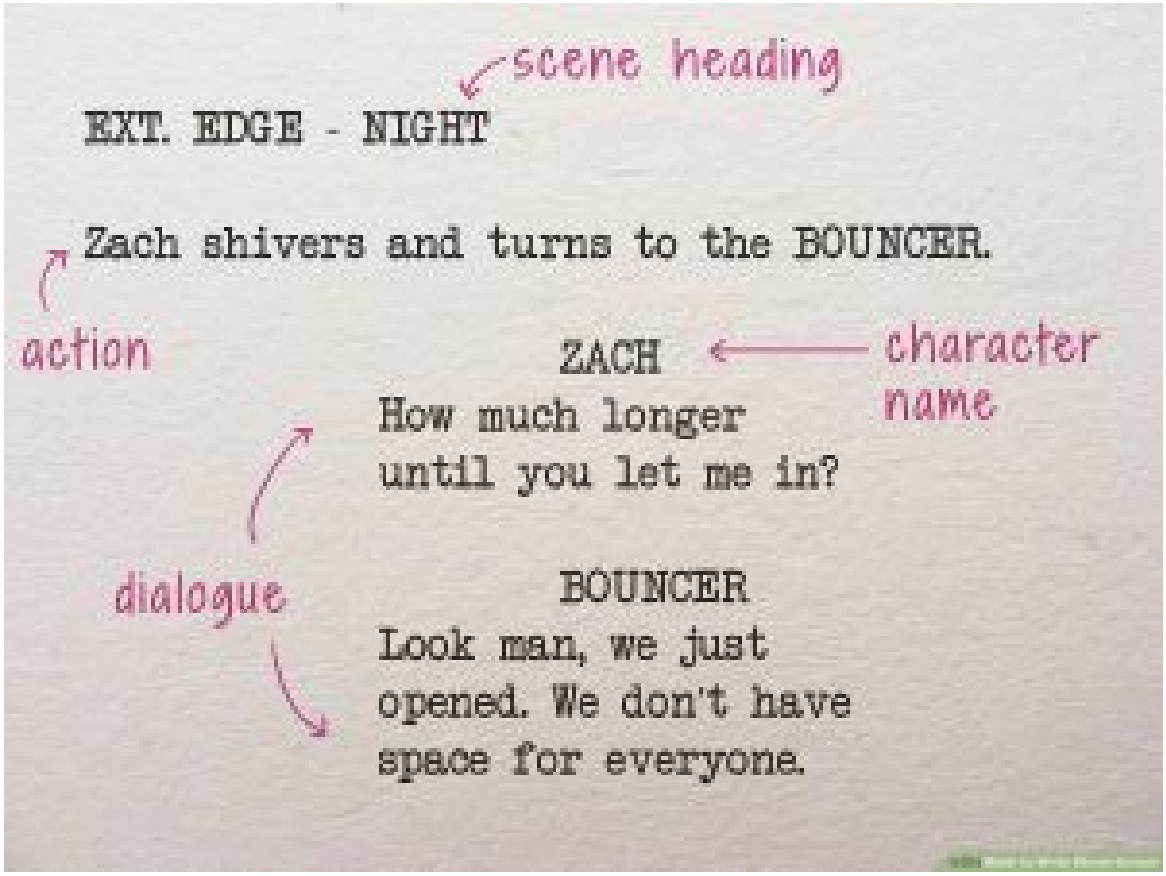
UTILIZACIÓN LENGUAJES SCRIPT – MIP

Conociendo JavaScript | Google Maps | Otras librerías de mapas | DataViz | Datos3D | Profundizar

Conociendo JavaScript

¿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.

