tema 4 ~ DEW

¿Qué es JavaScript?

> Un lenguaje de **scripting** → ***interpretado*** (en este caso por el navegador) y ***compilado*** al ejecutar el código por primera vez.

¿Cómo se puede incluir en una página HTML?

Externo

Hay un archivo separado que contiene el código JavaScript.

```
<script type='text/javascript' src='fichero.js'></script>
```

Empotrado

Como contenido del elemento <script>

```
<script type='text/javascript'> ... código ... </script>
```

Inline

Como valor del atributo de un evento HTML

Primer ejemplo

```
<script type="text/javascript">
    var age = parseInt(prompt("Please enter your age", 29));
    if (isNaN(age)) { // if age is Not a Number (NaN)
        alert("You must enter a number");
    } else if (age < 23) {
        alert("You look more mature and sophisticated than " + age);
    } else {
        alert("You look younger than " + age);
    }
    document.write("<em>By the way I lie a lot!</em>");
</script>
```

`var age = parseInt(prompt("Please enter your age", 29));`

- Produce una petición para que escriba el usuario
- Convierte la entrada a un entero - parseInt()
- Asigna el resultado a la variable age.
  - El segundo parámetro de prompt (29) es el valor por defecto a asignar (puede omitirse)
  - Si el valor recibido no es un número, se asignará el valor especial NaN a la variable age

`alert("You must enter a number");`

- Muestra al usuario una caja de alerta con un mensaje

`document.write("<em>By the way I lie a lot!</em>");`

- Coloca texto en el documento que se visualiza en la ventana del navegador. Será interpretado por el navegador como HTML.

Funciones JavaScript

💡 Las funciones se definen en la cabecera, pero no se ejecutan hasta ser invocadas

Las funciones se deben declarar en la parte `<head>` del documento. Se cargarán en memoria antes de su invocación.

`history.back()` Devuelve al usuario al URL anterior en la lista de la historia del navegador.

`window.location` Contiene información sobre el URL de la página que se está visualizando. Se puede usar para dirigir al navegador a otro URL.

```
window.location="http://es.wikipedia.org/";
```

▼ Formas de activar JavaScript

1. MANEJADORES DE EVENTOS

Se pueden asociar a elementos HTML (onFocus, onBlur, onChange, onSubmit...)

```
<body onload="alert('Welcome - do stay a while')">
```

`onload` Ocurre después de la carga del contenido (cuando se alcanza la marca de cierre </body>)

2. DESDE UN ENLACE USANDO UN URL JavaScript

```
<a href="javascript:void TodaysMenu()">Today's menu</a>
```

Cuando se use el enlace se ejecutará el código JavaScript, si el resultado del código es una cadena, ésta sustituirá al documento actual en la ventana del navegador. Si, como aquí, se especifica "devolver nada" (void) entonces simplemente se ejecuta el código.

- Si no se indica que el tipo a devolver sea void, la función puede devolver una cadena de texto a visualizar en la ventana del navegador.
- `javascript:void` → especificador pseudo-protocolo
- `TodaysMenu()` → código a ejecutar, en este caso una función. Se definiría como una función que devuelve un URL al navegador dependiendo del día de la semana. Al pulsar sobre el enlace se llamará a la función

▼ Invocación de JavaScript

1. `document.write()`

Sólo puede usarse para escribir HTML **en el documento actual** conforme el navegador lo vaya procesando

- Cuando haya finalizado la carga entonces `document.write()` sobrescribirá el contenido de la ventana actual.
- Debe usarse la propiedad `innerHTML` o seleccionar un elemento para escribir en él tras cargar el documento.
  - `innerHTML` → Propiedad de JavaScript que te permite leer o cambiar el contenido HTML de un elemento del Modelo de Objeto de Documento (DOM). Para manipular el contenido de una página web de manera dinámica.

2. CONFIRMACIÓN

```
onclick="return confirm('Really take the link?')"
```

Genera una caja de mensaje y devuelve un booleano para indicar que ha seleccionado el usuario (ok o cancel).

Si devuelve true: sigue al enlace, en caso contrario no.

