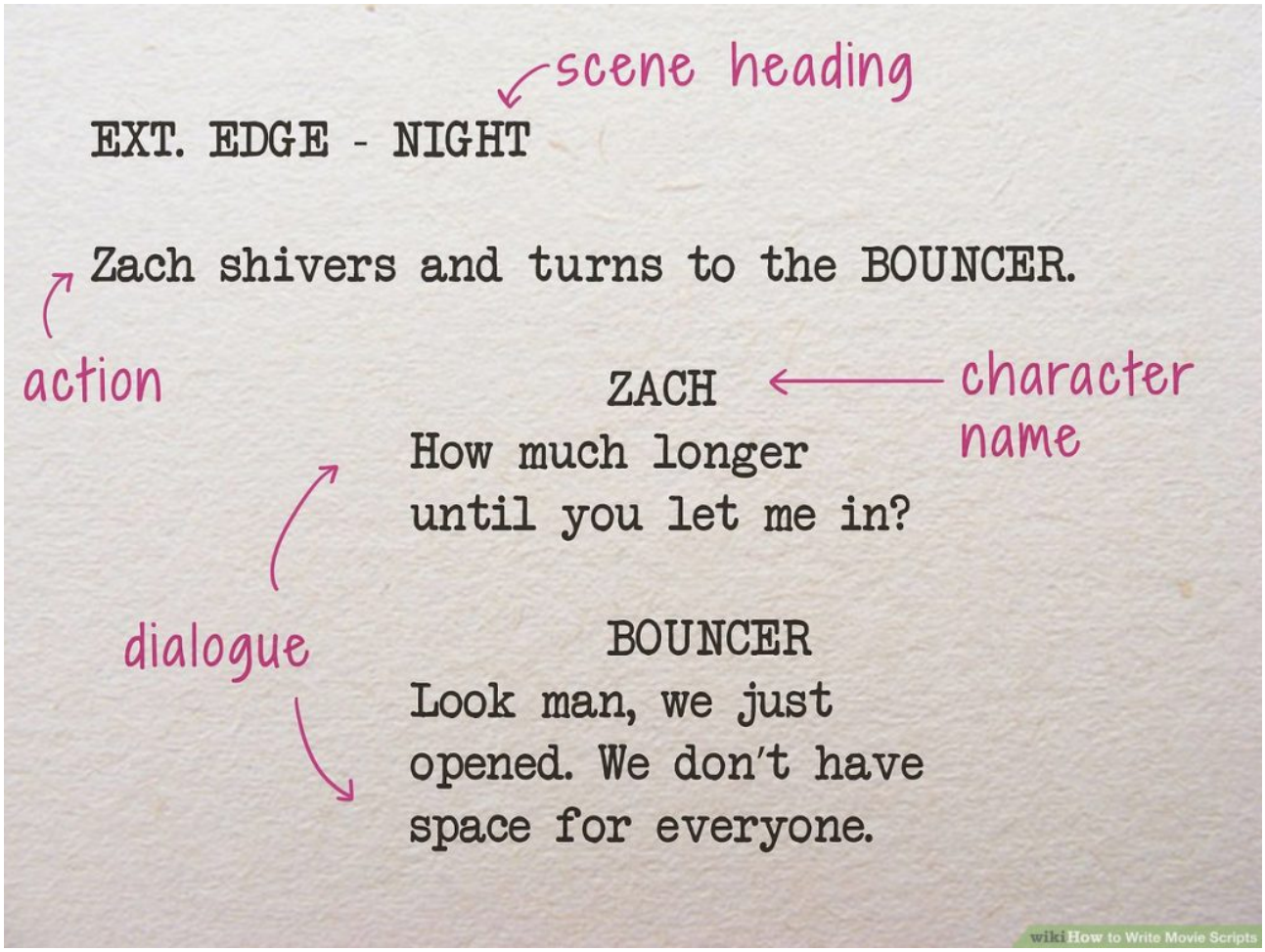
UTILIZACIÓN LENGUAJES SCRIPT – OLD

¿A qué nos referimos con un script?



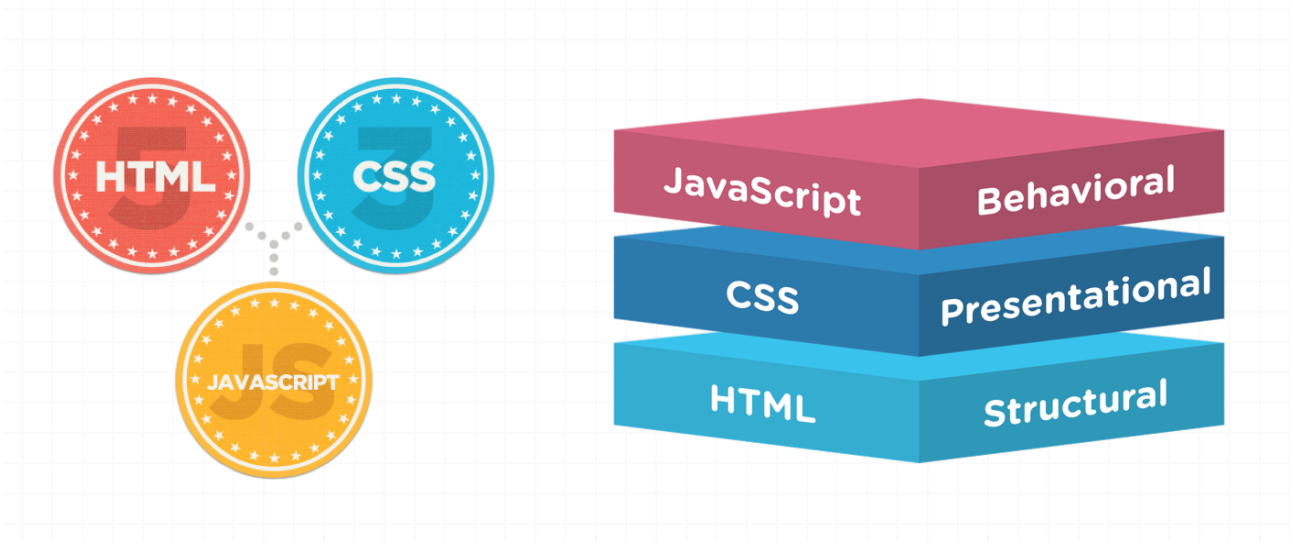
Cuando hablamos de un **script**, un guion o una secuencia de comandos es un pequeño (a veces no tanto) **programa** que sirve para «rutinizar» o automatizar ciertas tareas, habitualmente repetitivas, que se precisan para que funcione un proyecto más complejo.

En nuestro ámbito, nos interesan los scripts que permiten, como veremos, dotar de interactividad a visualizaciones de información o gestionar datos asociados a estas.