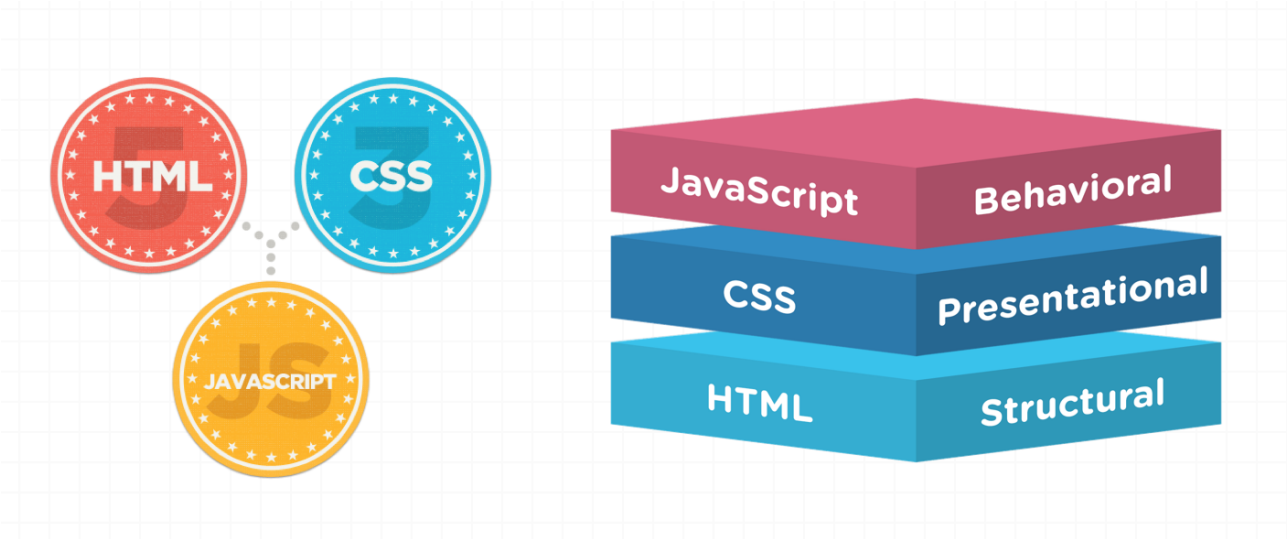
JAVASCRIPT EN ACCIÓN

Visualización de datos. Ejemplo de diagrama Sankey

Visualización de secuencias sunburts

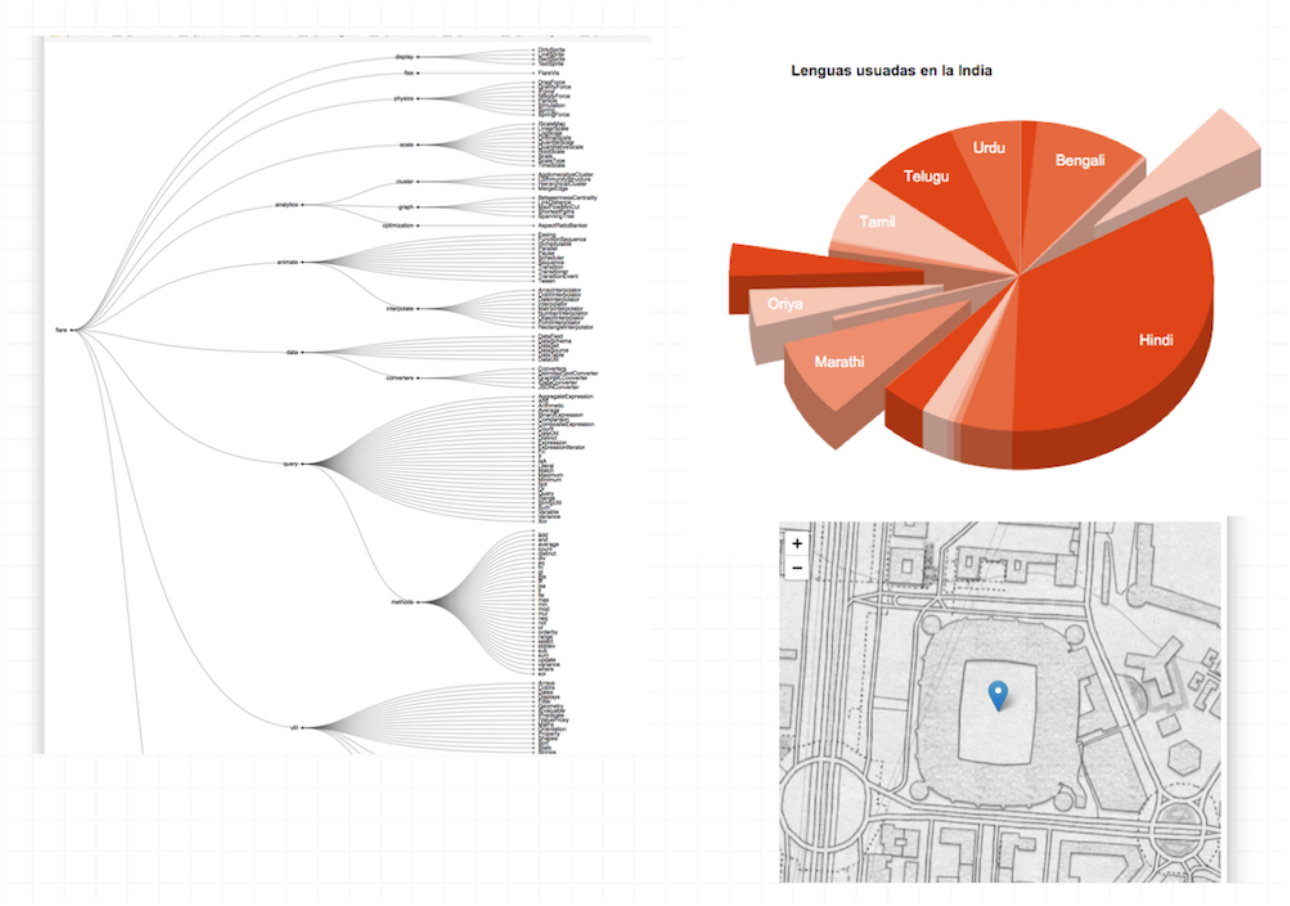
PRESENTANDO A JAVASCRIPT

JavaScript es el lenguaje de script nativo de la web.




```
<script>

    var grid_length = 75;
    // Note that grid_length in the book is 100.
    // I reduced it here for smooth performance on mobile devices.
    var grid = [];
    var temp_grid = [];
    var beta = 0.05;
    var gamma = 0.15;

    function get_random_int(min, max) {
        return Math.floor(Math.random() * (max - min + 1)) + min;
    }

    function init_grid() {
        for (var i = 0; i < grid_length; i = i + 1) {
            grid[i] = [];
            for (var ii = 0; ii < grid_length; ii = ii + 1) {
               grid[i][ii] = "S";
            }
        }
        grid[get_random_int(0,grid_length-1)][get_random_int(0,grid_length-1)] = "I";
    }

    init_grid();

    draw_grid(grid,["S","#dcdcdc","I","#c82605","R","#6fc041"]);
```

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.



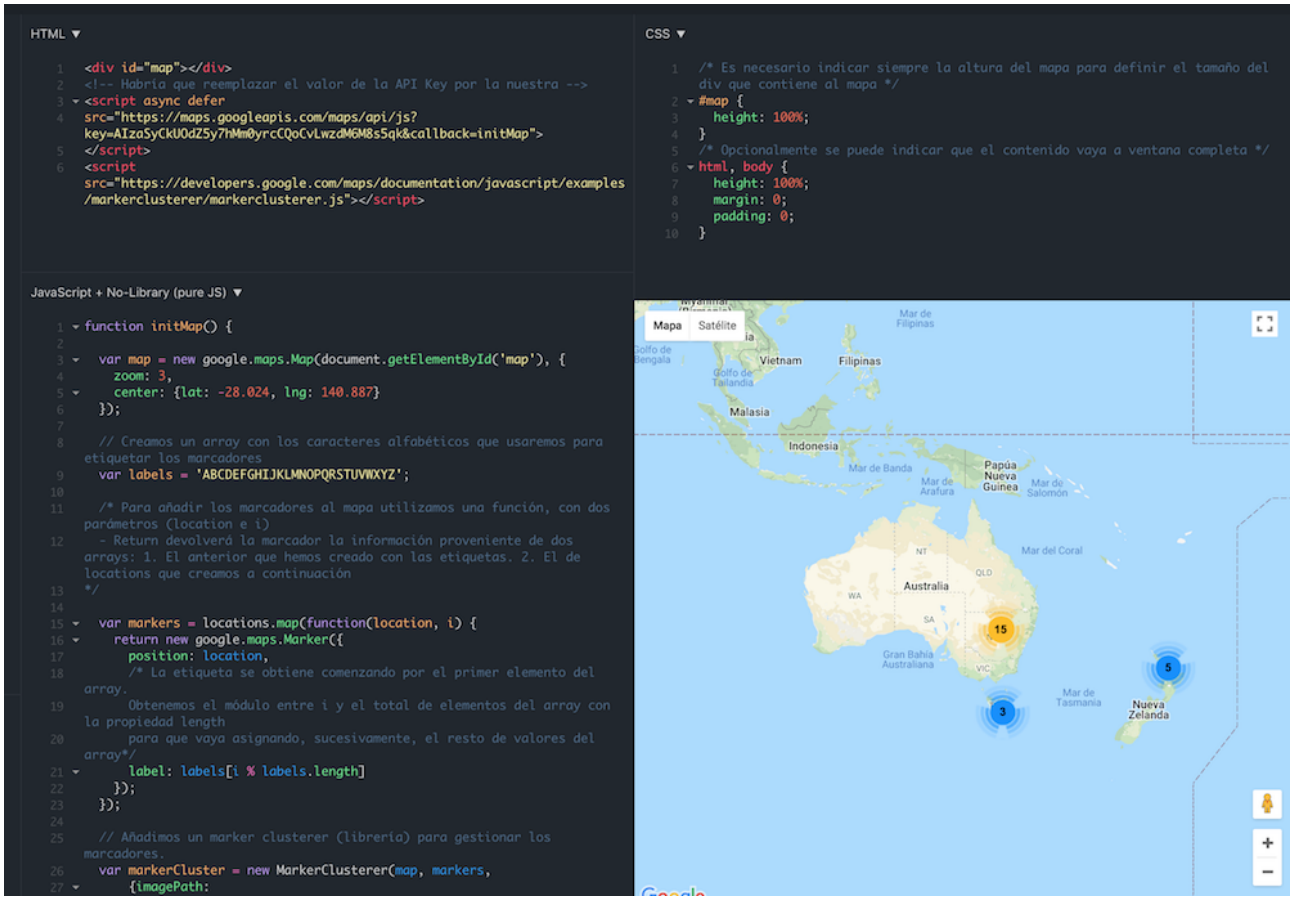
¿Para qué nos sirve JavaScript en un proyecto periodístico?

- 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 (Node.js)

¿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:



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.

> 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 scripts, así que os sonarán de lo trabajado con Phyton.

¿QUÉ ES JAVASCRIPT?

¿Qué es?

