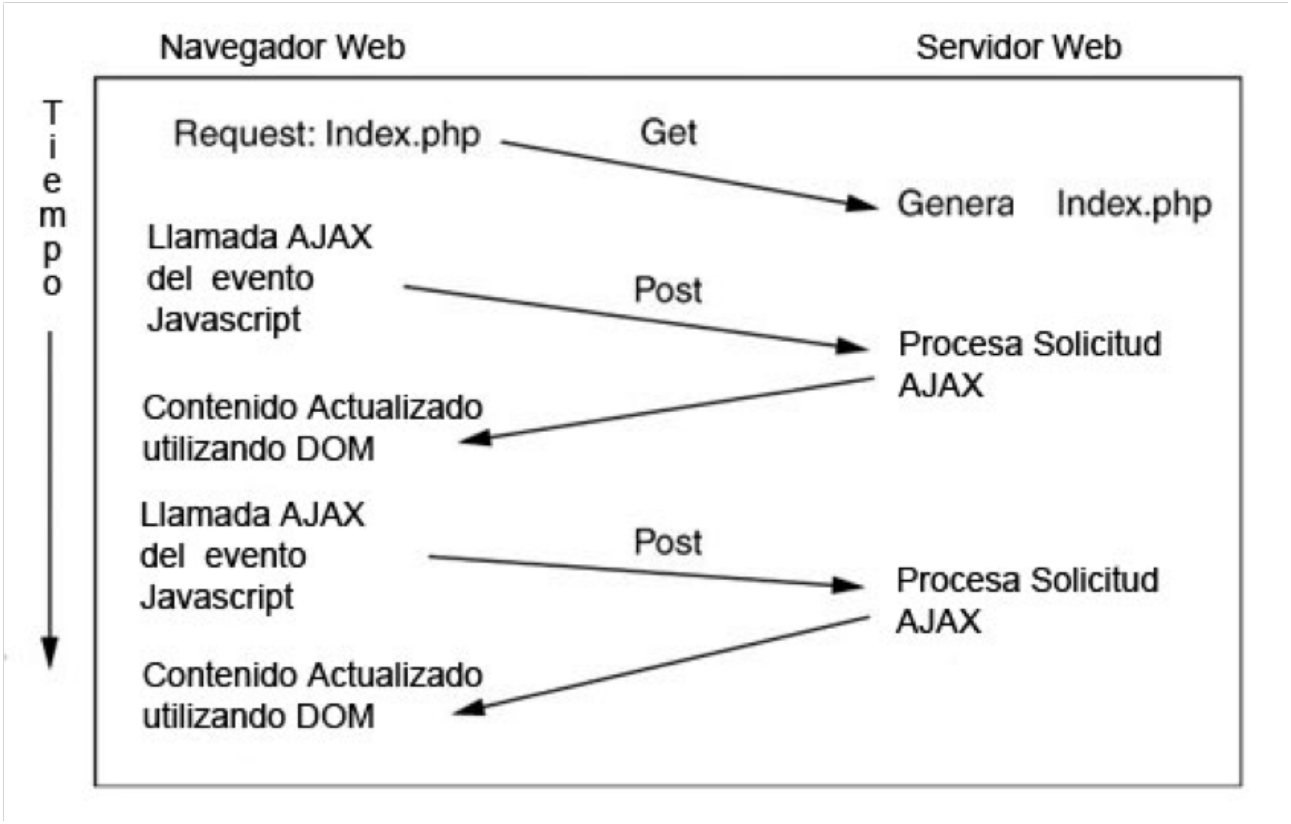


JSON y AJAX

AJAX es el acrónimo de (*Asynchonous JavaScript And XML*)

Permite actualizar partes de una página web de forma dinámica; es decir, modifica solo lo que ha cambiado.

Cuando hacemos una solicitud de datos almacenados en un fichero JSON externo, o cualquier otro fichero en este formato, hacemos una llamada mediante el objeto .



¿Por qué nos interesa JSON?

