

UT01.1: Introducción a JavaScript

En este ejercicio vamos a realizar una pequeña introducción para trabajar con JavaScript. Los objetivos que buscamos con la realización de este ejercicio son los siguientes:

- Reconocer dónde y cómo podemos integrar JavaScript en una página HTML.
- Introducir elementos básicos que utilizamos en JavaScript:
 - o Funciones en JavaScript.
 - o Objeto `console` y comentarios de línea.
 - o Búsqueda de elementos con el API de DOM.
 - o Acceso a los atributos de un elemento.
 - o Capturar eventos asociando manejadores de eventos.

Con este primer ejercicio vamos a sentar las bases para ir trabajando a lo largo del curso.

1. ¿Dónde localizar el código JavaScript?

Comenzamos el ejercicio creando una página HTML de prueba, debemos personalizar la página con nuestro nombre.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Ejercicio de Introducción a JavaScript</title>
</head>
<body>
  <h1>Nombre del alumno</h1>
</body>
</html>
```

Código 1 Creación de página base.

1.1. Elemento `<script>`

Mediante este elemento podemos insertar código JavaScript en nuestra página. El siguiente ejemplo muestra cómo añadimos código JavaScript en el elemento `<script>`.

```
<script>
  var nombre = "Nombre alumno";
  alert ("Bienvenido: " + nombre);
</script>
```

Código 2 Mi primer código JavaScript.

En este ejemplo estamos declarando una variable *nombre* a través de la palabra reservada `var` y le asignamos un valor. JavaScript tiene un control débil de tipos, por lo que no es necesario indicar el tipo de variable que estamos utilizando. Para continuar, estamos invocando la función predeterminada `alert` que muestra en un cuadro de diálogo un mensaje de texto que estamos

pasando como argumento. Para esta cuestión, tenemos una cadena de texto concatenada al valor de la variable declarada en primera instancia.

El código anterior está basado en la versión HTML5. En esta versión el atributo `type` es opcional, por lo que no es necesario especificarlo. En versiones anteriores de HTML el atributo es obligatorio, por lo tanto, debemos declarar el elemento de la siguiente manera `<script type="text/javascript">`.

1.2. Insertar el elemento en el <head>

Para integrar el código JavaScript en una página podemos hacerlo dentro del elemento `<head>`.

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Ejercicio de Introducción a JavaScript</title>
  <script>
    var nombre = "Nombre alumno";
    alert ("Bienvenido: " + nombre);
  </script>
</head>
```

Código 3 Código JavaScript en el Head.

El navegador al parsear el código HTML encuentra el elemento `<script>`, en ese momento el Intérprete de JavaScript tendrá que interpretar el código que contiene el elemento para su ejecución. La siguiente imagen muestra cuál es el resultado de su ejecución.

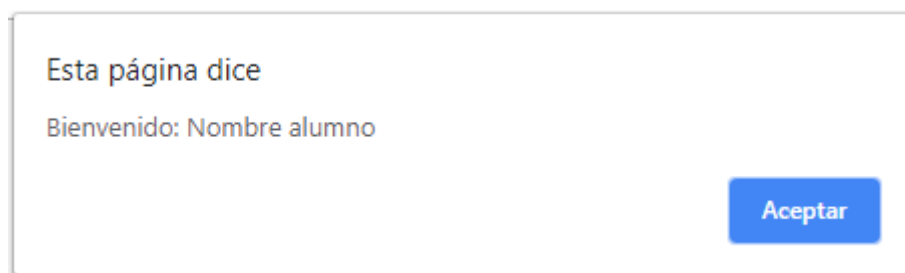


Imagen 1 Ejecución de función alert.

La función `alert` resulta muy útil para probar el resultado de nuestro código, pero no deberíamos utilizarla en una aplicación real, siendo mejor idea utilizar otros mecanismos para comunicar información a los usuarios, por ejemplo, utilizando el API de DOM, como veremos posteriormente.

1.3. Insertar el elemento en el <body>

En sitios web que hacen uso de HTML 4 es habitual localizar el código JavaScript en el elemento `<head>`. El problema de esta estrategia es que hasta que el motor del navegador no termina con la interpretación del código, no puede continuar con el renderizado de la página, produciendo un retraso en la finalización de la carga de la página.