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

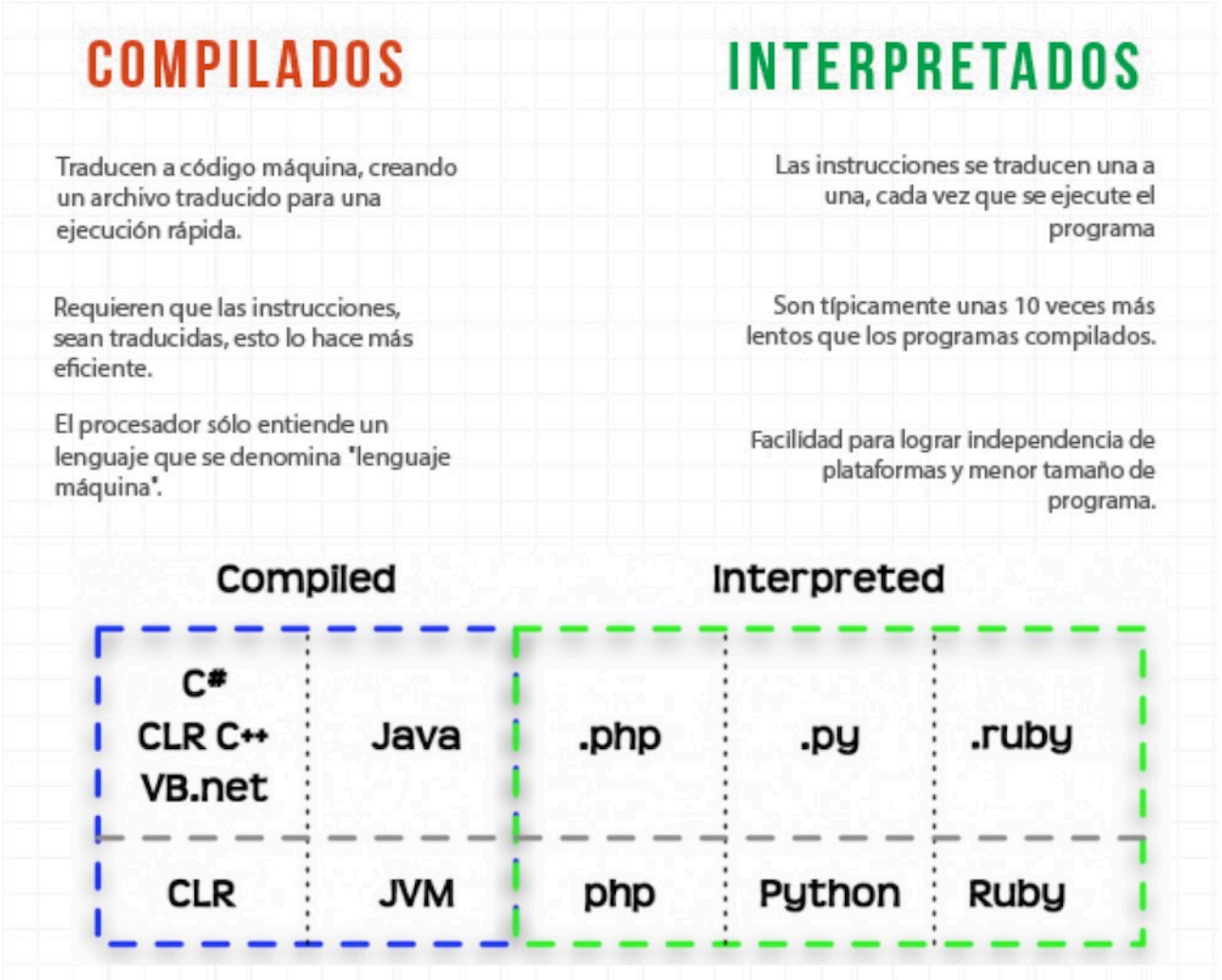


Figura. Lenguajes compilados vs Lenguajes interpretados

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.
- **Encapsulamiento**. Los datos se «encapsulan» o aislan para evitar que puedan ser manipulados o cambiados de estado.
- **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

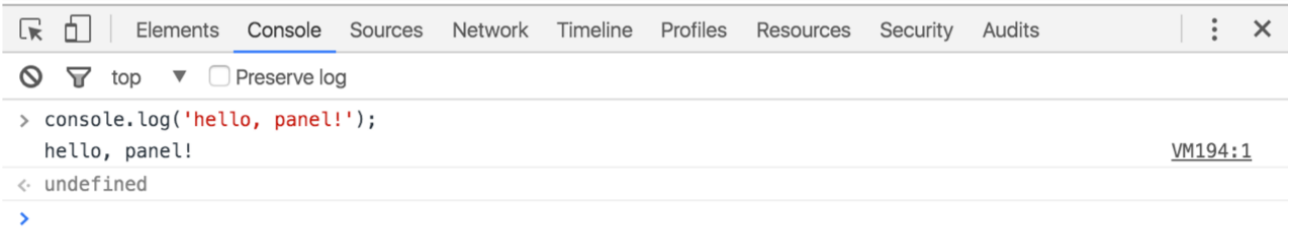
1. *Abre el editor de código en JSFiddle con el ejercicio.*
2. *Revisa el código html. Fíjate que el código JS aparece en el fichero HTML. Prueba el resultado que produce el botón. ¿Cómo crees que funciona comprobando el código?*

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.*

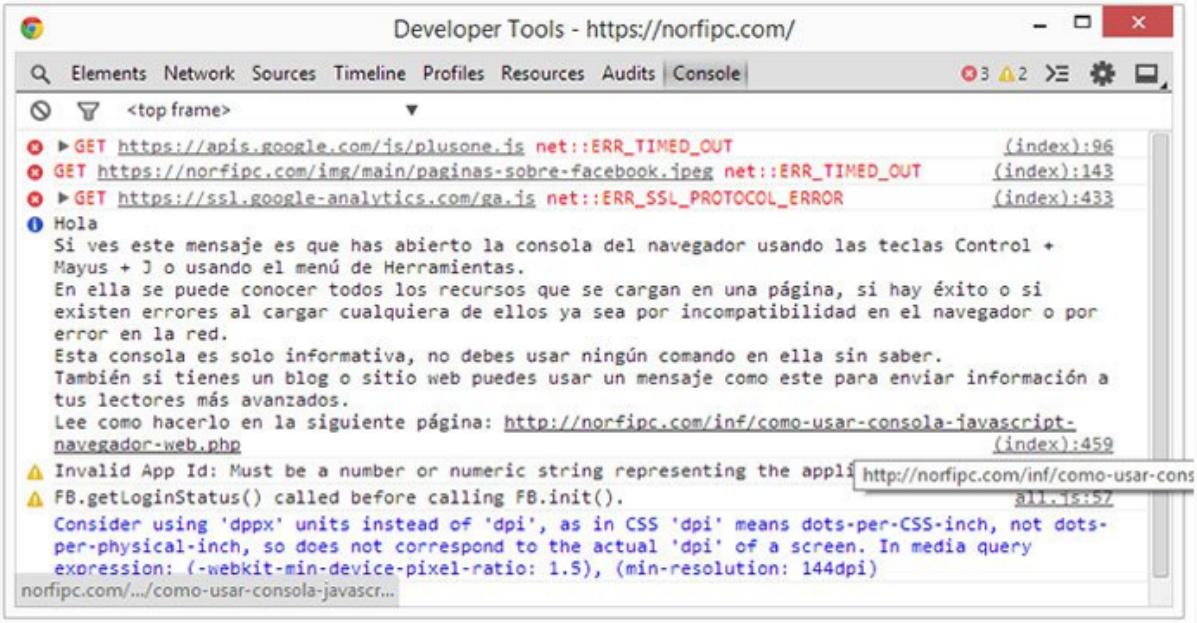
Ejemplo de código en la consola del navegador

