

Tema 2: JavaScript

PROGRAMACIÓN EN INTERNET

Pedro Delgado Pérez

Juan Boubeta Puig

Grado en Ingeniería Informática
Departamento de Ingeniería Informática



2023-2024

Contenido

- **Introducción**
- Programación básica
- DOM: acceso al contenido
- Respuesta a eventos
- Validación de formularios



Introducción

Definición y versiones

- **JavaScript** (JS) es un lenguaje que, interpretado en el navegador (lado del cliente), permite *cambiar el contenido del HTML, estilos CSS* y el comportamiento de la web.
- **Historia:** lo inventa Brendan Eich (1995) como solución a la baja velocidad de internet (reducción peticiones al servidor).
- JavaScript implementa el estándar *ECMAScript*:
 - Primera edición: ECMAScript 1 / ES1 (1997)
 - Última edición: ECMAScript 2023

Las más conocidas:

- **ES5:** [ECMAScript 5](#) (2009)
- **ES6:** [ECMAScript 6](#) (2015)

Diferencias:

<http://es6-features.org/>

Introducción

¿Cómo hacer uso de JavaScript?

Tres formas para incluir JS en HTML:

1. *Dentro de etiquetas HTML:*

```
<input type="button" onclick="alert('Hola.');" />
```

2. *Dentro del HTML* (en cualquier lugar, recomendado al final del *body*):

```
<script type="text/javascript"> alert('Hola.*)</script>
```

3. *Archivo externo:* (de uso mayoritario en la asignatura)

```
<script type="text/javascript" src="file.js"></script>
```

Consultar archivo **alert.html**

Al inicio de *body* (por si el navegador no soporta JS o está deshabilitado):

```
<noscript>Tienes JavaScript deshabilitado.</noscript>
```

Lista de frameworks JS:

http://en.wikipedia.org/wiki/Comparison_of_JavaScript_frameworks

Contenido

- Introducción
- **Programación básica**
- DOM: acceso al contenido
- Respuesta a eventos
- Validación de formularios

Programación básica

Sintaxis

Aspectos básicos iniciales:

- Se recomienda **indentar** el código para mejorar la lectura.
- No es obligatorio, pero sí altamente recomendable, terminar cada sentencia con ';'.
- Define dos tipos de valores: *literales* y *variables*.
- **Case sensitive**: Se distingue entre mayúsculas y minúsculas.
- **Comentarios**: son del estilo C o Java
 - `/* Comentario multilínea */`
 - `// Comentario en una línea`
- Palabras reservadas.

false	return
for	switch
if	break

Buenas prácticas:

Las típicas de cualquier lenguaje, pero también algunas específicas:

https://www.w3schools.com/js/js_best_practices.asp

Programación básica

Variables

- **Variables:** Se define *solo el nombre, no su tipo*.
 - **let**: declara una variable local con ámbito de bloque.
 - **const**: el valor asignado inicialmente no puede cambiar.
- Nomenclatura de declaración:
 - **let** nombreVariable; (no recomendado)
 - **let** nombreVariable = 1; | **const** nombreVarConst = 1;

Nota: Se recomienda **dar siempre un valor de inicialización**.

- El nombre de una variable puede estar formado por letras, números (excepto primer carácter), dólar (\$) y guion bajo (_).

```
let num1 = 1;
let num2 = 2;
let resultado = num1 + num2;
alert(resultado); // Mostraría 3
```

Programación básica

Tipos de variables

Una variable puede almacenar durante la ejecución (entre otros):

- Entero o decimales:

- `let iva = 21;`
- `let precio = 13.50;`

```
> let iva = 21
```

```
< undefined
```

```
> typeof(iva)
```

```
< 'number'
```

```
> let falso = false;
```

```
< undefined
```

```
> typeof(false)
```

```
< 'boolean'
```

- Booleanos:

- `let falso = false;`
- `let verdadero = true;`

- Cadenas: (también existe objeto *String*, pero es mejor usar el literal).