Una buena práctica en HTML5 es declarar los múltiples elementos `<script>` que disponga la página justo antes del cierre del elemento `<body>`. Con esta estrategia permitimos que se renderice el contenido de la página sin retrasos, mostrando contenido al usuario para que posteriormente se finalice con la interpretación del código.

```
<body>
  <h1>Nombre del alumno</h1>
  <script>
    var nombre = "Nombre alumno";
    alert ("Bienvenido: " + nombre);
  </script>
</body>
```

Código 4 Código JavaScript en el Body.

El resultado en esta ocasión es exactamente igual que el resultado que en el caso anterior, aunque la interpretación del código se hace al finalizar el proceso de carga, en lugar de hacerse al comienzo.

1.4. Código JavaScript externo

Otra buena práctica al desarrollar un sitio web es externalizar el código JavaScript del código HTML, lo que permite su reutilización, así como mejora el rendimiento del sitio web ya que el código JavaScript al tratarse de un recurso estático, puede ser cacheado por el navegador.

Para externalizar el código JavaScript creamos un nuevo fichero con extensión `.js`, con nuestro código JavaScript.

```
var nombre = "Nombre alumno";
alert ("Bienvenido: " + nombre);
```

Código 5 Fichero JavaScript externo.

Para hacer referencia al código externo utilizamos el atributo `src` haciendo referencia al fichero `js`.

```
<body>
  <h1>Nombre del alumno</h1>
  <script src="Ejercicio1.js"></script>
</body>
```

Código 6 Código js externo.

El resultado es el mismo que en los ejemplos anteriores.

2. Funciones en JavaScript

Las funciones son elementos del lenguaje que nos permite modularizar el código de nuestra aplicación. Las funciones funcionan de manera similar a los métodos de un lenguaje orientado a objetos, pero a diferencia de ellos, las funciones no están vinculadas a la instancia de un objeto, pudiendo ejecutarse de forma aislada.

Para comprobar el funcionamiento de una función vamos a añadir una nueva al fichero externo creado en el paso anterior. En el siguiente ejemplo, hemos definido una función denominada `cuadrado` que espera como argumento un número, y devuelve como resultado el cuadrado de dicho número.

```
function cuadrado(numero){  
    var resultado = numero * numero;  
    return resultado;  
}
```

Código 7 Definición de una función.

Como podemos observar en el ejemplo, utilizamos la palabra reservada `function` para definir una nueva función, seguida del nombre de la función que estamos creando. Entre paréntesis, y separados por comas, tenemos la lista de parámetros que acepta esta función.

La palabra reservada `return` nos permite devolver el control al punto donde se invocó la función, devolviendo el valor calculado en el cuerpo de ésta, en esta ocasión un valor numérico.

Para invocar a una función lo hacemos utilizando el nombre de la función, y entre paréntesis pasamos los argumentos a la función separados por comas.

```
var nombre = "Nombre alumno";  
alert ("Bienvenido: " + nombre + " " + cuadrado(5));  
  
function cuadrado(numero){  
    var resultado = numero * numero;  
    return resultado;  
}
```

Código 8 Invocación de la función cuadrado.

Como podemos observar en el código, la función `cuadrado` ha sido invocada dentro de la función `alert`, pasándole como argumento el número 5. El resultado de esta ejecución es la siguiente imagen.

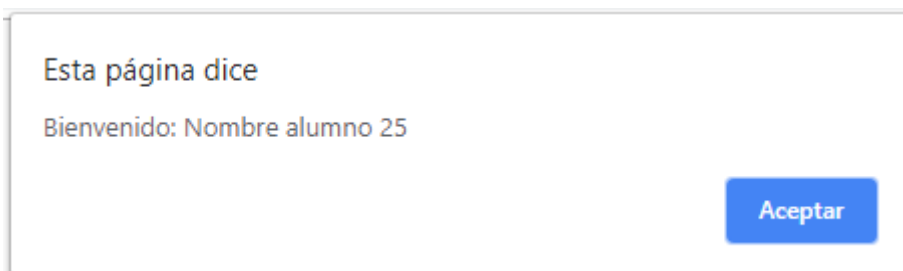


Imagen 2 Resultado de invocar la función cuadrado.

