INTRODUCCIÓN A JAVASCRIPT

JAVASCRIPT EN ACCIÓN

Visualización de datos. Ejemplo de diagrama Sankey

Ver versión alternativa en Codepen.io

Visualización de secuencias sunburts

PRESENTANDO A JAVASCRIPT

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, 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 o CodePen.io). 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.

¿QUÉ ES JAVASCRIPT?

¿Qué es?

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

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. (De manera alternativa puedes abrirlo en Codepen)*
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)
- O usar botón derecho, y seleccionar la opción Inspeccionar.


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

¿QUÉ ES EL DOM? DOCUMENT OBJECT MODEL

El **DOM** es una **representación en forma de árbol** de tu página web. Cada etiqueta HTML (y su contenido) es un **nodo** que JavaScript puede leer y

manipular: crear, borrar, mover, cambiar clases, estilos o texto.

Ideas clave

- **Documento**: la página cargada en el navegador es document.
- **Nodos**: elementos (, <p>), atributos, y nodos de texto.
- **Árbol**: un nodo raíz <html> con ramas <head> y <body>.
- **Interacción**: JS accede al árbol para *leer* y *cambiar* el contenido o el estilo.

Operaciones típicas con JS

1) Seleccionar nodos

```
const titulo = document.querySelector('h1');
const botones = document.querySelectorAll('.btn');
```

2) Leer / escribir contenido

```
titulo.textContent = 'Nuevo título';
// Evita abusar de innerHTML si no necesitas HTML
```

3) Clases y estilos

```
titulo.classList.add('resaltado');
titulo.style.color = '#0ea5e9';
```

4) Crear e insertar nodos

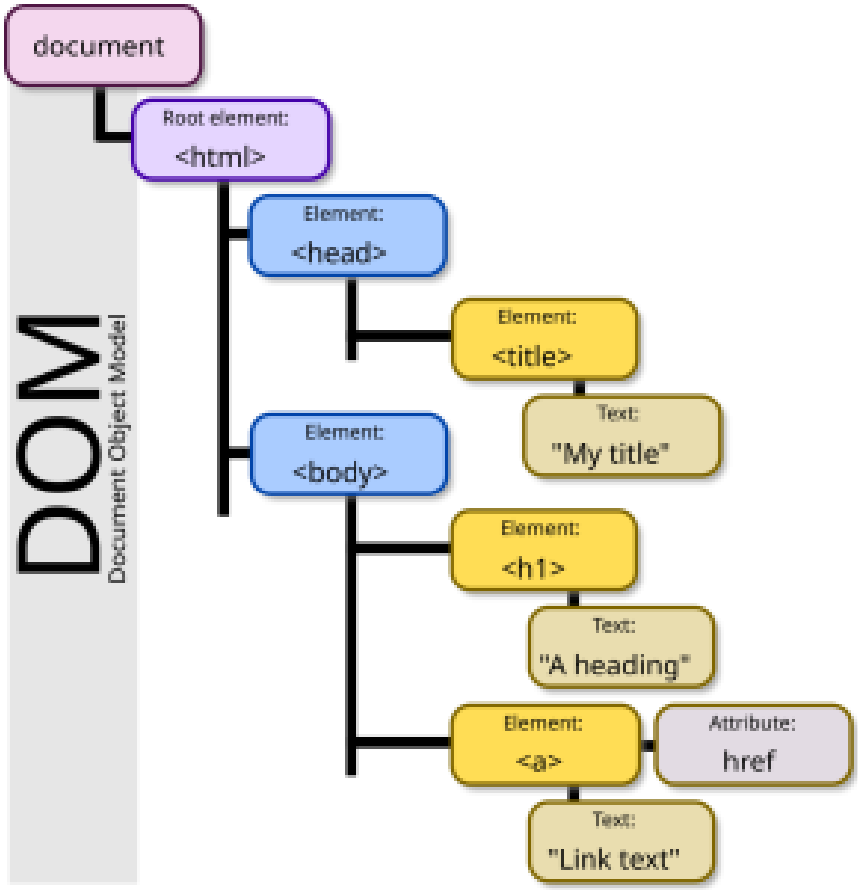
```
const li = document.createElement('li');
li.textContent = 'Elemento dinámico';
document.querySelector('ul').appendChild(li);
```

5) Eventos

```
document.getElementById('cta').addEventListener('click', () => {
alert('Hiciste clic');
});
```

Árbol DOM

El navegador convierte tu HTML en un **árbol DOM**. JavaScript navega este árbol para seleccionar, crear, mover o eliminar nodos.



¿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 la red X 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.

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 WebDevelopper 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 Sraping 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 en JSFidle o en este de Codepen.

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 en JSFidle o en CodePen

2. Reaccionar a eventos

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 de JSFidle o en este de Codepen.

Ejercicio

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

Solución en JsFidle o en Codepen

3. Modificar contenido en HTML

Otra función es modificar contenido que ya está cargado en el html (JsFidle) – Ver en Codepen

Ejercicio

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

Solución JsFidle – Codepen

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 (JsFidle) o Codepen, 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.

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 JsFidle – Codepen

Ejemplo de validación de contraseña con condicionales

En este ejemplo podemos ver un caso más complejo (JSFidle) o en Codepen en el que validamos una contraseña en un formulario para que cumpla tres condiciones.

5. Cambiar o modificar atributos

En este ejemplo (JsFidle) o Codepen 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.

Ejercicio

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

- https://files.ciberimaginario.es/07-master/js/lighton.png
- https://files.ciberimaginario.es/07-master/js/lightoff.png

Solución JSFidle – *CodePen*

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 JSFidle – Codepen]
- Salida html – *document.write()* [Ver ejemplo JsFidle – Codepen]
- Elemento html – *inner.html()* [Ver ejemplo JsFidle – Codepen]
- Consola de navegador – *console.log()* [Ver ejemplo JsFidle – Codepen]

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.