3. URL JavaScript ( )

> Se usa para crear dinámicamente contenidos en el documento

```
<a href="javascript:void TodaysMenu()">Today's menu</a>
```

4. USO DE FECHAS

> Crea un nuevo objeto (variable) de tipo fecha.

```
var today = newDate()
```

Por defecto → representa la hora y fecha actuales.

Guarda una referencia a la fecha en la variable `today`

METODO `today.getDay()`

- Devuelve un número que representa el día de la semana (0 para el Domingo, 6 para el Sábado)

```
// el número del día para generar el número de un documento HTML a cargar
var menuNumber = today.getDay() % 2;
window.location = "menu" + menuNumber + ".html";
```

▼ Introducción a los objetos JavaScript

3 TIPOS

1. **Tipos básicos** → `Object`, `Array`, `String`, `Date`, `Math`, `RegExp`
2. Los objetos de la **parte cliente/servidor para interactuar** con el navegador y el documento (o con el entorno del servidor)
   - `document`, `history`, `window`, `navigator`
3. Los que **define el programador**

PROPIEDADES Y MÉTODOS

```
//como la notación en Java
window.location / history.back()
```

- Con `new` se crean nuevas instancias

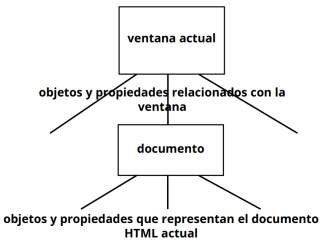
```
vay today = new Date()
```

- Se pueden añadir propiedades dinámicamente

```
var today = new Date()
today.forecast = "rainy"
alert(today.forecast)
```

▼ Objetos en el cliente

> 💡 **HAY UNA JERARQUÍA** para organizar los objetos relacionados con la ventana del navegador y el documento HTML.



No se trata de una jerarquía de herencia → sino de **obejtos que reflejan la estructura de una página HTML.**

VENTANA ACTUAL: Nivel superior

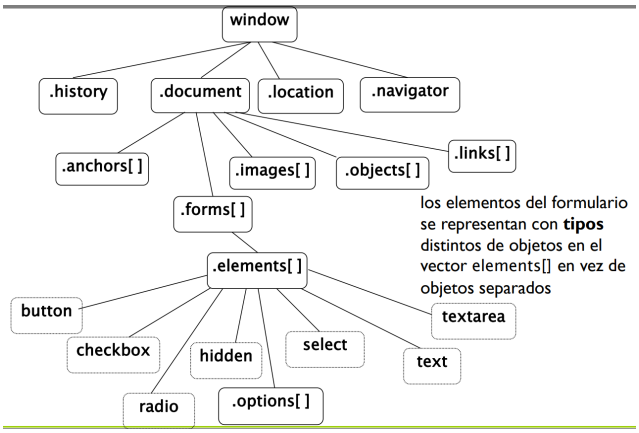
- Los demás objetos del lado del cliente se relacionan directa o indirectamente con ella

DOCUMENTO: Relacionado directamente con la ventana actual.

- Constituye el origen de los demás objetos que representan partes de la página HTML

▼ Modelo de objetos del documento (nivel 0)

ESQUEMA GLOBAL



▼ JERARQUÍA DE OBJETOS

> 💡 **El objeto** `document` en JavaScript representa el **documento HTML actual** en el navegador. A través de él, se puede acceder a varios aspectos del documento y manipular su contenido utilizando el **DOM (Modelo de Objetos del Documento)**.

El objeto `document` es la **raíz de la jerarquía** de objetos en el DOM.

- Representa todo el documento HTML cargado en la ventana o el marco del navegador.
- A través de sus propiedades y métodos, **se pueden acceder a otros objetos** que representan elementos específicos en la página, como elementos `<html>`, `<head>`, `<body>`, etc.
  - `document.images[]`: Colección de todas las imágenes en la página.
  - `document.forms[]`: Colección de todos los formularios en la página.
  - `document.anchors[]`: Colección de todos los enlaces anclados en la página.

El `DOM` se divide en diferentes niveles (como DOM Nivel 1, Nivel 2, etc.).