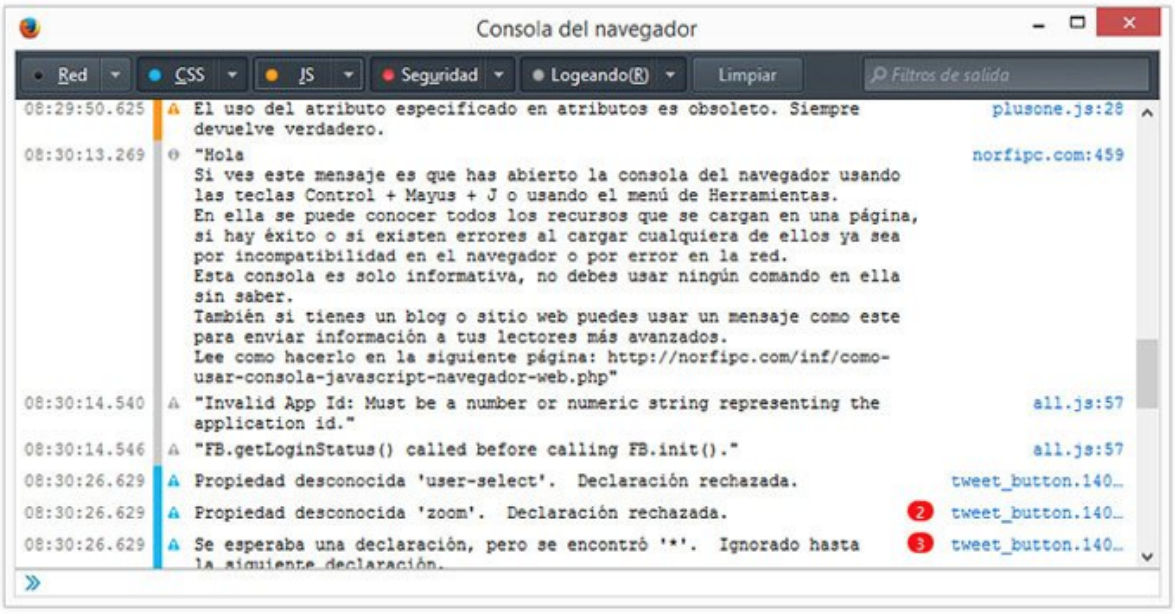
Utilidad de la consola. Para qué sirve.

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

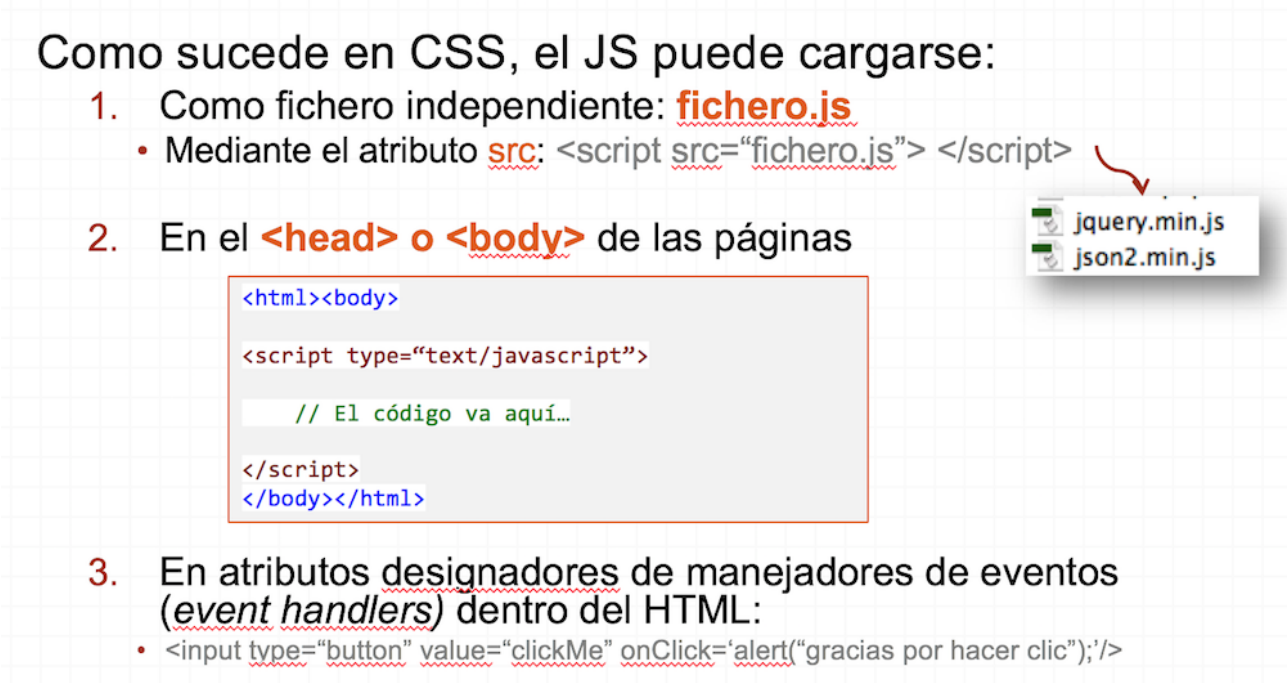
Permite:

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





¿DÓNDE APARECE?



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

Revisemos ahora, en detalle, alguna de estas funciones.

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 web de Maldita.es (maldita.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

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

TIPOS DE DATOS

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

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

String

```
// string
var companyName = “Google”;
```

Number

```
// number
var pi = 3.14;
var year = 2013;
```

Boolean

```
// boolean
var flag = true;
var FALSE = false;
```

Para estructuras condicionales

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 EN JAVASCRIPT

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 JavaScript, los usuarios pueden declarar una variable usando 3 palabras clave que son **var**, **let** y **const**.

