iesperemariaorts

Desarrollo web en entorno cliente

Tema 2

Javascript

BOM – DOM – Eventos

iesperemariaorts

BOM y DOM

iespereMariaorts

BOM y DOM

- BOM: Browser Object Model
* Objetos y funciones para interactuar con el navegador.

- DOM: Document Object Model
* * Objetos y funciones para interactuar con el HTML.

iesperemariaorts

BOM: Objeto Window

- Representa el navegador y es el objeto principal

- Todo está contenido dentro de window
(variables globales, el objeto document, el objeto location,
el objeto navigation, etc.)

iesperemariaorts

BOM: Timers

- 2 tipos (valores en milisegundos): timeouts e intervals
- setTimeout(función, milisegundos): se ejecuta sólo una vez
- cleartimeout(timeoutId) → Cancela un timeout (antes de ser llamado)

- setInterval(funcion, milisegundo): se ejecuta cada X milisegundos hasta que la cancelemos.
- clearInterval(idIntervalo) → Cancela un intervalo (no se repetirá más)

iesperemariaorts

BOM: Location

- Contiene información sobre la url actual del navegador
- Podemos acceder y modificar dicha url usando este objeto

```
console.log(location.href); // Imprime la URL actual
console.log(location.host); // Imprime el nombre del servidor (o la IP) como "localhost" 192.168.0.34
console.log(location.port); // Imprime el número del puerto (normalmente 80)
console.log(location.protocol); // Imprime el protocolo usado (http ó https)

location.reload(); // Recarga la página actual
location.assign("https://google.com"); // Carga una nueva página. El parámetro es la nueva URL
location.replace("https://google.com"); // Carga una nueva página sin guardar la actual en el objeto history
```

- Para navegar a través de las páginas que hemos visitado en la pestaña actual, podemos usar el objeto history.

```
console.log(history.length); // Imprime el número de páginas almacenadas

history.back(); // Vuelve a la página anterior
history.forward(); // Va hacia la siguiente página
history.go(2); // Va dos páginas adelante (-2 iría dos páginas hacia atrás)
```

iesperemariaorts

BOM: Diálogos

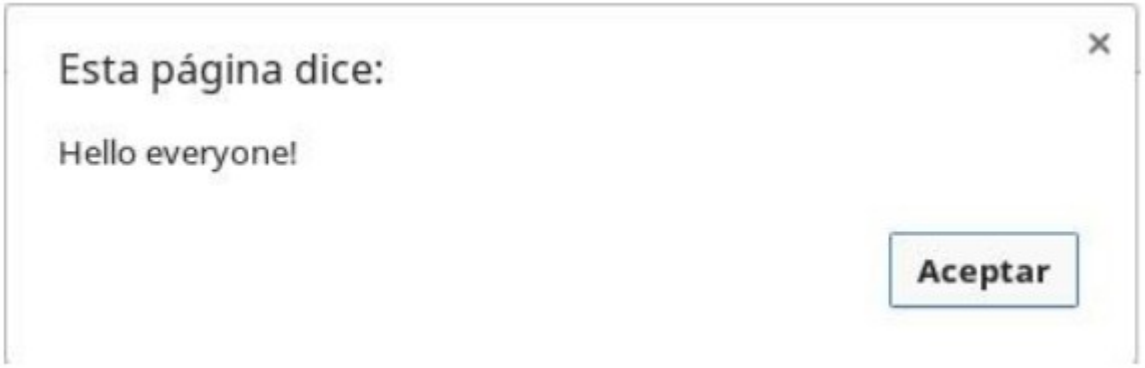
- En cada navegador, tenemos un conjunto de diálogos para interactuar.