- `let cadena = "Cadena";` o `let cadena = 'Cadena';`
- `let cadena_vacia = "";` o `let cadena = '';`
- *Plantilla de cadena (ES6):* ``Mi cadena es: ${cadena}``

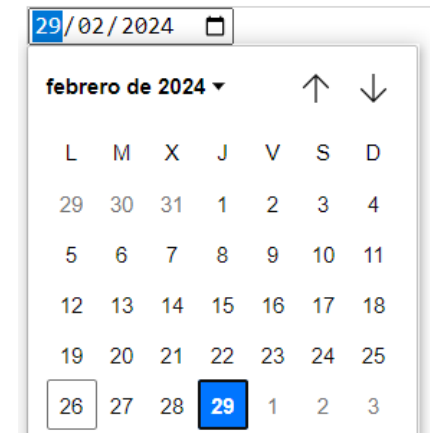
- Vectores o arrays:

- `let dias = ["Lunes", "Martes", ..., "Domingo"];`
- `let lunes = dias[0];`
- `let martes = días.at(1);` *Introducido en ES2022*

Programación básica

Tipos de variables

- Objetos (pueden contener propiedades y métodos)
 - `let persona = { nombre: "Lola", edad: 21 };`
 - `let n = persona.nombre;` o `let n = persona["nombre"];`
- Date: objetos fecha:
 - `let fecha1 = new Date();` //F. actual
 - `let fecha2 = new Date(2020, 2, 5);`
- Nuevas clases de ES6:
 - Map, Set, Promise,...
- Valores primitivos:
 - Variables sin definir (*undefined*): `let a;` o `a = undefined;`
 - Indicar que un objeto es inexistente (*null*): `persona = null;`



Más información sobre tipos.

Programación básica

Estructuras de control y operadores

- **Mismas estructuras de control/operadores que C, C++ o Java:**
 - Estructuras de control: Ejemplo:

```
if (num > 0 && num < 5) { ... }
```

 - Incluye también el bucle de rango “*for...in*”:

```
for (i in dias) { alert(dias[i]); }
```
 - Operadores: Entre otros:
 - Aritméticos (+,-,*,/), lógicos (!,&&,||) y relacionales (>, >=, ==, !=,...)
 - Asignación (=) y asignación especiales (+=, -= ...)
 - Comparación estricta (===, !==) sin conversiones ([más info](#)).
- Algunos operadores realizan una función distinta *según el contexto*:
 - Suma de valores enteros: `5 + 3 //Devuelve 8`
 - Concatenación de cadenas: `alert("Hola" + "mundo");`
- En caso de ambigüedad, existen unas reglas de prioridad.

Programación básica

Funciones

- **function** nombreFuncion ([param1,param2,...]) {
 ...
 [return valor;]
}

- Puede tener o no parámetros.
- Puede devolver o no valores (al llegar a un *return* la función para).
- No se indica si la función devuelve o no un valor.
- Los parámetros pueden tener *valores por defecto* (a partir de ES6).

- Ejemplo:

- Definición:

```
function suma ( n1, n2){  
    return n1 + n2;  
}
```

- Invocación: suma (1, 2);

Consultar el fichero
[cambiar-color.html](#)

Programación básica

Notación flecha y Funciones útiles (i)

ES6 incorpora una notación adicional (**flecha**) para definir *funciones anónimas*:

- Definición: lo marcado en naranja es opcional para bloques de una instrucción
`let suma = (n1, n2) => { return n1 + n2; }`
 - Invocación: `suma (1, 2);`
-

Métodos propios de String y funciones relacionadas:

- toUpperCase / toLowerCase:
 - `s.toUpperCase();` //pasa a mayúsculas la variable s
- substring(inicio[,final])
 - `s.substring(2);` //devuelve la cadena a partir del índice 2.
 - `s.substring(1, 5);` //devuelve la cadena entre las posiciones 1 y 5.
- split(separador) //divide el contenido de la variable por el
//carácter separador. El resultado es un Array.
 - `s.split(" ");`
- parseInt / parseFloat:
 - `i = parseInt("4");` //recibe la cadena / devuelve un entero.

Programación básica

Funciones útiles (ii)

Métodos comunes de String y Array:

- length: `v.length` //devuelve la longitud del String/Array
- at / IndexOf:
 - `v.at(0);` //devuelve el elemento/carácter en la posición 0
 - `v.indexOf('b');` //devuelve la posición de 'b' (si no, devuelve -1)

Métodos y operadores propios de Array:

- pop / push:
 - `arr.pop();` //elimina el último elemento del Array
 - `arr.push(3);` //inserta al final del Array
- join (separador): //une los elementos del Array, separados por ','
 - `arr.join(",");` //El resultado es un String
- Operador spread: propagación de elementos de una colección (ES6)
 - `let p1 = ["hello", true, 7];`
`let p2 = [1, 2, ...p1]` // [1, 2, "hello", true, 7]
- Iteradores array: *forEach, filter, find...* [Más info](#)

Programación básica

Posibles salidas

- Escribir contenido dentro de una etiqueta HTML:
 - `innerHTML = content`
- Escribir directamente en la salida del documento HTML:
 - `document.write()`

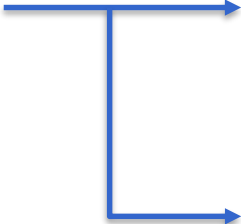
Nota: Llamarlo tras la carga completa del HTML borra su contenido.