JSON resulta relevante por:

- Es la respuesta de datos que devuelven la mayoría de las APIs web
- Muchos portales de datos abiertos ofrecen la información en este formato
- Porque facilita la integración y visualización de información



Sintaxis de JSON

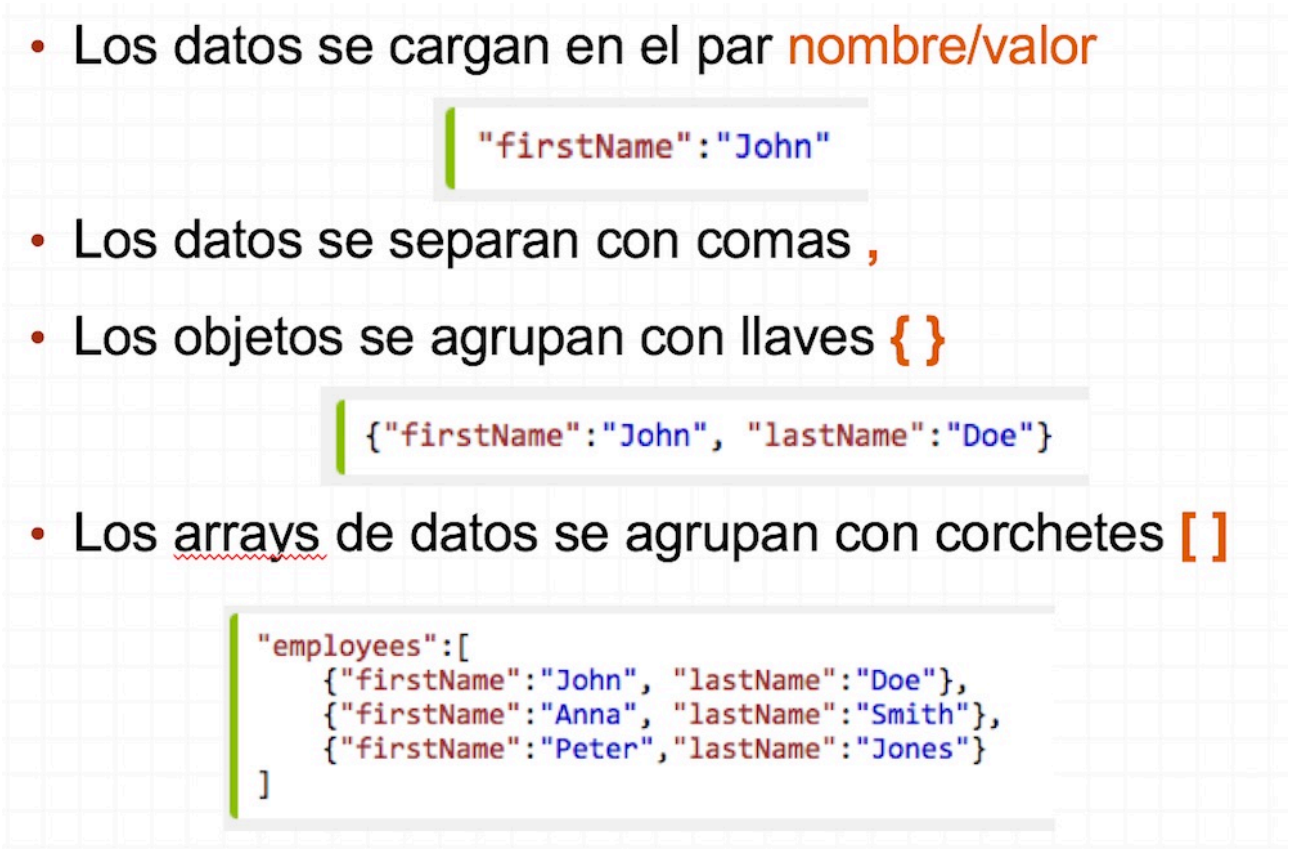
El formato JSON es idéntico al que se utiliza en JS para crear **objetos.**

Los datos son convertidos y tratados como objetos, por lo que:

- Se le aplican métodos, se crean variables, etc.
- Es más sencillo que XML ya que es **más corto** y **no requiere etiquetas de cierre**
- Puede usar arrays de datos.
- Puede ser “parseado” con una función JS estándar.

En este ejemplo vemos cómo creamos un objeto JSON en Javascript.