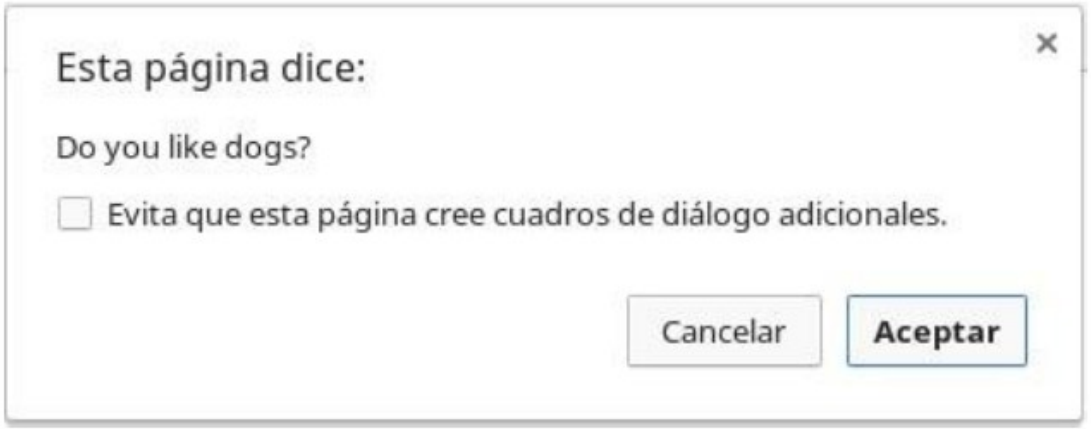
- No son personalizables ni uniformes.

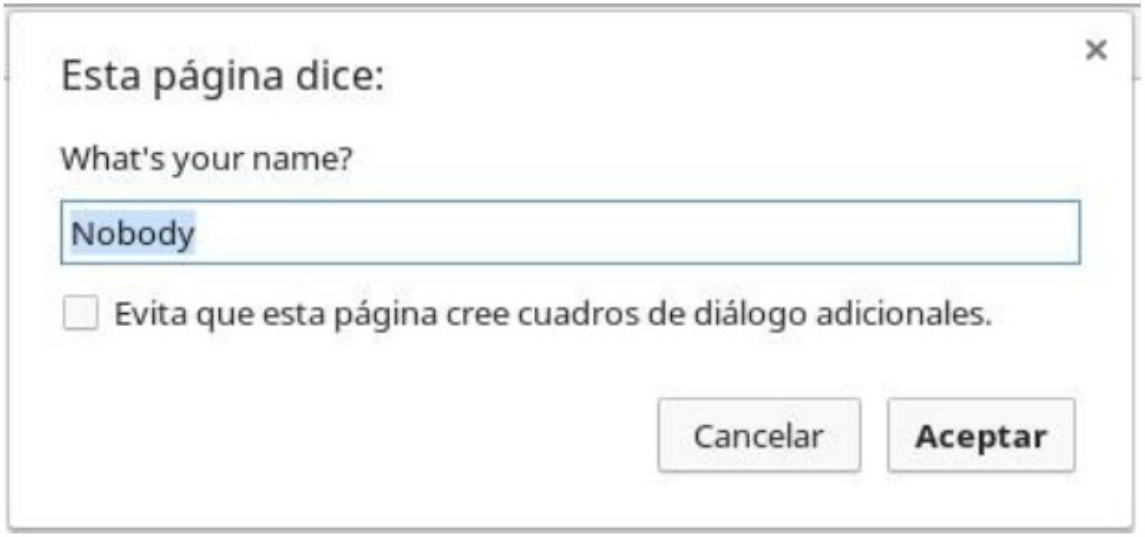
- No se recomienda usar en producción.

- 3 tipos: alert, confirm y prompt.

iesperemariaorts







iesperemariaorts

DOM

iesperemariaorts

DOM

- Es una estructura en árbol que contiene una representación de todos los nodos del HTML incluyendo sus atributos.

- En este árbol, todo se representa como nodo y podemos añadir, eliminar o modificarlos.

- Objeto principal: document

- Cada nodo HTML dentro de document es un objeto "element" con otros nodos con atributos y estilos