- Consola de alerta / de confirmación:

- `alert()` / `confirm()`

- Escribir en la consola:

- `console.log();`



chrome://new-tab-page dice
Información enviada con éxito

Aceptar

chrome://new-tab-page dice
¿Seguro que desea eliminar?

Aceptar

Cancelar

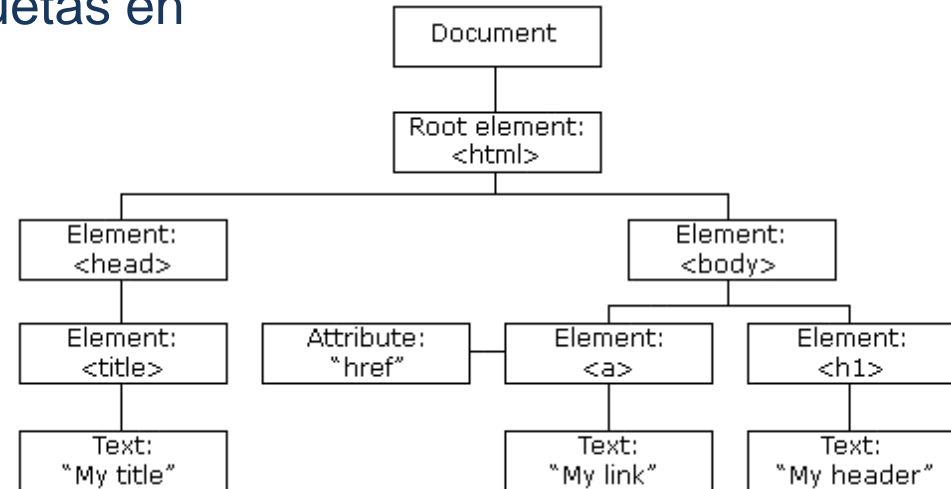
Contenido

- Introducción
- Programación básica
- **DOM: acceso al contenido**
- Respuesta a eventos
- Validación de formularios

DOM

Árbol de nodos

- Gracias al **DOM** (*Document Object Model*) se puede modificar con JS el contenido de etiquetas, eliminarlas o añadirlas,...
- Los navegadores transforman el texto del *.html* en una *estructura en árbol* con todas las etiquetas en forma de objetos.



- Los más comunes: (+info)

- **Document**: Nodo raíz.
- **Element**: Representa a las etiquetas (el único tipo de nodo que puede tener atributos y ser padre de otros elementos).
- **Attr**: Representa los atributos.
- **Text**: El texto dentro de una etiqueta.
- Otros: *Node*, *Comment*, *Event*, *DocumentType*...

DOM

Acceso a los nodos

- **Acceder a etiquetas:**

```
let inputs = document.getElementsByTagName("input");
```

- Obtiene un *HTMLCollection* con todos los inputs de la web. Acceso al 1er input:
`inputs[0];`

- **Acceder a elementos por nombre:**

```
let formulario=document.getElementsByName("form_cont");
```

- Obtiene el formulario cuyo *name* = "form_cont"
- El atributo name suele ser único. Si no lo es, devuelve un *HTMLCollection*.

- **Acceder a elementos por ID:** consultar archivo [añadir-texto.html](#)

```
let form_id = document.getElementById("form_cont_id");
```

- Obtiene el formulario cuyo *id* = "form_cont_id"

- **Acceso con selector CSS:** [+info](#)

```
let elemsClase1 = document.querySelectorAll(".clase1");
```

- Obtiene todos los elementos con *class* = "clase1", y lo guarda como *NodeList*.