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

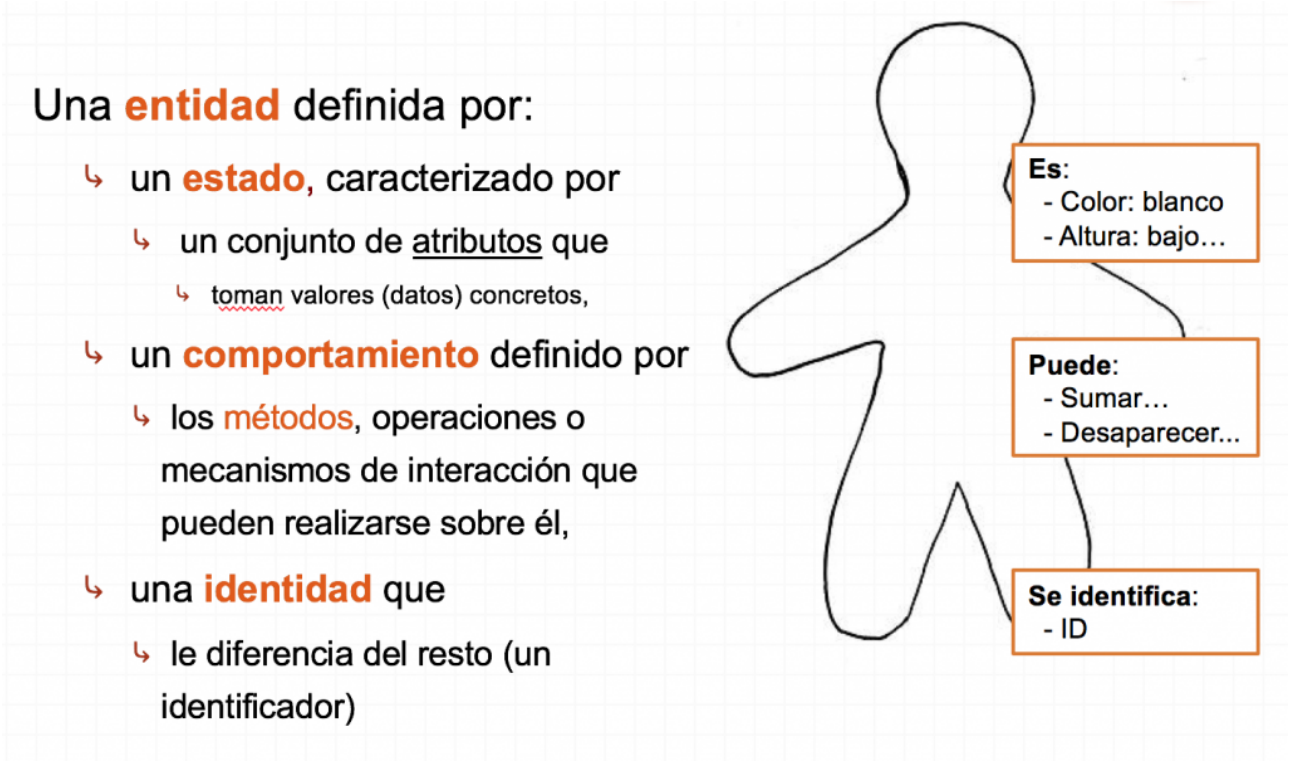
CONCEPTOS DE POO

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

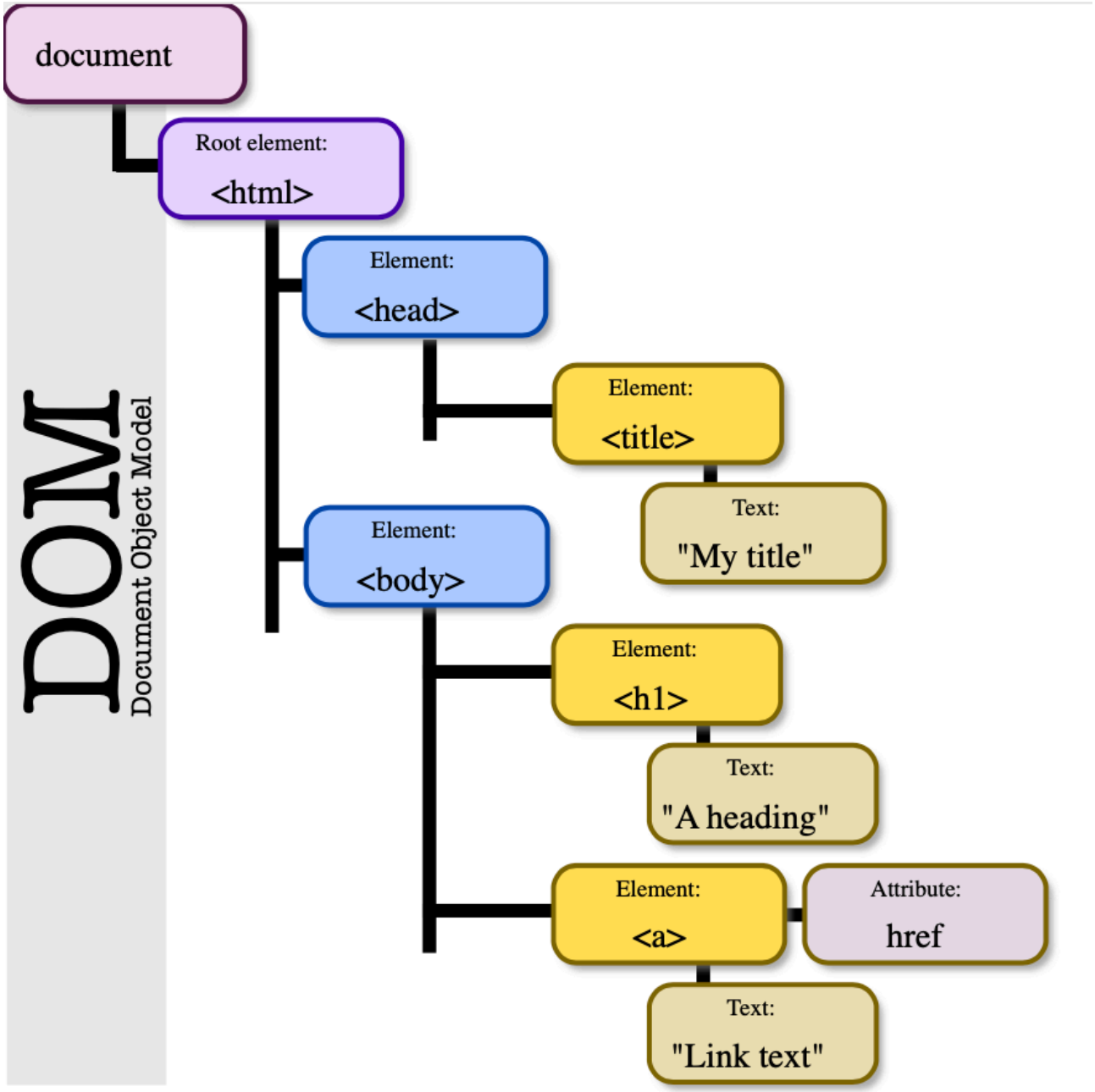
Objetos



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.

Los eventos se producen sobre alguno de los elementos del **DOM (Document Object Model)**



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, solo **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 emplea la API de Google Maps para crear un mapa que agrupa (cluster) marcadores en función del nivel de zoom.

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.

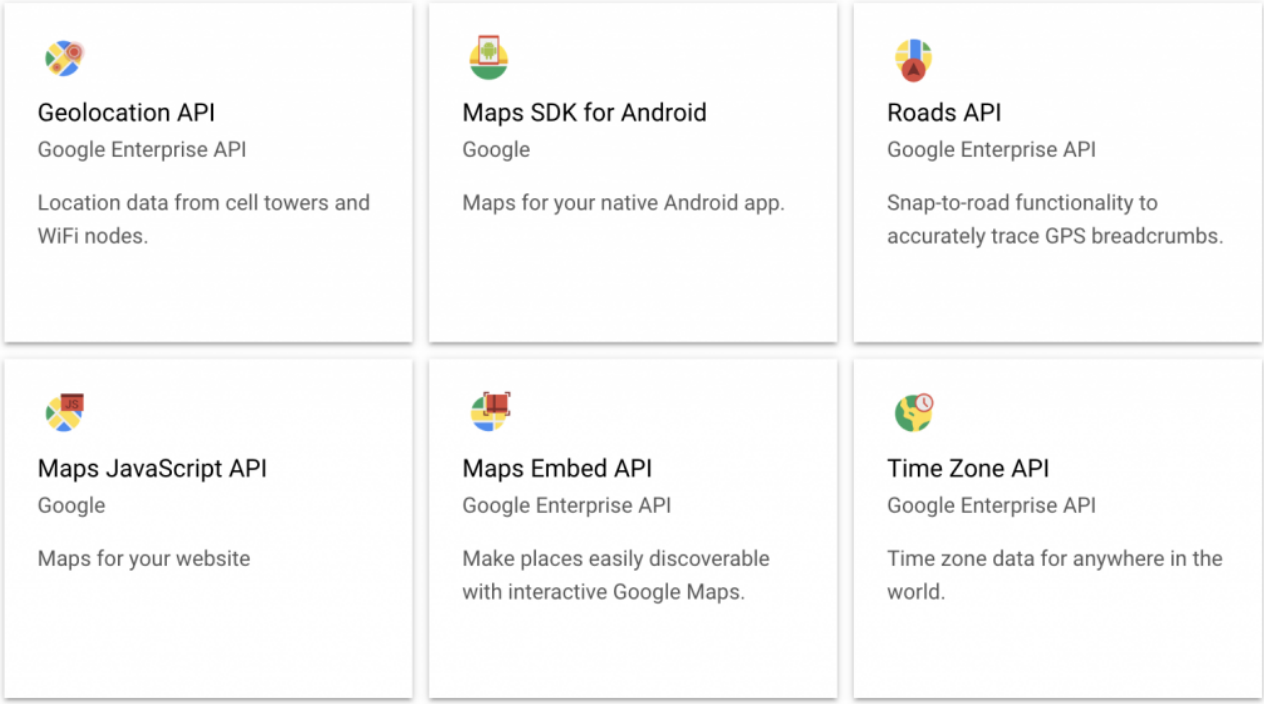
Una API es el libro de instrucciones que nos permite conocer cómo trabajar con

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. Actualmente, por ejemplo, para las APIs en Google Maps y el resto de aplicaciones de Google, se gestionan a través de la Google Maps Platform de la Google Cloud Platform.