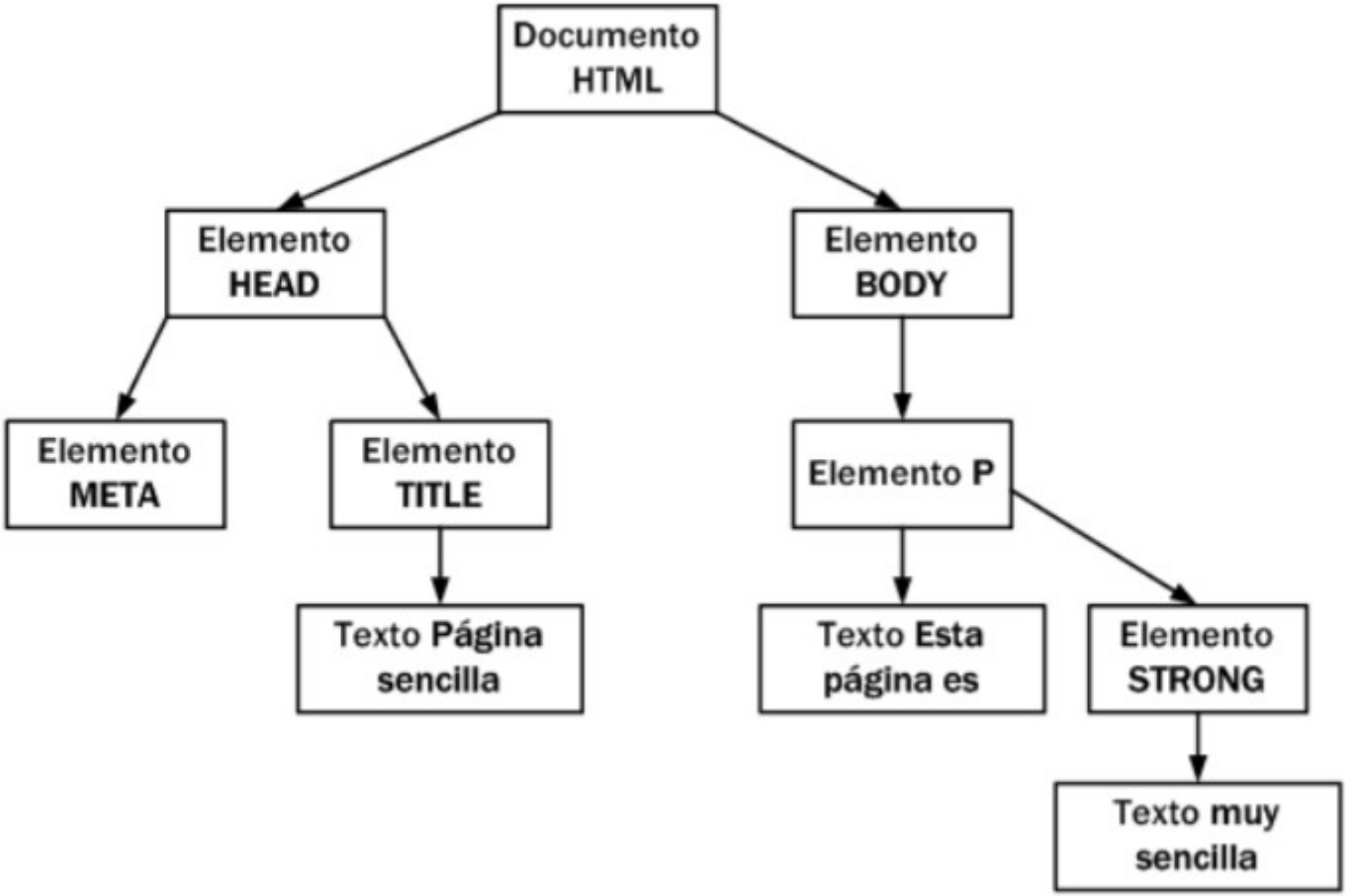
iesperemariaorts

DOM

```
<!DOCTYPE html>
        <html lang="es">
            <head>
                <meta charset="utf-8" />
                <title>Página sencilla</title>
            </head>
            <body>
                <p>Esta página es <strong>muy
                sencilla</strong></p>
            </body>
        </html>
```

iesperemariaorts

DOM



iesperemariaorts

DOM

- Cada árbol: nodo superior llamado document.
- Cada nodo (excepto raíz) tiene un padre.
- Cada nodo puede tener X hijos.
- Hoja: Nodo sin hijos.
- Nodos con mismo padre: hermanos.

- Document: Nodo raíz.
- Element: Representa cada una de las etiquetas. Puede contener atributos y nodos hijos.
- Attr: Nombre del valor o atributo.
- Text: Almacena información contenida en element.
- Coment: Representa comentario HMTL.

iesperemariaorts

DOM: Navegando por el DOM

- Una vez construido arbol DOM utilizamos funciones DOM para acceder de forma directa a nodos, o lo que es lo mismo a "un trozo" de la página.

- Solo es posible al haberse construido roralmente el árbol DOM (página completamente cargada).