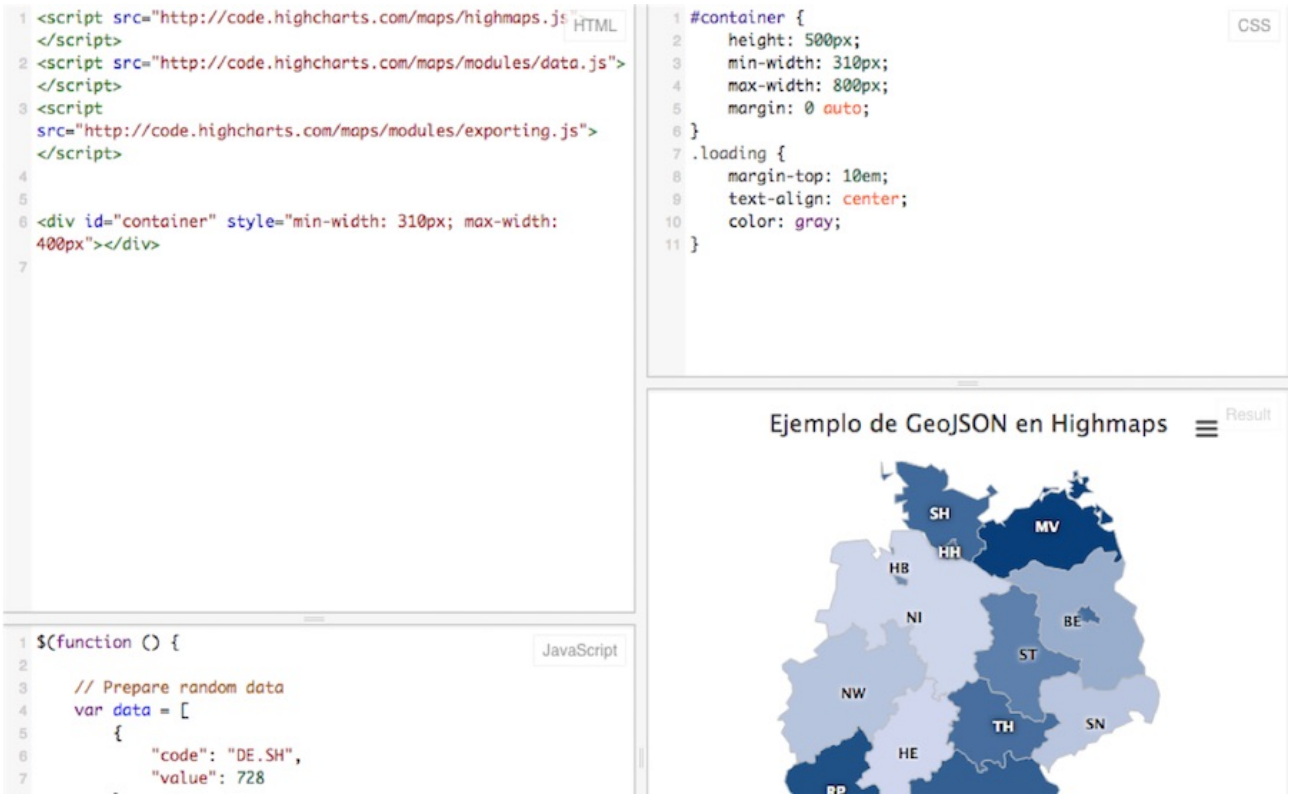
La principales reglas que debemos seguir en JSON son:



Los nombres en JSON requieren ir entre dobles comillas (a diferencia de JS)

Ejemplo de JSON en visualizaciones

En este ejemplo de visualización creada con HighChart podemos ver cómo se usan datos en formato JSON para cargarlos en un mapa.