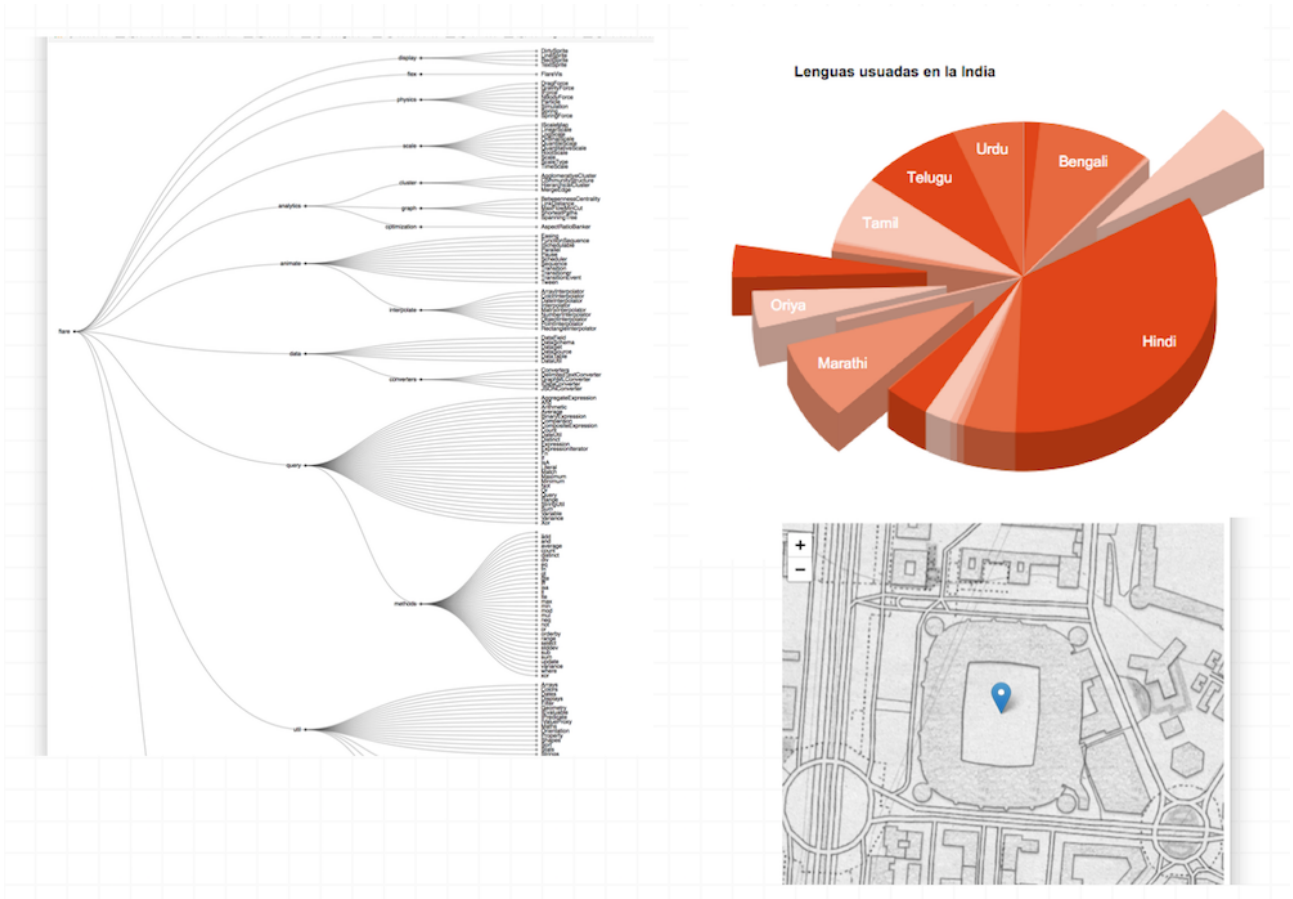
¿Con qué lenguajes programamos scripts?

Existen numerosos lenguajes de script, tanto de lado cliente como del lado servidor (C, C++, Python, Java, Go…)

En el **lado cliente**, Javascript es el lenguaje de script nativo de la web.



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



¿Para qué nos puede servir los lenguajes de script, y concretamente 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 adentrarnos en la creación de visualizaciones basadas en librerías JS

como jQuery
- Para hacer scraping de datos con Javascript (Jquery) o, como vais a ver posteriormente, con Phyton

En los ejemplos que vamos a ir viendo utilizaremos un editor de código online, JsFiddle.net. Podrás acceder a los ejemplos de forma directa a través de objetos embebidos como este, en el que podrás trabajar en las diferentes partes del código accediendo a las diferentes pestañas disponibles.

> **Las cuestiones que vamos a tratar en esta sesión de introducción son conceptos generales que, aunque se ejemplifican con Javascript, son generales que nos sirven para la mayoría de los lenguajes de script y nos introducen en la siguiente sesión en la que empezaréis a trabajar con Phyton.**

Nociones básicas de JS

¿Qué es?

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

Ejercicio

- *Abre el editor de código en JSFiddle con el ejercicio.*
- *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 revisando el código?*

¿Para qué sirve?

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 Twitter de un tema, o para hacer scraping de web utilizando soluciones como Node.js, Artoo.js o pjscrape)

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

Ejercicio

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 usando alguna herramienta como WebDevelopper Toolbar. Comprueba qué sucede. ¿Puedes navegar el mapa? ¿Compartir en redes sociales?

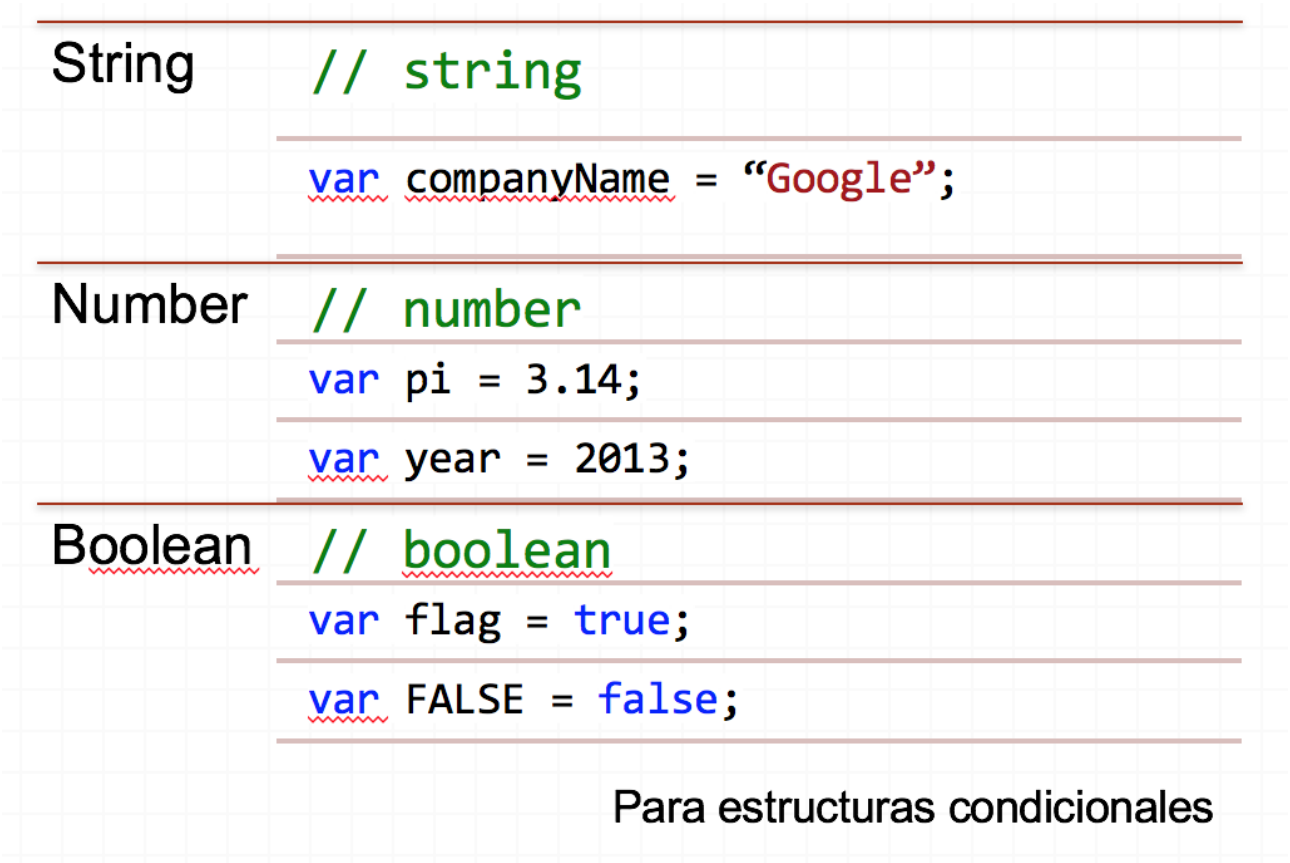
Veamos ahora un ejemplo del uso de javascript para **cambiar o modificar atributos** de html. Aquí las posibilidades son muy amplias, dado que podemos modificar los valores de cualquier atributo, desde el color de un texto, el tipo de fuente, etc.