- Cada nivel agrega nuevas funcionalidades y capacidades para acceder y manipular elementos en la página.
- Los niveles posteriores (como DOM Nivel 2) permiten acceder a más elementos y proporcionan métodos más avanzados.

EJEMPLO CON `elements[]`

> Representa todos los componentes de un formulario

`document.forms[0].elements[3]`:

- `document.forms[0]` representa el primer formulario en el documento actual.
- `elements[3]` se refiere al cuarto elemento dentro de ese formulario.

```
<form action="dummy">
    <h3>There are:</h3>
    <p>
        <input type="text" name="txtDays" size="5" /> Days<br />
        <input type="text" name="txtHours" size="5" /> Hours<br />
        <input type="text" name="txtMins" size="5" /> Minutes<br />
        <input type="text" name="txtSecs" size="5" /> Seconds
    </p>
</form>
```

- `document.forms[0].elements[3]` se referiría al cuarto elemento, que es el campo de entrada para los minutos (`<input type="text" name="txtMins" size="5" />`).

▼ ACCESO UNIVERSAL A PARTES DEL DOCUMENTO

El método `getElementById()` sirve para acceder a elementos específicos.

1. `document.getElementById('foo')`:

- Permite acceder a un elemento HTML según su atributo `id` (`id="foo"`).

Los IDs deben ser **únicos dentro de un documento** → No se pueden repetir en diferentes elementos.

2. **OTROS MÉTODOS de Selección Adicionales**:
   - `getElementsByClassName()`: Devuelve una colección de elementos con una clase específica.
   - `getElementsByTagName()`: Devuelve una colección de elementos con una etiqueta HTML específica (como `<div>`, `<p>`, etc.).

> 💡 Si el método es PLURAL → devuelven **colecciones** o **vectores** de elementos.
>
> - Por ejemplo, `getElementsByClassName('myClass')` devuelve todos los elementos con la clase `myClass`.

El **DOM (Modelo de Objetos del Documento)** no se construye hasta que la página finaliza su carga. Por lo tanto, las técnicas HTML que dependen de la estructura del DOM deben ejecutarse después de que la página esté completamente cargada.

3. `innerHTML`

> Permite controlar el contenido de los **elementos HTML**

Puede contener **código HTML**, lo que facilita la modificación dinámica.

```
document.getElementById('pageTitle').innerHTML = "Shopping days until Xmas"
```

Esto cambiaría el contenido del elemento con el atributo `id` igual a “pageTitle” a “Shopping days until Xmas”.

- También puedes acceder a los **atributos de los elementos**.

  ```
  document.getElementById('update').title = "click here to update the countdown":
  ```

  Esto establecería el atributo `title` del elemento con `id` igual a “update” como “click here to update the countdown”.

- El atributo relativo al **estilo del elemento** (`style`) permite controlar su presentación

  ```
  document.getElementById('clock').style.fontSize = "1.5em"
  ```

  Esto cambiaría el tamaño de fuente del elemento con `id` igual a “clock” a 1.5 veces el tamaño de fuente normal.

▼ ANEXO

EVENTOS MÁS COMUNES

Evento	Descripción
onFocus	usuario mueve el foco al objeto
onBlur	usuario quita el foco del objeto
onSelect	usuario selecciona texto
onChange	usuario cambia el valor de un objeto
onSubmit	usuario envía el formulario
onClick	usuario hace click en un botón o enlace
onMouseOver	usuario pone el cursor sobre un enlace
onMouseOut	usuario mueve el cursor fuera del enlace
onLoad	página termina de cargarse
onUnload	usuario abandona la página
onAbort	usuario cancela la carga de la página
onError	se encuentra un error en el script

CONSULTANDO EL TIPO CON TYPEOF

Variable *a*	typeof *a*
undefined	"undefined"
null	"object"
true / *false*	"boolean"
cualquier número o "*NaN*"	"number"
cualquier cadena	"string"
cualquier función	"function"
cualquier objeto nativo (no función)	"object"
cualquier otro objeto	Cadena que depende de la implementación, salvo "undefined", "boolean", "number" o "string".
undefined	"undefined"