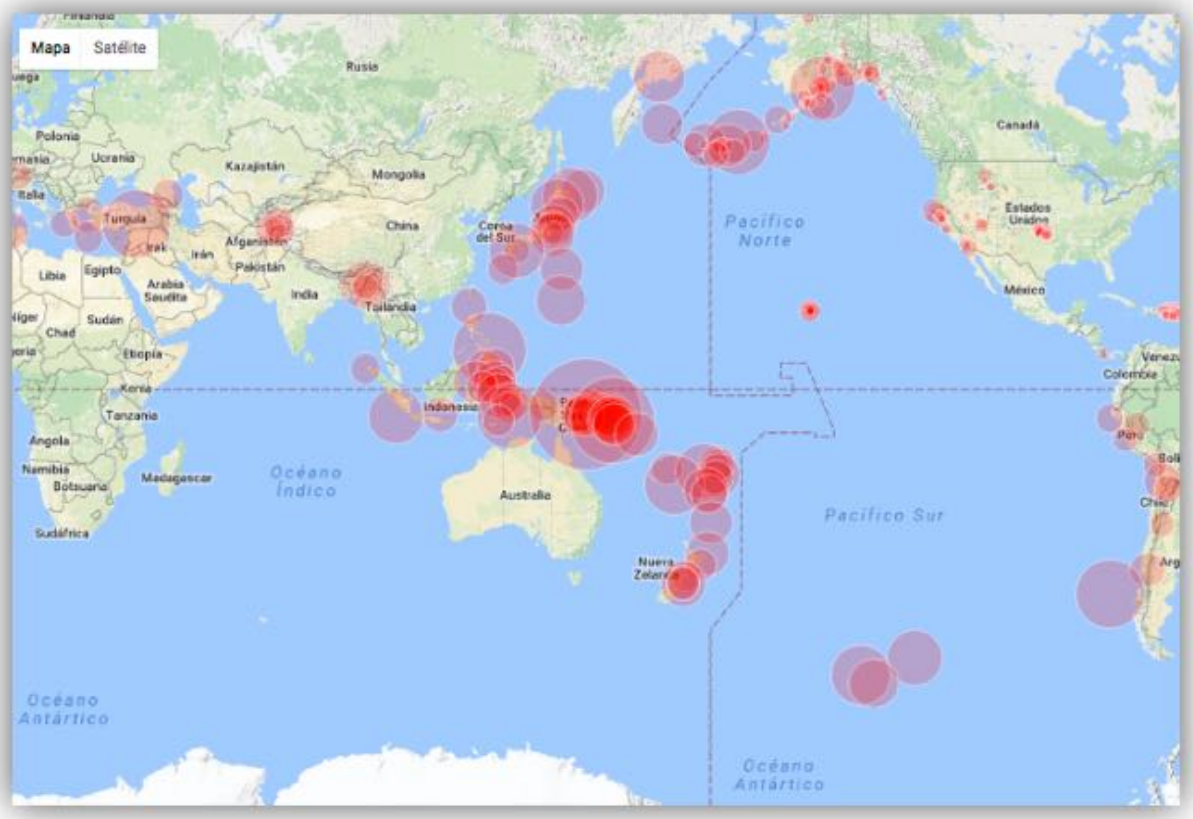
1. En primer lugar preparamos los datos con un array en formato JSON, mediante pares nombre/valor, que serán los que se carguen cuando el usuario explore el mapa
2. Luego jacemos la llamada al fichero json que contiene las localizaciones para que se cargue, y creamos una función
3. 3. Por último, iniciamos el mapa

GeoJSON

GeoJSON es un formato para el intercambio de **datos geoespaciales** *basado en JSON*.

Permite informar **diferentes tipos de geometrías**: puntos, multipuntos, líneas, multilíneas, polígonos, múltiples polígonos, y colecciones de geometrías

Uso de GeoJSON para cargar datos masivos



Vamos a trabajar con ejemplo de un mapa creado con la API de Google Maps. Vamos a **cargar ficheros GeoJSON** con miles de datos y customizar los resultados del mapa convirtiendo los marcadores en círculos.

Los datos los obtenemos, para este ejemplo, del sistema de feeds del USGS que ofrece datos en varios formatos, entre ellos GeoJSON, sobre diferentes sucesos naturales, en este caso, terremotos.

Como su sistema no permite directamente hacer llamadas de manera gratuita, tenemos que descargarnos el fichero que queramos y ubicarlo en servidor propio.

Una vez que tenemos el fichero en nuestro servidor, con una URL pública, podemos cargarlo y customizar el mapa obteniendo este resultado.

Nota: La función Math.pow() devuelve la base elevada al exponente para calcular el tamaño del círculo en función de la magnitud del terremoto.

Uso de JSON para personalizar mapas



Otra de las utilidades de JSON es la personalización de estilos. La API de Google Maps facilita un completo sistema de estilos para customizar el mapa a voluntad, algo que con otras soluciones de Google como Mymaps es más limitada.

Vamos a ver un ejemplo usando el elemento *StyledMapType*. En este, los estilos se aplican a través de:

- La modificación de las **funciones de mapas** (elementos geográficos incluidos)
- Los **parámetros de estilo** (color y visibilidad) de dichas funciones

Todos los estilos se agrupan en un ***array de estilos*** en formato JSON.