En este ejemplo generamos un efecto de sustitución modificando el atributo src de una imagen.

Tipos de datos

Javascript puede almacenar los siguientes **tipos de datos**:

- string (cadenas)
- number (números)
- boolean (booleanos)
- function (funciones)
- array (arreglo)
- object (objetos)



Function

```
// function
var sayHello = function () {
alert('hello world!');
}
```

Array

```
// array
var numberArray = [1, 2, 3];

var animals = new Array["cat", "dog", "mouse",
"lion”];
```

Object

```
// object / json
var person = {
name: 'Barack Hussein Obama II',
age: '51',
title: '44th President of the United States’ }
```

Variables

¿Qué es una variable?

Una variable es un elemento que **contiene información** (datos) que puede ser ejecutada por el código.

Las variables sirven para **guardar** información que será utilizada posteriormente.

En este ejemplo x se define como una variable. Luego, asignamos a x el valor de 6:

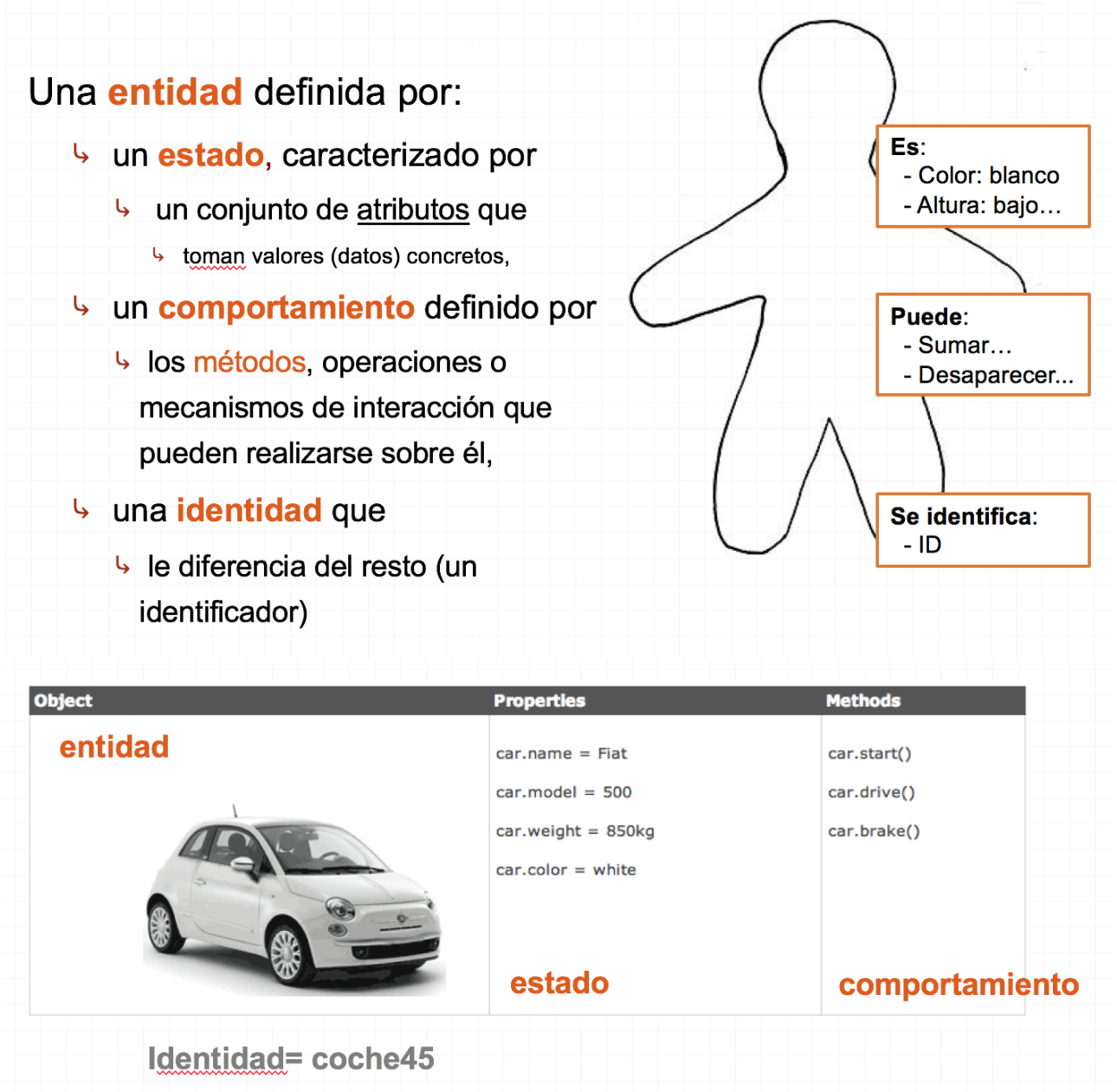
Conceptos de POO

La **POO (Programación Orientada a Objetos)** es un paradigma de programación que permite optimizar los procesos de programación y los resultados que se obtienen con ellos. Permite pre-diseñar objetos que son almacenados en librerías o bibliotecas para que puedan ser reutilizados por los programas sin necesidad de tener que volver a escribir las funciones necesarias cada vez que se quiere hacer uso de ellas.

Los elementos fundamentales de la POO incorporan un número amplio de componentes (clase, herencia, objeto, método..), pero nos centraremos solo en los que resultan fundamentales para manejarnos en el entorno de trabajo de un proyecto periodístico

- Objetos
- Eventos
- Funciones
- Métodos

Objetos



Eventos

Los eventos son **sucesos**, producidos normalmente por una acción del usuario, que **producen algún efecto**. Por ejemplo, cuando un usuario pulsa un botón o hace clic sobre un enlace.

Algunos de los eventos DOM más habituales son:

Evento	Descripción	Elementos para los que está definido
onclick	Pinchar y soltar el ratón	Todos los elementos
onload	La página se ha cargado completamente	<body>
onmouseout	El ratón "sale" del elemento (pasa por encima de otro elemento)	Todos los elementos
onmouseover	El ratón "entra" en el elemento (pasa por encima del elemento)	Todos los elementos
onsubmit	Enviar el formulario	<form>

Funciones

Todo evento lleva asociado, normalmente, una **función**.

Una función es un **bloque de código que puede ser ejecutado** cuando es llamada por un evento

Son muy útiles porque es muy habitual reutilizar código con diferentes argumentos a lo largo de un programa, por lo que podemos utilizar la función sin tener que escribir de nuevo el código, sólo **invocándola.**

Para **llamar a una función** hay que invocarla en cualquier parte de la página web.

Cuando se invoca una función, todo el código que contenga entre **llaves {}** se ejecuta

Para invocarla, basta con escribir su nombre seguido de paréntesis

Ejemplo con Google Maps. En este caso, se utiliza la API de Google Maps para crear un mapa que agrupa (cluster) marcadores en función del nivel de zoom.



Nota: Ver documentación

Librerías y APIs

Estos dos conceptos están directamente vinculados.

Técnicamente una **librería o biblioteca** (library) es una colección de implementaciones de comportamiento, pero lo entenderemos mejor si decimos que es una larga lista de código de programación que incluye un amplio número de funciones que podemos utilizar desde nuestro programa con un esfuerzo limitado.

Por su parte, una **API** (***Aplication Programming Interface***) es una especificación formal que permite comunicar componentes de software de dos sistemas distintos. Para entendernos, es una especie de "subcontratación" de funciones.

Una API es el libro de instrucciones que nos permite conocer cómo trabajar con el **conjunto de funciones**, procedimientos y objetos contenidos en la librería o biblioteca con la que vamos a trabajar.

Como señala de forma gráfica Diego Ceballos, en un ejemplo cotidiano, una cafetera Nespresso sería una librería para hacer café de forma rápida, y su libro de instrucciones sería la API.