El código anterior puede parecer a primera vista que debería lanzar un error, ya que estamos invocando una función antes de su declaración. El intérprete realiza un proceso denominado **hoisting** por el cual todas las declaraciones de funciones y de variables son ordenadas al comienzo del código, evitando este tipo de errores. El siguiente código muestra el orden en el que realmente es interpretado el código que estamos escribiendo.

```
function cuadrado(numero){  
    var resultado = numero * numero;  
    return resultado;  
}  
var nombre = "Nombre alumno";  
alert ("Bienvenido: " + nombre + " " + cuadrado(5));
```

Código 9 Proceso hoisting del código JavaScript.

3. Objeto console y comentarios de línea

La consola del Intérprete de Comandos es una herramienta fundamental en el testeo de nuestro código. Si abrimos las herramientas de desarrollo del navegador veremos como una de las pestañas se trata de la consola, en ella podremos observar cada uno de los errores que el Intérprete está generando para poder resolverlos.

Tenemos un objeto que nos permite interactuar con la consola, publicando mensajes para observar el resultado de la ejecución de nuestro código, estamos hablando del objeto `console`. Este objeto tiene varios métodos que permiten mostrar mensajes en la consola.

- `log`. Muestra un mensaje por la consola.
- `error`. Muestra un mensaje por la consola, etiquetando el mensaje de error.
- `warn`. Muestra un mensaje por consola, etiquetando el mensaje como advertencia.

El siguiente código muestra como una función sin argumentos y sin valores de retorno, invoca a los diferentes métodos del objeto `console`.

```
function ejemploConsola(){  
    console.log("Mensaje mostrado por consola.");  
    //Mensaje mostrado por consola.  
    console.log("Cuadrado de 5: " + cuadrado(5));  
    //Cuadrado de 5: 25  
    console.error("Mensaje de error."); //Mensaje de error.  
    console.warn("Mensaje de aviso."); //Mensaje de aviso.  
}  
ejemploConsola();
```

Código 10 Uso del objeto console

El código anterior muestra cuatro mensajes en la consola, dos orientados a mostrar mensajes por la consola, uno de error, y uno de aviso. El código en verde muestra comentarios de línea con el resultado que obtendríamos en la consola. Los comentarios en JavaScript coinciden con los tipos de comentarios del lenguaje Java. Los caracteres `//` se utilizan para comentar una línea. La consola de la ejecución de esta función quedaría de la siguiente manera.

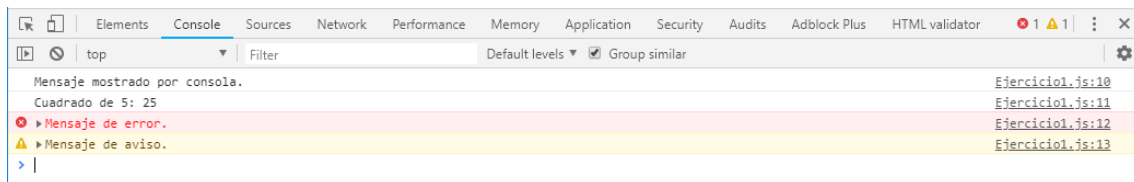


Imagen 3 Consola del Intérprete de Comandos

4. Búsqueda de elementos con el API de DOM

Vamos a integrar la ejecución de la función `cuadrado` con el código HTML. Añadimos tres nuevos elementos, un input de tipo numérico para recoger un valor entre 1 y 10, un botón que permita realizar el cálculo, y por último una capa para mostrar el resultado. El siguiente código muestra estos cambios.

```
<body>
  <h1>Nombre del alumno</h1>
  Elige un número:
  <input type="number" id="valor" max="10" min="1"
value="5"><br>
  <button>Obtener cuadrado</button>
  <div id="resultado"></div>
  <script src="Ejercicio1.js"></script>
</body>
```

Código 11 Código HTML para calcular el cuadrado de un número.

Como podemos observar en el código HTML, los elementos `<input>` y `<div>` definimos dos atributos `id`, por los cuales podemos seleccionar ambos elementos y acceder a sus atributos y propiedades utilizando el API de DOM. La siguiente función recoge el valor del `<input>` y muestra el valor seleccionado por la consola.

```
function calcularCuadrado(){
  // Seleccionamos los elementos HTML a través de su id.
  var input = document.getElementById("valor");
  var resultado = document.getElementById("resultado");

  console.log("Valor obtenido del input: " + input.value);
  //Valor obtenido del input: 5
  resultado.innerHTML = "El cuadrado es: " +
cuadrado(input.value);
}
calcularCuadrado();
```

Código 12 Función que utiliza el API de DOM.