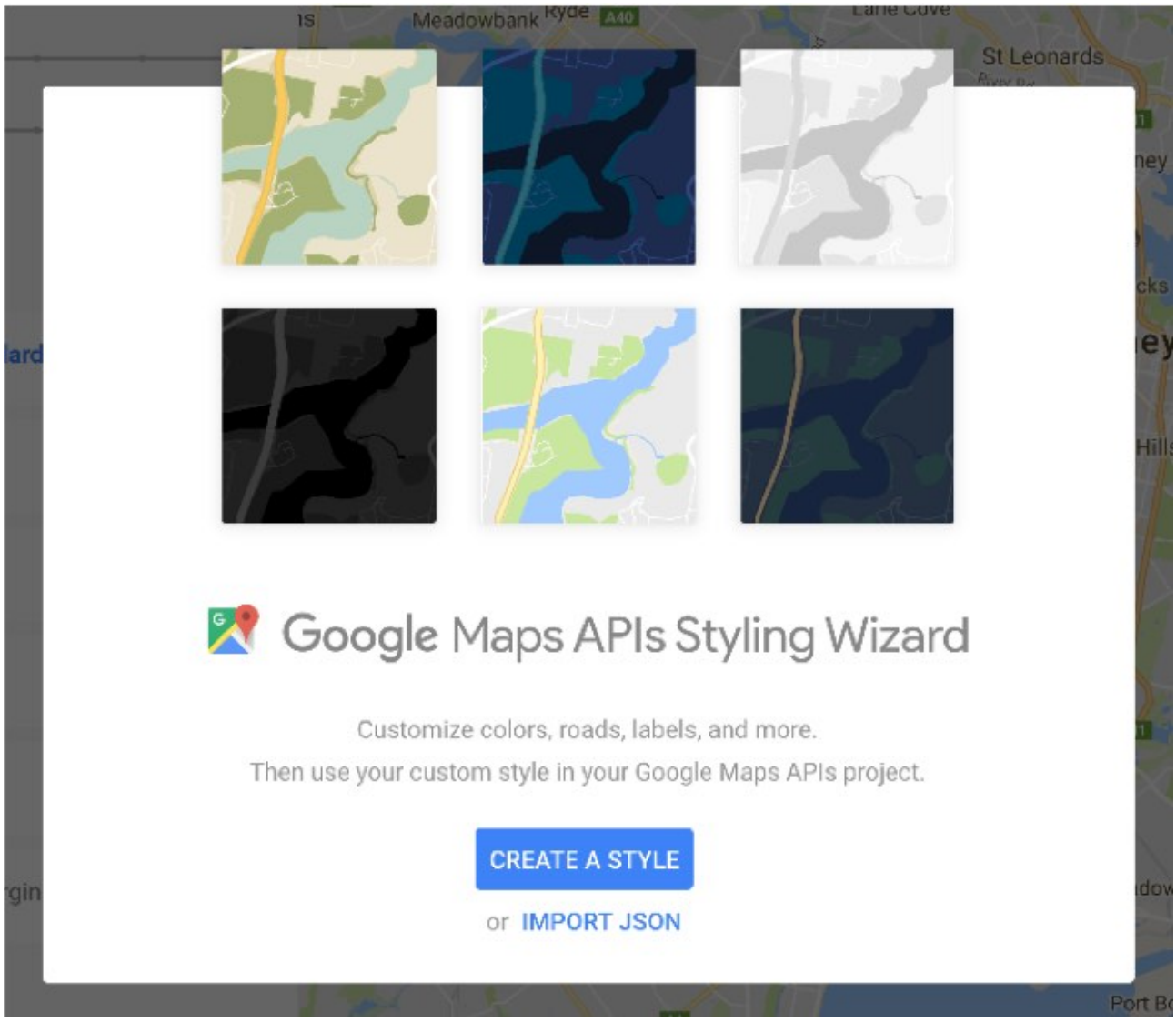
Veamos los pasos para crear este ejemplo:

1. Creamos el array con el nuevo estilo mediante el objeto google.maps.StyledMapType, y le pasamos a la función los valores cargados en el arreglo.
2. Le pasamos el nombre del estilo
3. Creamos una variable para asignar el estilo nuevo
4. Creamos el objeto mapa e incluimos el mapTypeId para añadir al control del mapa el nuevo tipo
5. Asociamos el estilo del mapa con el MapTypeId y lo presentamos para que se muestre

Asistente de mapa de Google Maps

En todo caso, configurar estilos complejos creando escribiendo directamente el

código de los estilos lleva tiempo. Así que muchas veces lo más eficaz es usar el **asistente de estilos de mapa** de la API de Google Maps.



Los pasos para hacerlo son:

1. Accedemos al asistente.
2. Configuramos el estilo bien con las features básicas, bien con las avanzadas que nos permiten modificar y acceder a cualquiera de las funciones y parámetros de estilo.
3. Finalizamos el estilo y copiamos el codigo JSON del array de estilos (será muy largo)
4. Vamos en el mapa al objeto google.maps.StyledMapType y copiamos el array con todo el contenido entre [ ] incluidos estos.

Ejercicio. Personalización del mapa

1. Retoma el código del ejemplo del Bucle for / in, de las aglomeraciones urbanas
2. Customiza el mapa mediante el editor de estilos de mapa
3. Copia el array de estilos en el editor dentro del objeto google.maps.StyledMapType [*Recuerda: como es un array JSON es todo el código entre [ ], incluidos estos*]

Solución

Grupo Ciberimaginario | Manuel Gertrudix - Alejandro Carbonell | 2025/2026 | 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