API Keys

Para usar una API normalmente se requiere una API Key de autenticación para conectarse con el servicio. Ello permite establecer los límites de cuota por Key y no por IP. Además, esto permite comunicar al servicio con la aplicación solicitante.

> *Por ejemplo, en Google Maps, si el número de llamadas que van a hacer nuestros mapas es muy alto, debido al tráfico de la página, debe utilizarse una Google API Key*

```
<script src="http://maps.googleapis.com/maps/api/js?key=YOUR_KEY"></script>
```

Tipos de Librerías y APIs

Podemos clasificar las APIs en base a la funcionalidad que aportan. Siguiendo este criterio podemos considerar, desde el punto de vista de la utilidad en proyectos periodísticos, las siguientes:

- Para crear mapas
- Parar incorporar animaciones
- Para desarrollar aplicaciones
- Para acceder y manipular los elementos del DOM de forma sencilla
- Para trabajar con imágenes y gráficos
- Para visualizar datos

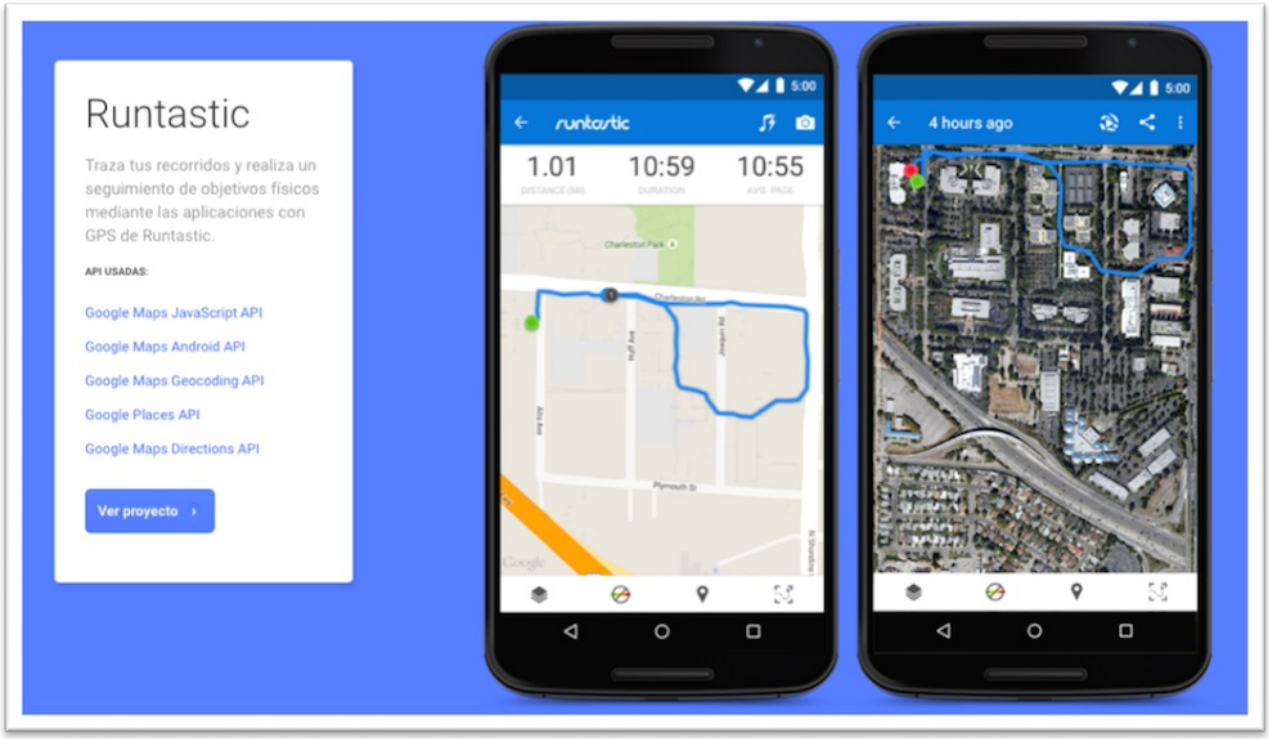


En este enlace puede verse una amplia comparativa de librerías.

Google Maps API

Google Maps API

Vamos a adentrarnos en el uso de funciones. Para ello, vamos a trabajar con la API de Google Maps que es una amplia librería dirigida a la creación de complejas aplicaciones de mapas.



Crear un mapa simple incluyendo un marcador

La elaboración de este mapa nos permitirá comprender mejor cómo operan las funciones.

Vamos a elaborar un **mapa sencillo en el que incluiremos un marcador**:

1. **Creamos el código html**. Fíjate que en el head se hace la llamada a la API. El parámetro callback ejecuta la función initMap cuando se ha cargado la API. Los atributos async y defer permiten que se siga renderizando la página mientras se carga la API. La key hace la llamada mediante la API Key para acceder al servicio.
2. **Dibujamos el mapa y el marcador en el fichero JS**. Creamos una función que inicializa y añade el mapa, con los siguientes elementos: a) Una variable que inicializamos con las coordenadas que centrarán el mapa. b) Una variable que crea el objeto mapa, utiliza el método getElemenById y le pasa dos propiedades: center y zoom (escala: cuanto más bajo, más general). Creamos una variable para incluir el marcador, inicializándola con el objeto marcador, y dos propiedades: position y map

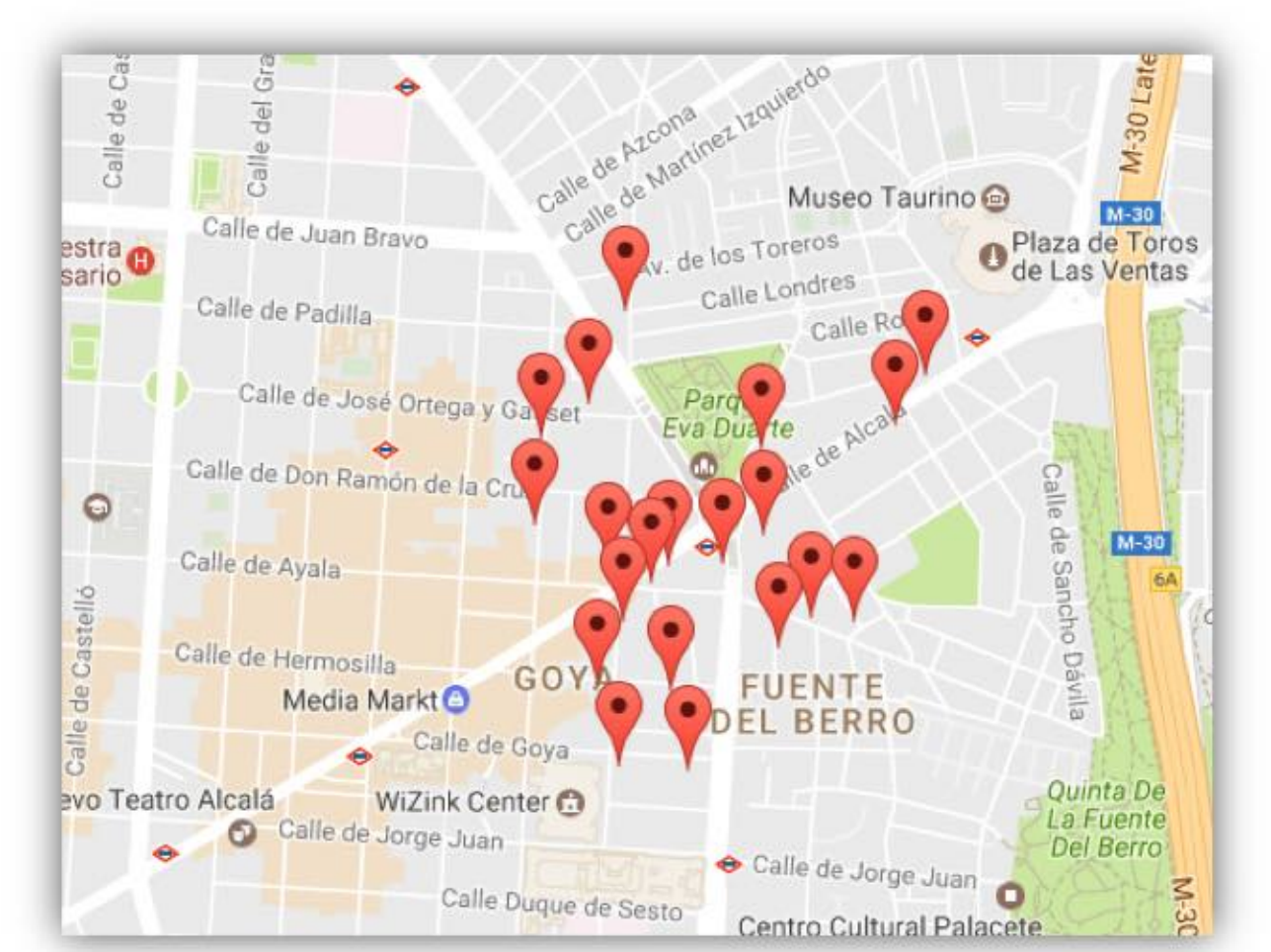
En este enlace puedes ver el detalle explicado de todo el procedimiento.