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

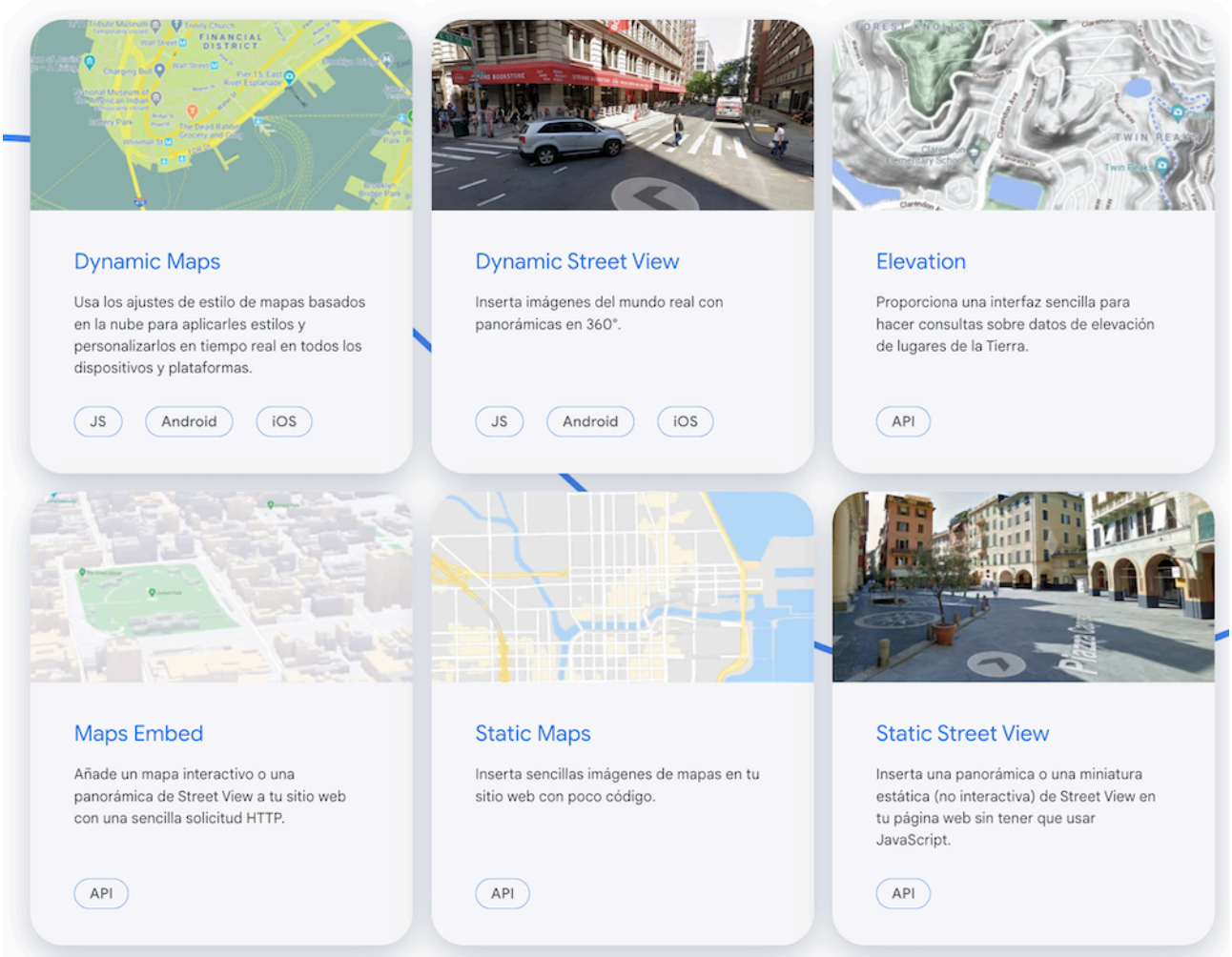
En este enlace puede verse una amplia comparativa de librerías.

API de Google Maps

GOOGLE MAPS API

Veamos todos estos elementos de JavaScript de forma práctica. 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.

Actualmente, es gratuito su uso hasta 28.500 llamadas o cargas mensuales (equivalente a unos 200 dólares). A partir de esa cantidad, esta es la relación de precios.



CREAR UN MAPA SIMPLE INCLUYENDO UN MARCADOR

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

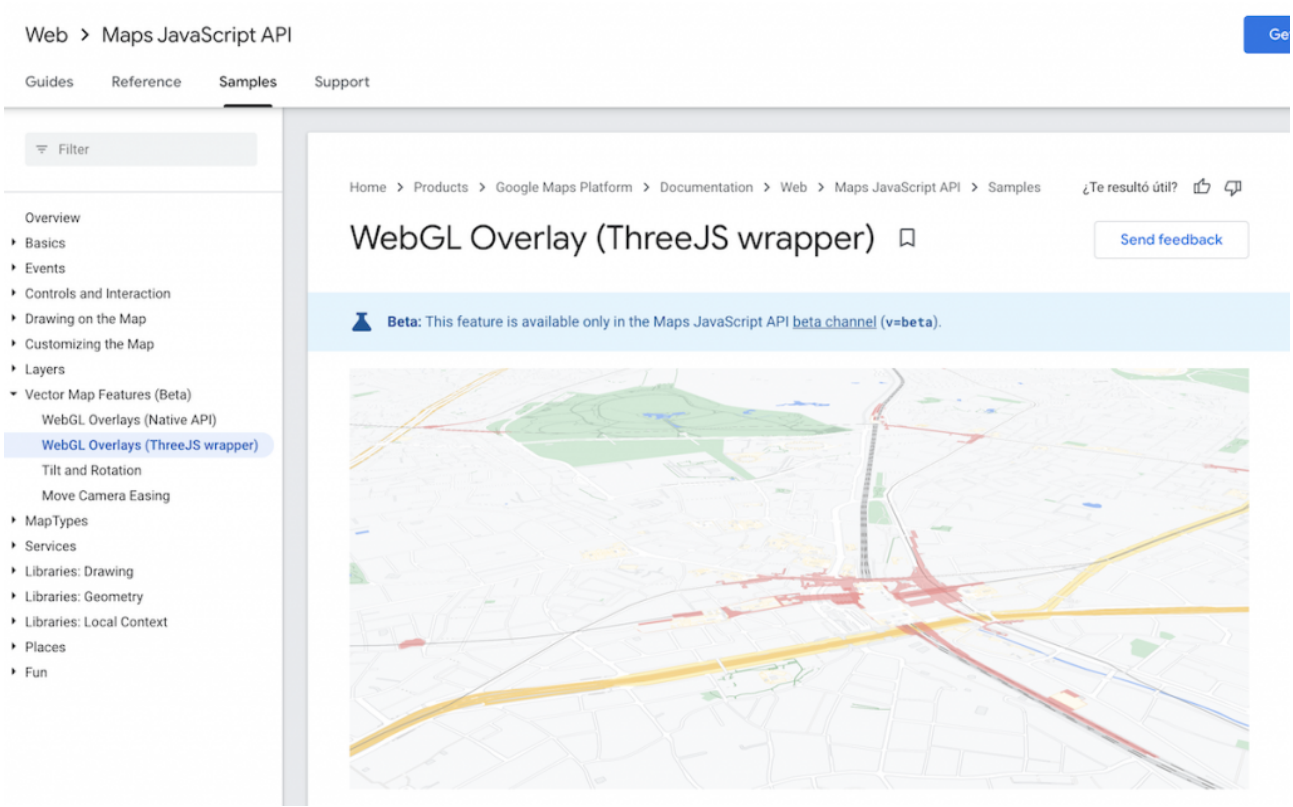
Vamos a elaborar un **mapa sencillo en el que incluiremos un marcador**:

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

VERSIÓN ACTUAL DE LA API

La nueva versión de la API ha introducido cambios en algunos aspectos, simplificando el código necesario, incluyendo también la codificación en TypeScript, o usando constantes.