- Dos metodos de acceso:
* Acceso a través de nodos padre: No es útil.
* Acceso directo a los nodos: Mucho más rápido y efectivo.

iesperemariaorts

DOM: Navegando por el DOM

- document.documentElement → Devuelve el elemento <html>
- document.head → Devuelve el elemento <head>
- document.body → Devuelve el elemento <body>
- document.getElementById("id") → Devuelve el elemento que tiene el id especificado, o null si no existe.
- document.getElementsByClassName("class") → Devuelve un array de elementos que tengan la clase especificada. Al llamar a este método desde un nodo (en lugar de document), buscará los elementos a partir de dicho nodo.
- document.getElementsByTagName("HTML tag") → Devuelve un array con los elementos con la etiqueta HTML especificada. Por ejemplo "p" (párrafos).
- element.childNodes → Devuelve un array con los descendientes (hijos) del nodo. Esto incluye los nodos de tipo texto y comentarios.
- element.chidren → Igual que arriba pero excluye los comentarios y las etiquetas de texto (sólo nodos HTML). Normalmente es el recomendado.
- element.parentNode → Devuelve el nodo padre de un elemento.
- element.nextSibling → Devuelve el siguiente nodo del mismo nivel (el

iesperemariaorts

DOM: querySelector

- Acceder a los elementos HTML basándose en selectores CSS (clase, id, atributos,….)

- document.querySelector("selector") → Devuelve el primer elemento que coincide con el selector

- document.querySelectorAll("selector") → Devuelve una colección con todos los elementos que coinciden con el selector

---

a → Elementos con la etiqueta HTML <a>
.class → Elementos con la clase "class"
#id → Elementos con el id "id"
.class1.class2 → Elementos que tienen ambas clases, "class1" y "class2"
.class1,.class2 → Elementos que contienen o la clase "class1", o "class2"
.class1 p → Elementos <p> dentro de elementos con la clase "class1"
.class1 > p → Elementos <p> que son hijos inmediatos con la clase "class1"
#id + p → Elemento <p> que va después (siguiente hermano) de un elemento que tiene el id "id"
#id ~ p → Elementos que son párrafos <p> y hermanos de un elemento con el id "id"
.class[attrib] → Elementos con la clase "class" y un atributo llamado "attrib"

iesperemariaorts

DOM: Manipulando el DOM

- document.createElement("tag") → Crea un elemento HTML. Todavía no estará en el DOM, hasta que lo insertemos (usando appendChild, por ejemplo) dentro de otro elemento del DOM.
- document.createTextNode("text") → Crea un nodo de texto que podemos introducir dentro de un elemento. Equivale a element.innerText = "texto".
- element.appendChild(childElement) → Añade un nuevo elemento hijo al final del elemento padre.
- element.insertBefore(newChildElement, childElem) → Inserta un nuevo elemento hijo antes del elemento hijo recibido como segundo parámetro.
- element.removeChild(childElement) → Elimina el nodo hijo que recibe por parámetro.
- element.replaceChild(newChildElem, oldChildElem) → Reemplaza un nodo hijo con un nuevo nodo.

iesperemariaorts

DOM: Ejemplo creación nodos

- Como hemos visto necesitamos dos nodos.

1º nodo tipo Element, 2º tipo Text.

Cuatro pasos:

- Creación nodo tipo Element: representa al elemento.
- Creación nodo tipo Text: Representa al contenido.
- Añadir el nodo Text como hijo de Element.
- Añadir nodo Element a la página en forma de nodo hijo

iesperemariaorts

DOM: Ejemplo creación nodos

```
// Crear nodo de tipo Element
var parrafo = document.createElement("p");

// Crear nodo de tipo Text
var contenido = document.createTextNode("Hola Mundo!");

// Añadir el nodo Text como hijo del nodo Element
parrafo.appendChild(contenido);

// Añadir el nodo Element como hijo de la pagina
document.body.appendChild(parrafo);
```

iesperemariaorts

DOM: Manipulando el DOM

```
let ul = document.getElementsByTagName("ul")[0]; // Obtiene la primera lista (ul)
let li3 = ul.children[2]; // Tercer elemento de la lista (li)
let newLi3 = document.createElement("li"); // Crea un nuevo elemento de lista
newLi3.innerText = "Now I'm the third element"; // Y le asigna un texto

ul.insertBefore(newLi3, li3); // Ahora li3 es el cuarto elemento de la lista (newLi3 se inserta antes)
li3. innerText = "I'm the fourth element..."; // Cambiamos el texto para reflejar que es el cuarto elemento
```