Funciones avanzadas

Ya hemos visto muchas de las posibilidades que ofrece Google Maps API.

En este apartado, veremos algunas características que nos pueden resultar útiles para la configuración de nuestros mapas.

Query Places



Query Places nos permite hacer una búsqueda automatizada de lugares.

Con la librería places podemos ubicar de forma automática lugares en un mapa.

A través de los parámetros de búsqueda del método *nearbySearch* podemos modificar la *query* que hace sobre los datos de places.

Veamos un ejemplo:

- Se declaran las variables map e infowindow
- Se inicializa el mapa mediante la función
- Se define el valor del infowindow a través de la clase google.maps.InfoWindow
- Se carga la biblioteca de sitios "places", para el subconjunto PlacesService
- Con el método nearbySearch se localizan aquellos servicios que cumplan los requisitos indicados en los parámetros de la búsqueda

Gmaps.js

Si trabajar directamente con la API de Google Maps te resulta complejo, Gmaps.js es una librería desarrollada para simplificar el trabajo con la API.

JSON y GeoJSON

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.



¿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



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

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.



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

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:

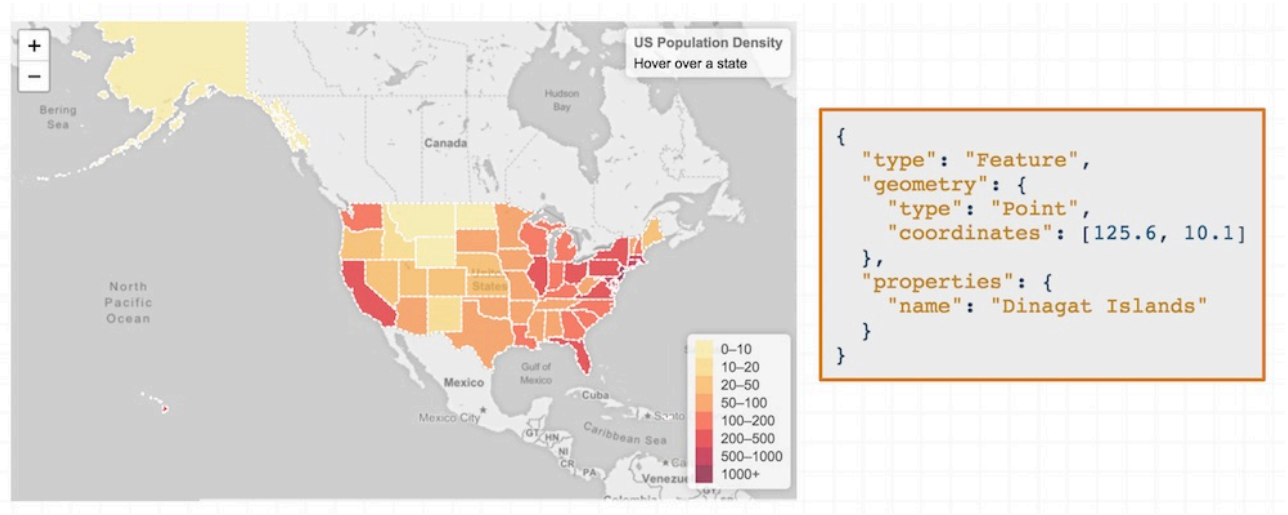
1. Accedemos al asistente.
2. Configuramos el estilo bien con las features básicas, bien con las avanzadas que nos permiten modificar y acceder a cualquiera de las funciones y parámetros de estilo.
3. Finalizamos el estilo y copiamos el codigo JSON del array de estilos (será muy largo)
4. Vamos en el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

LeafLeft

Leaflet es una librería especializada para la creación de **mapas interactivos.**

Tiene una amplia documentación y una galería de estilos de mapa (basados en Mapbox) para comenzar a trabajar rápidamente con ella.

Como toda librería basada en JS, utiliza la base de este, pero simplifica la sintaxis para hacer más sencillo y ágil el desarrollo de los mapas.



Vamos a crear un mapa básico con LeafleftJs:

1. En el head de html, enlazamos la hoja de estilo de Leaftletjs y, detrás, ponemos después el enlace al js de Leaftletjs.
2. En css Le damos un alto al div que contiene el mapa a través del id
3. En Js creamos el mapa y le asignamos las coordenadas decimales y el nivel de zoom
4. Cargamos una capa de mapa (tile layer) desde Mapbox. Necesitaremos incluir el accessToken de mapbox. Si no tenemos uno, tendremos que crearlo (En id podemos incluir cualquiera de los estilos de mapa base disponibles en Mapxbox)
5. Creamos un marcador
6. Creamos un círculo
7. Creamos dos popup sobre el marcador y el círculo
8. Creamos una función

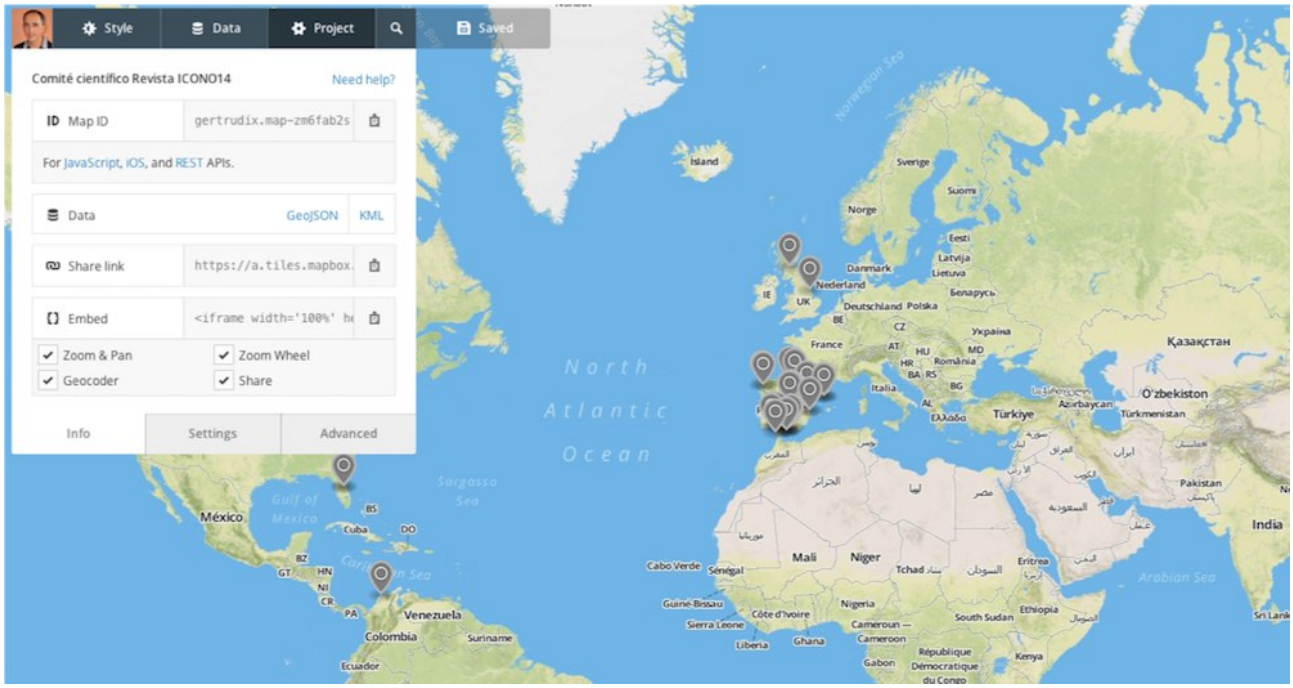
En este otro mapa, sobre la base del anterior hemos incorporado un marcador personalizado usando las opciones que LeafLeft tiene para ello.