Veamos algunas de sus múltiples opciones.



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 inform

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.

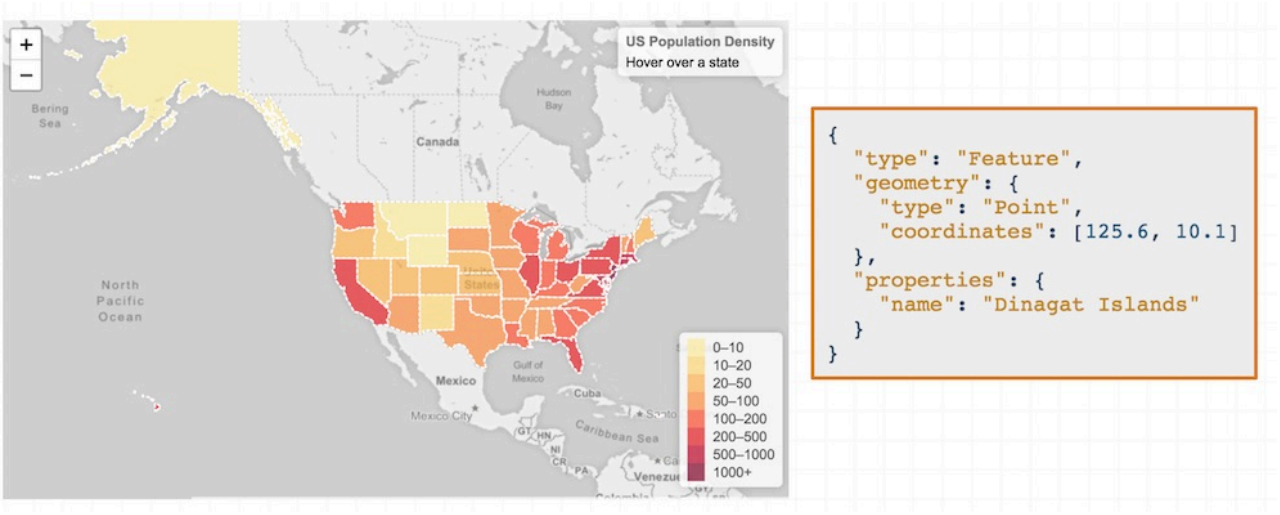
Otras librerías de mapas

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.

Ejemplo de mapa con imagen georeferenciada

En este enlace se encuentra la documentación completa del ejemplo

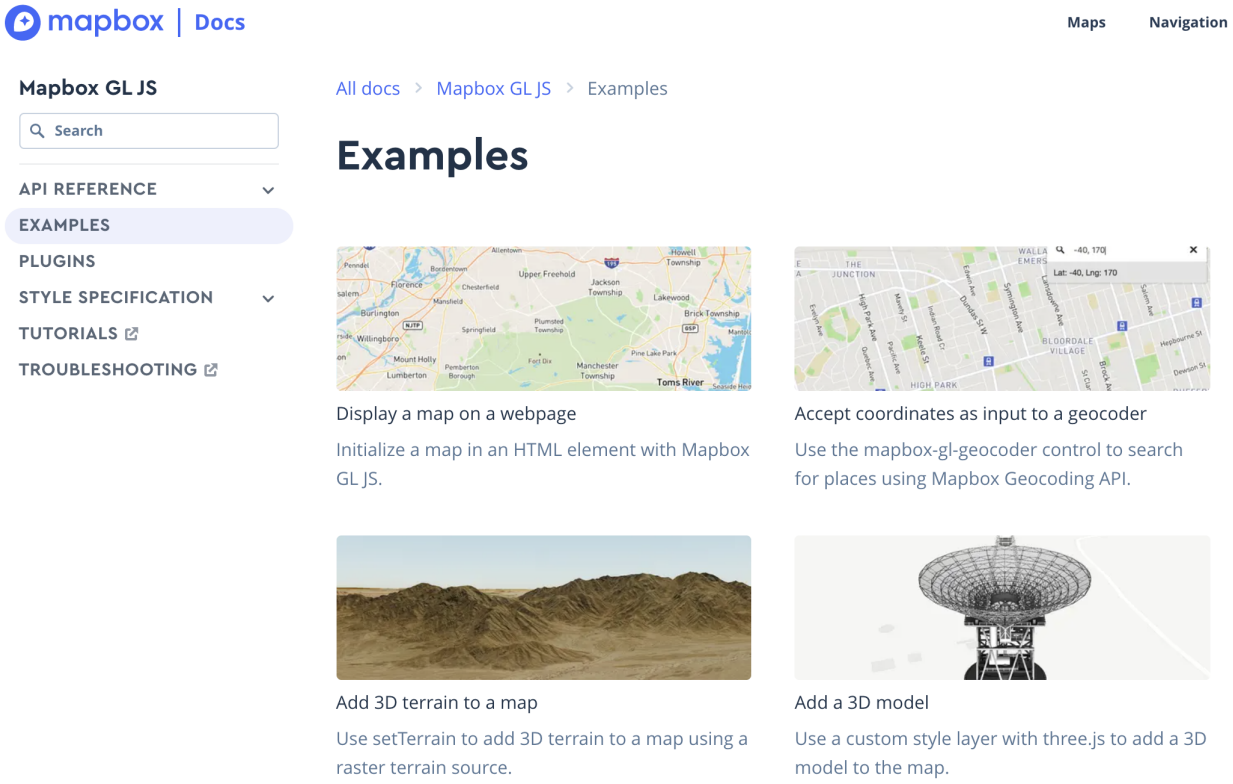
Una de las aportaciones interesantes de esta librería es que dispone de numerosos efectos que pueden darle un aspecto diferente a nuestras visualizaciones en mapas. La documentación de la API incluye muchas, pero destacaremos aquí algunas de ellas:

Ejemplo de mapa de terreno 3D con efecto niebla

Documentación completa.

Ejemplos de mapas que podemos crear

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



Una completa galería de estilos de mapa

Otro aspecto interesante es la amplia galería de estilos de mapa de la que dispone, así como de un editor (mapbox studio) que permite realizar cualquier edición de estilo que se precise, lo que ofrece enormes posibilidades de personalización del mapa.

Google Chart

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.

OTRAS LIBRERÍAS DE DATAVIZ

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

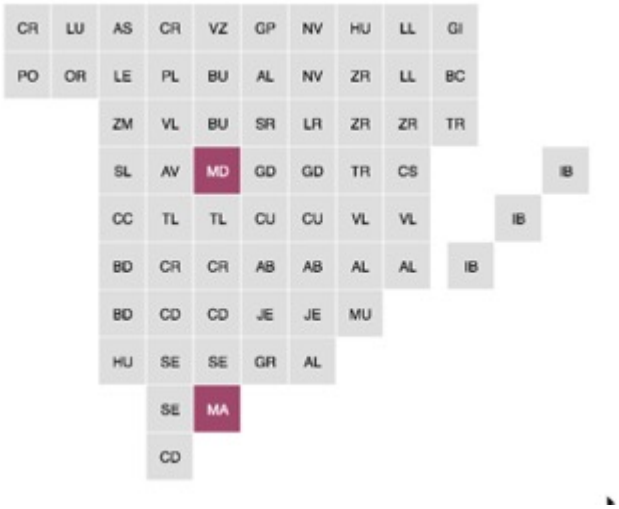
D3.js es una librería JS para manipular documentos basados en datos. Se utiliza para realizar visualizaciones complejas. Cuenta con una amplia galería de visualizaciones.