DOM

Creación y eliminación de nodos

- **Creación:**

- En primer lugar se crea un elemento:

- `let parrafo = document.createElement("p");`

- Se crea el nodo para introducir texto:

- `let contenido = document.createTextNode("Texto");`

```
> parrafo
```

```
< <p></p>
```

```
> contenido
```

```
< "Texto"
```

- El texto se añade al elemento:

- `parrafo.appendChild(contenido);`

```
> parrafo
```

```
< <p>Texto</p>
```

DOM

Creación y eliminación de nodos

- El elemento se añade a otro elemento (al *body* u otro elemento):
 - `document.body.appendChild(parrafo);`
 - `document.body.children[1].appendChild(parrafo);`

```
> document.body
```

```
< ▼ <body>
```

```
...
```

```
<p>Texto</p>
```

```
</body>
```

- **Eliminación:**

- Para eliminar un nodo, se busca y, a través de su padre, se borra. Además, se borran todos los hijos de dicho nodo automáticamente:
 - `let parra = document.getElementById("parrafo");`
 - `parra.parentNode.removeChild(parra);`

```
> parra
```

```
< <p id="parrafo">Texto</p>
```

```
>
```

```
> document.body
```

```
< ▼ <body>
```

```
...
```

```
</body>
```

DOM

Acceso a atributos y manejo CSS

- **Al tomar un objeto, se puede acceder/modificar su/s atributo/s:**

- objeto.**setAttribute**("atributo", "valor");
- objeto.**getAttribute**("atributo"); / objeto.**atributo**

Ejemplo: `alert(document.getElementById("link_logo").href);`

```
> document.body.appendChild(enlace)
```

```
< <a id="link_logo" href="https://www.uca.es"></a>
```

```
> alert(document.getElementById("link_logo").href);
```

chrome://new-tab-page dice

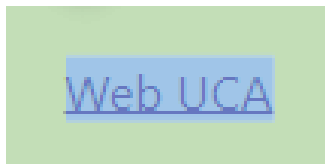
https://www.uca.es/

- **Se puede acceder a los estilos CSS definidos:**

`alert(document.getElementById("link_logo").style.padding); *`

```
> enlace
```

```
< ▶ <a id="link_logo" href="https://www.uca.es" style="padding: 1.2em">...</a>
```



chrome://new-tab-page dice

1.2em

** No recomendado uso de atributo style*

DOM

Acceso a atributos y manejo CSS

- **Estilo lower camel case.** Si el atributo CSS al que se accede es, por ejemplo, `border-top-color`, en DOM se elimina guiones y cada palabra empieza en mayúscula (a excepción de la primera):

```
document.getElementById("link_logo").style.borderTopColor
```

- **Se puede agregar o eliminar dinámicamente estilos CSS:**

```
document.getElementById("form").querySelectorAll("input")  
    .forEach((inp) => inp.classList.add("greenBorder")) ;
```

Complete name (*):

Carlos Fernández

Email (*):

carlos.fernandez@uca.es

A todos los *input* del formulario con id *form* se le añade la clase *greenBorder*.

Contenido

- Introducción
- Programación básica
- DOM: acceso al contenido
- **Respuesta a eventos**
- Validación de formularios

Eventos

Tipos de eventos

- Los eventos se denominan con **on** más la acción del evento:
 - **onload** → se llama tras cargar un elemento. Se suele usar en *body*.
 - **onclick** → click en una etiqueta
 - **onmouseover** → el ratón “pasa” por encima de una etiqueta
 - **onmouseout** → el ratón “deja” la etiqueta
 - **onscroll** → controlar el scroll. ([Ver ejemplos](#) del final de la página).
 - **onsubmit** → para controlar el envío de datos de formularios

- Ejemplo: `<input type="button" id="b1" value="Dar saludo" onclick="alert('HOLA'); "/>`

- **Recomendación:** invocar una función externa en un fichero JS:

`<input type="button" id="b1" onclick="saludar();" />`

[Más información](#) y [lista completa de eventos](#).

Eventos

Listeners

- Para una *mayor separación de HTML y JavaScript*, los eventos se registran directamente en el script (**Forma recomendada**)
- Ejemplo:

```
document.addEventListener('DOMContentLoaded', () => {  
    let but = document.getElementById("b1");  
    but.addEventListener("click", () => saludar());  
    //más listeners  
})  
  
function saludar() {  
    alert('HOLA');  
}
```

- También se pueden borrar los eventos con *removeEventListener*.
- Consultar el archivo **evento-listener.html**

Eventos

Delegación de eventos

Utilidad: Gestión de elementos cuyo manejo de eventos sea similar.

Uso: Se consulta el **target** para localizar el lanzador del evento: [+info](#)

```
document.addEventListener('click', ev => {  
    if(ev.target.matches(".claseBtn")) saludar();  
    else if(ev.target.tagName === "INPUT") despedir();  
})
```

- **Ejercicio:** ¡Démosle vida a la calculadora!