Mapbox

Mapbox es otra estupenda librería para crear mapas. De hecho, tiene su propio sistema de mapas que es usado por otras librerías como base, tal como sucede como LeafLeft.

Ofrece una documentación muy amplia lo que facilita trabajar con ella.

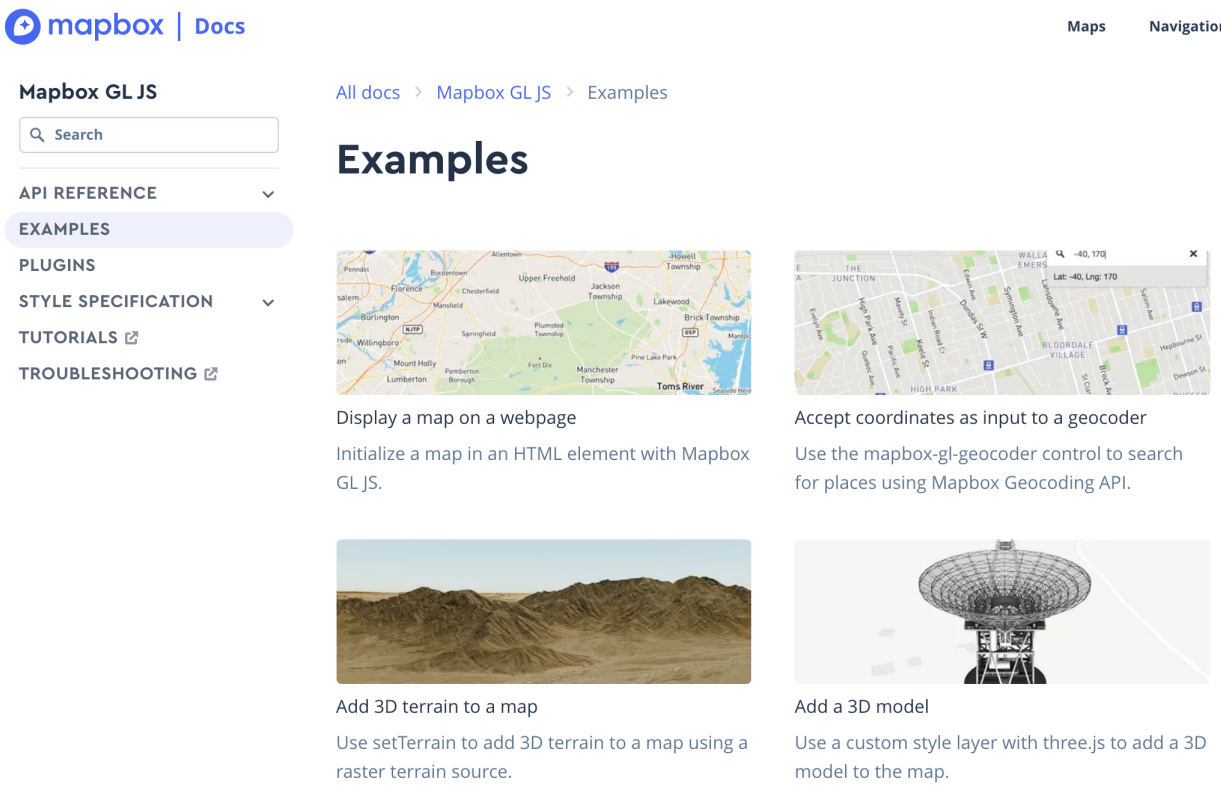
De manera similar a lo que sucede con Google Maps, existe la posibilidad de trabajar directamente con la API o hacerlo mediante el editor de mapas (studio). Esto último simplifica aún más el desarrollo dado que, además, luego podremos customizar o personalizar elementos accediendo al código resultante.



Una de las aportaciones interesantes de esta librería es que dispone de numerosos efectos que pueden darle un aspecto diferente a nuestras visualizaciones en mapas. La documentación de la API incluye muchas, pero destacaremos aquí algunas de ellas:

Ejemplos de mapas que podemos crear

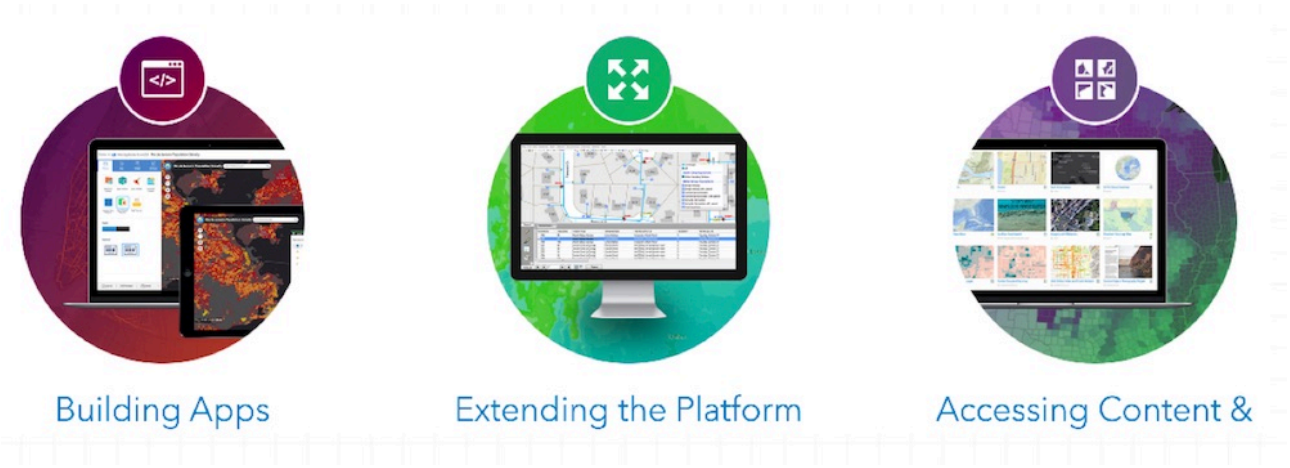
En este apartado de la documentación podemos revisar todas las posibilidades que ofrece esta librería.



ArcGIS online

ArcGIS online es una completa suite de aplicaciones y APIs. Dispone de un Gestor web y de numerosas APIs para interactuar y adaptar los resultados.

En este enlace puedes acceder a un tutorial básico para empezar a crear mapas 2D.



Un aspecto diferencial de esta solución, es que ofrece un editor de **escenas**. Esto facilita un modo de crear escenas 3D sobre la representación de la esfera terrestre, como en este ejemplo:

O sobre zonas concretas en las que pueden incluirse capas como edificios, etc:

Google Chart

Google Chart es una librería orientada a la creación de una amplia variedad de gráficos dinámicos.