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.

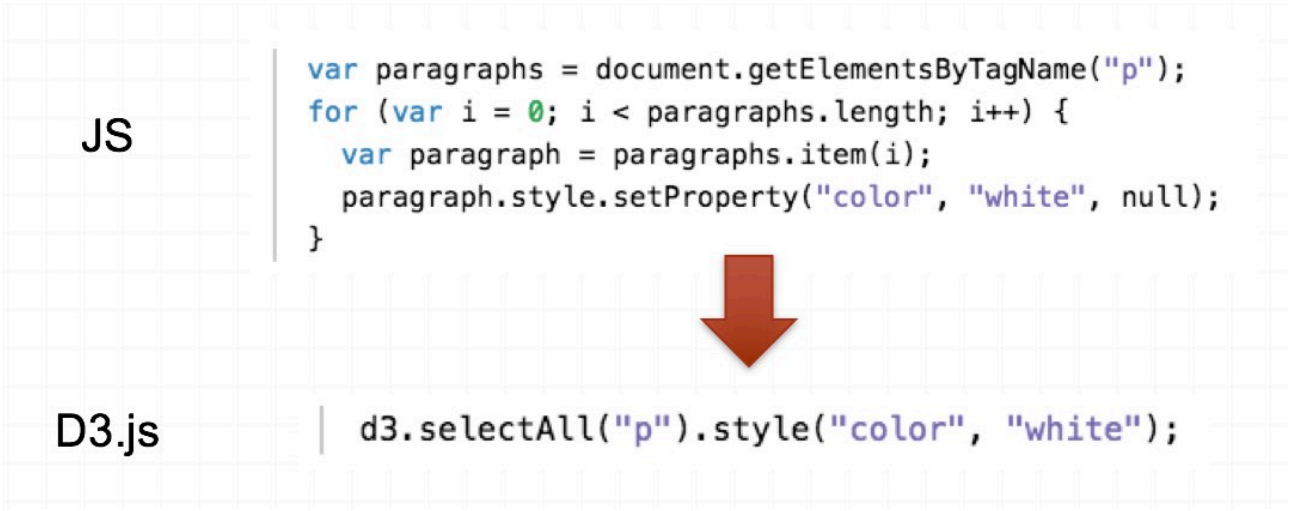
Sobre un mismo conjunto de datos permite realizar varias transformaciones. Por ejemplo, sobre un array podemos:

- Crear una tabla
- Generar un gráfico interactivo en SVG

Y de una manera muy flexible y rápida.

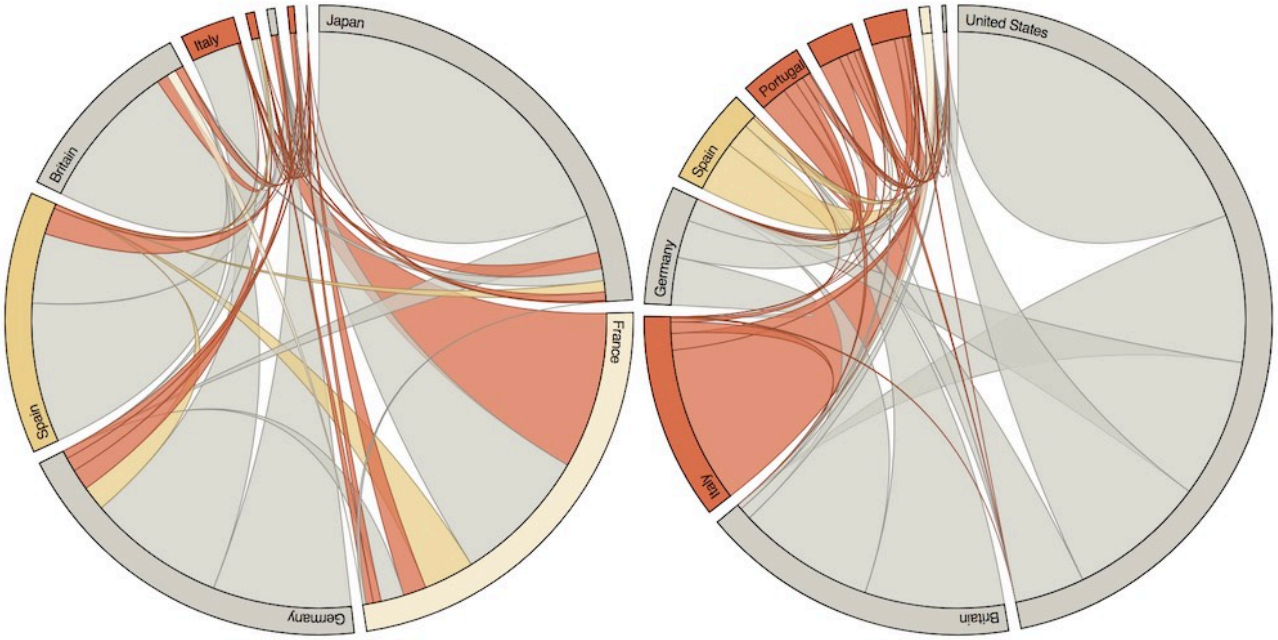
Sintaxis

Utiliza una sintaxis simplificada de JS para acceder a los selectores del DOM: W3C Selectors API



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,

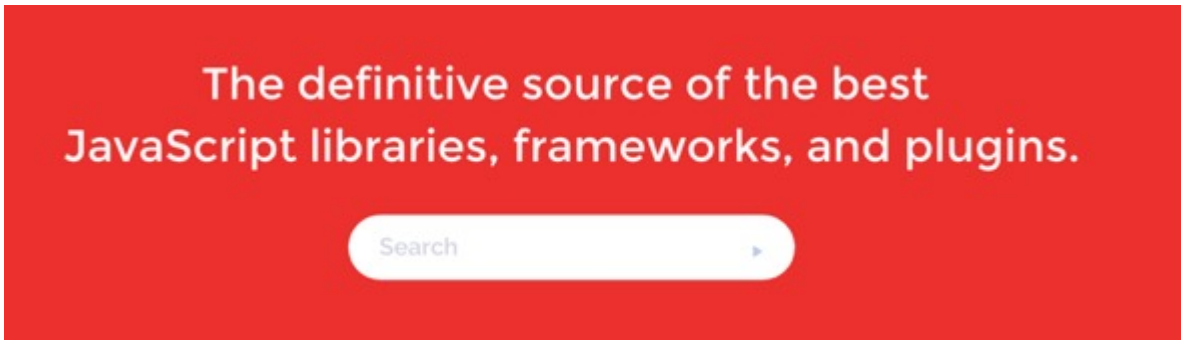
pero las visualizaciones de gráficas que hace son muy limpias y ligeras.

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.

FRAMEWORKS PARA REPRESENTAR DATOS EN ENTORNOS 3D

El desarrollo de entornos de visualización inmersivos ofrece una nueva oportunidad para explorar las posibilidades de visualizar datos.

Actualmente, el framework A-FRAME, usando tecnología de WebGL, HTML, CSS y JavaScript, permite representar modelos 3D directamente en el navegador.

BABIAXR

Usando A-FRAME como base, BabiaXR es una librería en desarrollo orientada a la visualización de datos en entornos 3D.

Para profundizar

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.