El objeto `document` representa al documento HTML de la página que tenemos cargada. Uno de sus métodos es `getElementById`, por el cual obtendremos una referencia al objeto con identificador que pasamos como argumento. La variable `input` tiene una referencia al elemento con identificador `"valor"`.

La variable `input`, al ser una referencia al elemento HTML, puede acceder a sus atributos. En el ejemplo la consola muestra el valor del `<input>` utilizando el atributo `value` de la referencia.

De la misma forma podemos acceder a las propiedades del elemento `<div>`. Con la variable `resultado` tenemos la referencia al elemento. Con la propiedad `innerHTML` podemos asignar el contenido HTML de un elemento de la página.

5. Capturar eventos

Un evento es una acción que el usuario o que el navegador puede desencadenar. Con nuestro código JavaScript podemos **capturar** cada uno de los eventos que se producen independientemente de su naturaleza, y asignarles una función que se ejecuta cada vez que ocurra dicho evento. La función que se ejecuta cuando un evento es capturado se denomina **manejador de evento**.

5.1. Eventos de producidos por el ratón

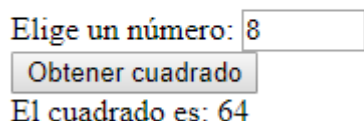
Estos eventos se producen por el uso del ratón. Algunos ejemplos pueden ser clicar en un elemento, pasar o sacar el ratón por un elemento, hacer un doble cliqueo en un elemento. Para este ejemplo vamos a utilizar el más típico que es clicar en un elemento. En el elemento `<button>` de nuestra página le vamos a añadir un atributo `onclick`. El contenido del atributo es código JavaScript que se ejecuta cada vez que cliqueamos en el botón, este código es por tanto el manejador de evento. Veamos el siguiente ejemplo.

```
<button onclick="calcularCuadrado()">Obtener cuadrado</button>
```

Código 13 Manejador de evento onClick.

El código anterior ejecuta el contenido del atributo `onclick` cada vez que el usuario cliquee en el botón, y por tanto este código es el manejador del evento. Al leer el código vemos que estamos invocando la función `calcularCuadrado` creada anteriormente. El resultado lo podemos ver en la siguiente imagen.

Nombre del alumno



Elige un número:

El cuadrado es: 64

Imagen 4 Resultado de clicar en el botón "Obtener cuadrado".

5.2. Eventos de carga de la página

Otra categoría de eventos que pueden ocurrir en una página son los eventos relacionados con su carga. Los más típicos aquí son el que avisa de que ha finalizado su carga o el que lo hace cuando vamos a cambiar a otra página, nos vamos a centrar en el primero de ellos.

Queremos ejecutar una función justo en el momento en el que termine de cargar la página, es decir, se hayan descargado todos los recursos que estamos definiendo en la página, CSS, imágenes, etc, y el motor del navegador haya finalizado la fase de pintura de todos los elementos contenidos en ella. Un objeto predeterminado que podemos utilizar en nuestro código es el objeto `window`, el cual hace referencia a la ventana del navegador con la que estamos trabajando, le vamos a asociar un evento `onload` para que se dispare al finalizar la carga de la página.

```
window.onload = ejemploConsola;
```

Código 14 Manejador de evento para el evento onload.

Como podemos ver, además de en el código HTML, podemos asignar manejadores de evento en el propio código JavaScript. En el ejemplo vemos como la función `ejemploConsola` será invocada al terminar de cargarse la página, mostrando los mensajes contenidos en la función en la consola.

Debemos fijar nuestra atención que hemos asignado la función manejadora del evento, al evento del objeto `window`, es decir, no lo estamos invocando al no utilizar los paréntesis. Veamos el siguiente ejemplo para ver qué pasa cuando invocamos la función porque es un error común.

```
window.onload = ejemploConsola();
```

Código 15 Manejador de evento para el evento onload invocando una función.

El Intérprete de Comandos al encontrar esta sentencia va a invocar la función `ejemploConsola`, la función se va a ejecutar mostrando los mensajes por consola, pero al no devolver nada, el evento `onload` no tiene ninguna función que ejecutar. No estamos garantizando que la función se ejecute al finalizar la carga de la página.