Se proporciona un fichero **cálculos.js**:

- Estudia su código y enlázalo con **calculadora.htm** (tema CSS).
- Completa la función “computar”.
- Haz que funcione el botón de suma e igual de dos formas:
 1. Método 1: Mediante la invocación directa a las funciones.
 2. Método 2: Mediante *listeners*.

Contenido

- Introducción
- Programación básica
- DOM: acceso al contenido
- Respuesta a eventos
- **Validación de formularios**

Formularios

Validación (i)

Aunque los input de HTML5 facilitan la validación de los formularios, **el resto de las validaciones se realizan mediante JS:**

- Al cargar una página, en el DOM se crea un array (*forms*) hijo de *document*, con todos los formularios de la página:
 - `document.forms[0];`
- A su vez, dentro de *forms*, se crea un array *elements* con todos los elementos del formulario (*inputs*, *text*, etc.):
 - `document.forms[0].elements[3];`
- Sin embargo, **resulta más recomendable** acceder al formulario a través del atributo *name* o *id* (y de forma análoga, a sus elementos):
 - `document.getElementsByName("formAlta");`
 - `document.forms["formAlta"].usuario;`

Donde usuario es un input cuyo nombre o id es “usuario”

Formularios

Validación (ii)

- Lo más usual es un método de validación que devuelve **true** o **false**. Se suele incluir en el evento **onsubmit** de un *form*.

```
<form ... .. onsubmit="return validacion()">...</form>
```

- *Comprobaciones*: valor de un campo relleno, con formato adecuado o dentro de un rango, o el valor de dos o más campos es coherente:

```
if(val == null || val.length == 0 || /^s+$/ .test(val))  
    return false;
```

- **Expresiones regulares**: Se usa [JS RegExp](#)

En el ejemplo anterior, “test” comprueba que el valor es una cadena formada por una o más letras “s” (únicamente, “s” es principio y fin).

- Métodos útiles: de RegExp (*test*, *exec*) o [String](#) (*match*, *search*, *replace*).
- Comprobación de expresiones: <https://regexr.com/>

Formularios

Validación (iii)

```
function esNumerico(elem) {  
    let expresionNumerica = /^[0-9]+$/;  
    if(elem.value.match(expresionNumerica)) {  
        return true;  
    }else{  
        alert("Por favor, inserta solamente números");  
        return false;  
    }  
}
```

Ejercicio: Usemos el formulario de [SGSOACS 2021 - página de ejemplo.html](#)

- Añade un input “Age” de tipo texto en el formulario que represente la edad:
Age(*):<input required type="text" id="age" placeholder="Insert age">
- Logra que, cuando se pulse el botón de “*Submit request*”, se haga la validación “*esNumerico*” sobre “Age” ([incluida ya en validation.js](#)).
- Extiende el código para comprobar que hay un espacio blanco intermedio en el campo “*Complete name*”. Por ejemplo, “Lola Mora”.
- ¿Cómo se puede lograr que se validen ambos campos de una sola vez?

Formularios

Validación (iv)

- También puede comprobarse si se ha marcado o no una opción:

```
<select id="opciones" ...>
  <option value="-1"> Seleccione un valor</option>
  <option value="1"> Rojo </option>
  <option value="2"> Verde </option>
</select>
```

Opción 1: por el “value” (cadena vacía si no hay nada seleccionado)

```
let val = document.getElementById("opciones").value;
if(val == -1 || val == "") return false;
```

Opción 2: por el índice en la lista (-1 si no hay nada seleccionado)

```
let indice =
  document.getElementById("opciones").selectedIndex;
if( indice == 0 || índice == -1) return false;
```

Referencias

Bibliografía:

- 📖 **JavaScript-The definitive guide. Master the World's Most-Used Programming Language.** D. Flanagan. O'Really Media, 7ª Ed., 2020
- 📖 **Eloquent JavaScript: A Modern Introduction to Programming.** M. Haverbeke. Starch Press, 2ª Ed., 2014
 - <http://eloquentjavascript.net/> (también Node.js)

Recursos web:

- Webs con tutoriales:
 - <http://www.w3schools.com/js/>
 - <https://desarrolloweb.com/manuales/manual-de-ecmascript-6.html> (ES6)
- DOM *events*:
 - <http://www.w3.org/TR/DOM-Level-3-Events/>
- Recursos para validación de formularios:
 - <https://evontech.com/component/easyblog/10-useful-javascript-form-validation-libraries.html?Itemid=159>