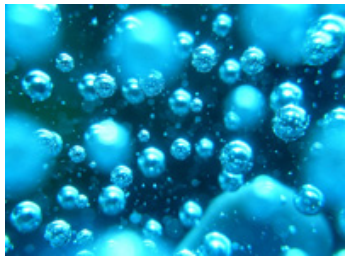
6.A. Modelo de objetos del documento en JavaScript

2.- Gestión de eventos.

2.1.- Modelos de eventos.



Vamos a hacer un pequeño repaso de los modelos de eventos, los cuáles fueron detallados en la unidad 5, apartados 3.1 al 3.4 y la fase de disparo de eventos apartado 3.5.

Veíamos que tenemos **4 modelos de registro de eventos:**

- Modelo de **registro de eventos en línea**:
  - Los eventos se añaden como un atributo más del objeto.
  - No es un modelo recomendado hoy en día, porque el código de JavaScript está integrado con el HTML y lo que se intenta conseguir es tener separación entre la estructura y la programación.
  - Ejemplo: `<A href="pagina.html" onClick="alertar()">Pulsa aqui</a>`
- Modelo de **registro de eventos tradicional**:
  - Los eventos se asignan como una propiedad del objeto y fuera de la estructura HTML.
  - No es un modelo estándar de registro, pero si utilizado ampliamente por Netscape y Microsoft.
  - Uso de la palabra reservada `this`, para hacer referencia al objeto dónde se programó el evento.
  - Para asignar un evento se podría hacer: `elemento.evento = hacerAlgo;`
  - Para eliminar ese evento del objeto: `elemento.evento = null;`
  - Ejemplo: `document.getElementById("mienlace").onclick = alertar;`
- Modelo de **registro avanzado de eventos según W3C**:
  - Es el estándar propuesto por el W3C.
  - Para asignar un evento se usa `addEventListener()`.
  - Para eliminar un evento se usa `removeEventListener()`.
  - Se puede programar cuando queremos que se dispare el evento: en la fase de captura o burbujeo.
  - Uso de la palabra reservada `this`, para hacer referencia al objeto dónde se programó el evento.
  - Por ejemplo: `document.getElementById("mienlace").addEventListener('click',alertar,false);`
- Modelo de **registro de eventos según Microsoft (Internet Explorer)**:
  - Se parece al utilizado por el W3C.
  - Para asignar un evento se usa `attachEvent()`.
  - Para eliminar un evento se usa `detachEvent()`.
  - Aquí los eventos siempre burbujean, no hay forma de captura.
  - No se puede usar la palabra reservada `this/code>, ya que la función es copiada, no referenciada.`
  - El nombre de los eventos comienza por **"on"** + **nombre de evento**.
  - Por ejemplo: `document.getElementById("mienlace").attachEvent('onclick', alertar);`

Y tenemos **3 modelos propuestos de disparo de eventos**, que clarificarán el orden de disparo de los mismos, cuando se solapen eventos sobre elementos anidados:

- Modelo de **captura de eventos:**
  - En este modelo los eventos se van disparando de afuera hacia adentro. Es decir, primero se disparará el evento asignado al elemento exterior, y continúa descendiendo y disparando los eventos que coincidan, hasta llegar al elemento interior.
- Modelo de **burbujeo de eventos**:
  - En este modelo los eventos se van disparando desde dentro hacia afuera. Es decir, primero se disparará el evento asignado al elemento interior, y continúa subiendo y disparando los eventos que coincidan, hasta llegar al elemento exterior.
- Modelo **W3C**:
  - En este modelo se integran los dos modelos anteriores. Simplemente se realiza la fase de captura de eventos primero y, cuando termina, se realiza la fase de burbujeo. En este modelo cuando registramos un evento con `addEventListener(evento, funcion, truelfalse)` tenemos la opción de indicar cuándo queremos que se dispare el evento:
    - en la **fase de** `captura ( , , true)`
    - en la **fase de** `burbujeo ( , , false)`
  - También disponemos de un nuevo método para cancelar eventos con `preventDefault()`, y de un método para detener la propagación de eventos en la fase de burbujeo, con `stopPropagation()`.

◀ Solución a la tarea para DWEC06

Ir a...

6.B. HTML DOM en W3Schools ▶