Creación de Chart básicos

```
// Create the data table.
var data = new google.visualization.DataTable();
data.addColumn('string', 'Topping');
data.addColumn('number', 'Slices');
data.addRows([
  ['Mushrooms', 3],
  ['Onions', 1],
  ['Olives', 1],
  ['Zucchini', 1],
  ['Pepperoni', 2]
]);
```

type: string label: Topping	type: number label: Slices
Mushrooms	3
Onions	1
Olives	1
Zucchini	1
Pepperoni	2

La creación de Chart requiere que los datos sean incluidos mediante una clase de JavaScript: *google.visualization.DataTable*

La clase está definida en la librería de visualización de Google.

La tabla de datos corresponde a una tabla similar a esta.

Veamos los pasos para crear nuestro primer ejemplo:

1. Cargamos la API de visualización del paquete que nos interesa, en este caso “corechart”
2. Establecemos el callback para que se ejecute la función cuando se haya cargado la API de visualización de Google y no antes
3. La devolución de llamada que crea y rellena una tabla de datos, crea una instancia del gráfico circular, pasa los datos y los dibuja.
4. Creamos la tabla de datos
5. Configuramos las opciones del chart
6. Instanciamos y dibujamos el chart pasándole la configuración de las opciones

Personalización del Chart

Para cada Chart podemos personalizar diferentes elementos como:

- Título, Color, grosor de línea, relleno de fondo, etc.
- Incluir elementos: títulos de los ejes, etc.

Las opciones se presentan como pares ***name.value***

```
var options = {
  width: 400,
  height: 240,
  title: 'Toppings I Like On My Pizza',
  colors: ['#e0440e', '#e6693e', '#ec8f6e', '#f3b49f', '#f6c7b6']
};

chart.draw(data, options);
```

- Las opciones pasan los valores al chart mediante el método *draw()*
- Cada chart posee los pares adecuados para la customización de ese tipo de visualización

En este ejemplo vemos un Pie Chart en el que se han establecido las siguientes opciones: Ancho y alto •Título •Colores con un array de hexadecimales •is3D, para dar el aspecto 3D

Las opciones de customización son muy variadas, lo que permite, y esta es su principal ventaja, adaptar los gráficos al estilo visual de nuestro proyecto.

En este se modifican los atributos color, pieHole

Y en este otro: slices, pieHole, is3D…

Otra de las opciones es incluir HTML en los tootilps:

O incluir otros Charts dentro de los tooltips:

Combo Charts

Podemos también crear charts agrupados, lo que se conoce como Combo Charts. En este caso, vemos dos ejemplos: en el primero los resultados de dos procesos electorales sin customizar, y el segundo, en el que e incluyen los resultados de unas terceras elecciones, y se customizan algunos elementos.

Google Public Data Explorer

Aunque no pertenece a la API de Google Chart, Google Public Data Explorer es una interesante herramienta de Google que permite elaborar gráficos sencillos de forma rápida, utilizando los datos públicos abiertos ya cargados por las principales entidades internacionales (Eurostat, US. Census Bureau, Data.gob.uk, Banco Mundial…) o nacionales (INE…)

Mediante un sistema de filtros se pueden crear visualizaciones tanto para realizar análisis como para crear gráficos que se vayan a insertar posteriormente en nuestro reportaje o web.

En este ejemplo, se usan datos del Banco Mundial para comparar la evolución de la ratio de niños y niñas fuera de la escuela por regiones mundiales:

[googlemaps https://www.google.com/publicdata/embed?ds=d5bncppjof8f9_&ctype=l&strail=false&bcs=d&nselm=h&met_y=children_out_of_school&fdim_y=education_level:2&scale_y=lin&ind_y=false&rdim=region&idim=region:EAS:ECS:LCN:MEA:SSF:SAS:NAC&ifdim=region&hl=en_US&dl=en_US&ind=false&w=100%&h=450]

Librerías DataViz (Datos)

Existen muchas alternativas actualmente para la visualización de datos. Muchas de las librerías tienen su propia sintaxis, basada o derivada de JS. La ventaja de disponer de esta variedad es que podemos tener un catálogo muy amplio de soluciones que permitan elegir la mejor opción para cada caso e ir introduciendo cierta variedad en nuestros proyectos.

Revisamos, a continuación, algunas de las soluciones disponibles. Aunque no se entra en su detalle, permite hacerse una idea de las numerosas posibilidades existentes para continuar explorando por nuestra cuenta.

D3.js

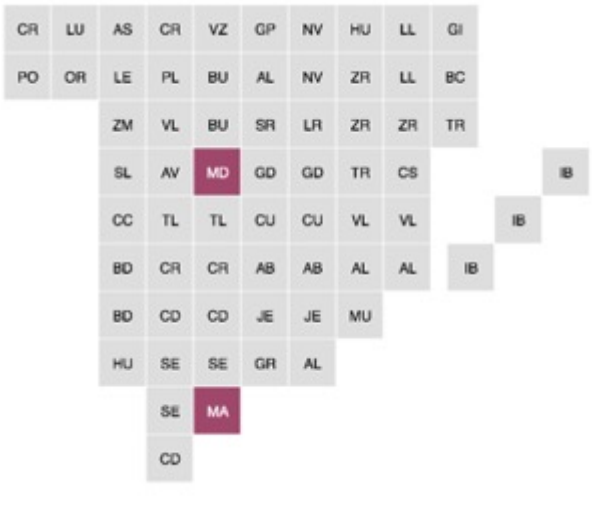
D3.js es una librería JS para manipular documentos basados en datos. Se utiliza para realizar visualizaciones complejas. Cuenta con una amplia galería de visualizaciones.

Algunos ejemplos interesantes aplicados de esta librería:

- China manufacture
- Similar song Networks

En este enlace puedes seguir un tutorial de esta librería D3.js

Funcionalidad



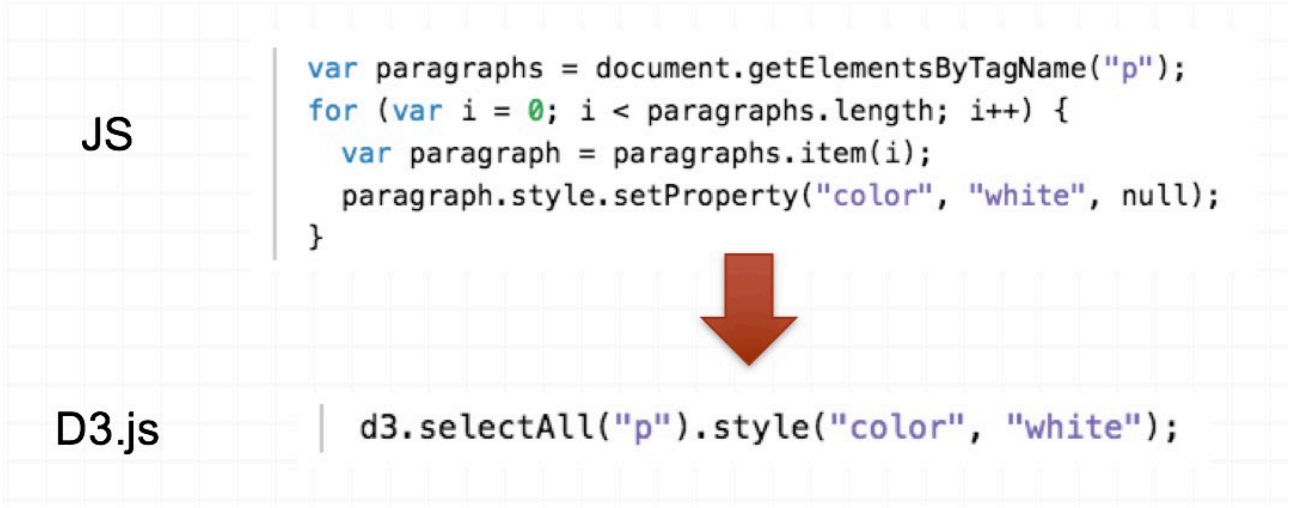
Permite obtener datos de cualquier elemento del DOM y aplicarle transformaciones en el documento.

Sobre un mismo conjunto de datos permite realizar varias transformaciones. Por ejemplo, sobre un array podemos:

- Crear una tabla
- Generar un gráfico interactivo en SVG

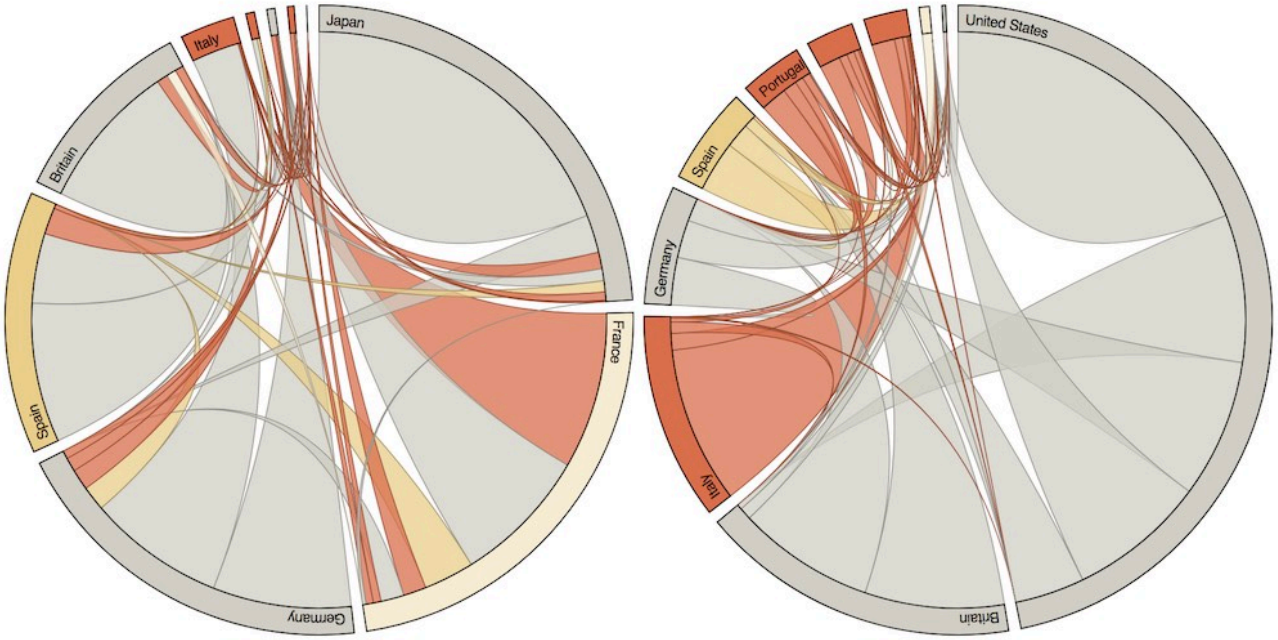
Y de una manera muy flexible y rápida.

Sintaxis



Ejemplos

En este primer ejemplo vemos un Diagrama de acorde, documentado en DJ3 (podemos reutilizar el código). En este caso, hemos reutizado el código y lo hemos adaptado para crear esta versión sobre el flujo de deuda entre países:



El funcionamiento de las visualizaciones básicas es relativamente sencillo. Los datos se cargan en ficheros .csv que se llaman desde el html para cargarlos en la visualización.

ZinkChart

ZinkChart es otra librería interesante. Orientada a gráficos habituales (líneas, histogramas, sectores, etc.) ofrece un muy buen resultado visual, incluyendo animación de los elementos y la incorporación de elementos gráficos añadidos que dan un aspecto muy completo.

Ejemplo: Barras

Ejemplo. Líneas

AmCharts

AmCharts es otra buena librería para crear gráficos y mapas. Puede descargarse, enlazarse o trabajar con el editor online live AmCharts.



Ejemplo de mapa

Ejemplo de gráfico

Highcharts

Highcharts es una buena librería de gráficos interactivos que cuenta también con una versión de creación online. Está muy orientada a gráficos de línea, columnas, barras para información económica.

En el editor online permite la carga de los datos en csv, lo que facilita la gestión de estos para la preparación del gráfico.

AnyCharts

AnyChart.JS es una completa librería para el desarrollo de:

- Charts
- Stocks
- Maps
- Gantt

Permite hacer algunas representaciones bastante singulares y diferentes, por lo que es una buena opción cuando se busca salirse de lo habitual.

Este gráfico es un buen ejemplo:

También resulta útil para generar **paneles de gráficos, en formato dashboard.**

Otras librerías

Chart.js

Chart.js pertenece al grupo de librerías ligeras. No permite hacer muchas cosas,

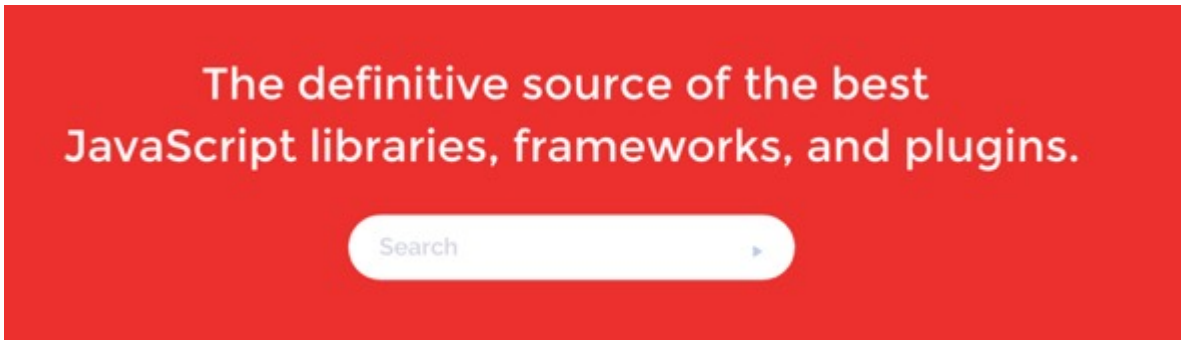
pero las visualizaciones de gráficas que hace son muy limpias y ligeras.

Sigma.js

Sigma.JS es una librería para crear dibujo gráfico, y está especializado en creación de gráficos de redes.

Listados de librerías

Javascripting es un directorio de librerías que colecta cientos de soluciones para problemas muy diversos basados en javascript.



Librerías Mapas

Cesium.js

CesiumJS es una potente biblioteca de JavaScript de código abierto para crear mapas y globlos terráqueos 3D de nivel internacional. Permite crear mapas tanto estáticos como dinámicos, y con la posibilidad de incluir líneas temporales que muestren la evolución de un fenómeno.

También permite levantar edificios sobre superficies.

New York 3D TilesSee over 1.1 million OpenStreetMap buildings in New York City.

Cesium ion!

Cesium también dispone de una versión de escritorio que incluye algunas de las principales funciones de la librería.

Dispone de un completo tutorial para conocer cómo trabajar con ella.

OpenLayers

OpenLayers facilita la creación de mapas dinámicos en cualquier página web. Puede mostrar mosaicos de mapas, datos vectoriales y marcadores cargados desde cualquier fuente. OpenLayers ha sido desarrollado para promover el uso de información geográfica de todo tipo.

Dispone, también, de un tutorial, y una librería de ejemplos.

Utilidades

Datos cartográficos

Global Administrative Areas contiene un amplio bancos de ficheros de shapefiles: KML, vectoriales… gratuitos de la mayoría de los países del mundo, hasta cuatro niveles administrativos: país, región, provincia y localidad.

Librerías pictográficas

Las librerías pictográficas web son colecciones de iconos que pueden cargarse vía web en nuestro proyecto mediante un enlace. •Las tres principales son:

- Font Awesome Icons. Es una colección libre de más de 600 iconos. Permite control CSS, sin manejo de JS.
- Bootstrap Icons
- Google Icons

Tutoriales

- Catálogo de visualización de datos. Extraordinario tutorial visual sobre las mejores formas para visualizar los datos.
- The Graphic continuum.

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