El HTML resultante será:

```
<ul>
   <li id="firstListElement">Element 1</li>
   <li>Element 2</li>
   <li>Now I'm the third element</li>
   <li>I'm the fourth element...</li>
</ul>
```

iesperemariaorts

Atributos

- Dentro de elementos HTML hay atributos (name, id, href...)
- Cada atributo tiene nombre (name) y valor (value)
- Pueden ser leído y/o modificado

element.attributes → Devuelve el array con los atributos de un elemento

element.className → Se usa para acceder (leer o cambiar) al atributo class. Otros atributos a los que se puede acceder directamente son: element.id, element.title, element.style (propiedades CSS), ...

element.classList → Array de clases CSS del elemento. A diferencia de className, que es una cadena con las clases separadas por espacio, este

element.hasAttribute("attrName") → Devuelve cierto si el elemento tiene un atributo con el nombre especificado

element.getAttribute("attrName") → Devuelve el valor del atributo

element.setAttribute("attrName", "newValue") → Cambia el valor

File: example1.html

```
<!DOCTYPE>
<html>
   <head>
      <title>JS Example</title>
   </head>
   <body>
      <p><a id="toGoogle" href="https://google.es" class="normalLink">Google</a></p>
      <script src="./example1.js"></script>
   </body>
</html>
```

File: example1.js

```
let link = document.getElementById("toGoogle");
link.className = "specialLink"; // Equivale a: link.setAttribute("class", "specialLink");
link.setAttribute("href", "https://twitter.com");
link.textContent = "Twitter";
if(!link.hasAttribute("title")) { // Si no tenía el atributo title, establecemos uno
   link.title = "Ahora voy aTwitter!";
}
```

File: example1.html

```
<!DOCTYPE>
<html>
   <head>
      <title>JS Example</title>
   </head>
   <body>
      <p><a id="toGoogle" href="https://google.es" class="normalLink">Google</a></p>
      <script src="./example1.js"></script>
   </body>
</html>
```

File: example1.js

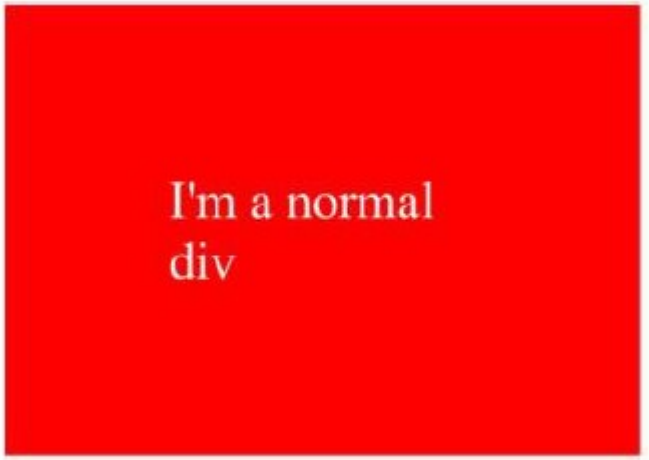
```
let link = document.getElementById("toGoogle");
link.className = "specialLink"; // Equivale a: link.setAttribute("class", "specialLink");
link.setAttribute("href", "https://twitter.com");
link.textContent = "Twitter";
if(!link.hasAttribute("title")) { // Si no tenía el atributo title, establecemos uno
   link.title = "Ahora voy aTwitter!";
}
```

```
<a id="toGoogle" href="https://twitter.com" class="specialLink"
   title="Ahora voy a Twitter!">Twitter</a> */
```

iesperemariaorts

Atributo style

```
<!DOCTYPE>
<html>
   <head>
      <title>JS Example</title>
   </head>
   <body>
      <div id="normalDiv">I'm a normal div</div>
      <script src="./example1.js"></script>
   </body>
</html>
```



File: example1.js

```
let div = document.getElementById("normalDiv");
div.style.boxSizing = "border-box";
div.style.maxWidth = "200px";
div.style.padding = "50px";
div.style.color = "white";
div.style.backgroundColor = "red";
```

iesperemariaorts

Eventos

- Nuestro código debe ser capaz de manejar eventos de forma adecuada.

- Existen muchos eventos que pueden ser capturados y manejados:
Web Eventos

- Eventos de la página:

- load → Este evento se lanza cuando el documento HTML ha terminado de cargarse. Es útil para realizar acciones que requieran que el DOM haya sido completamente cargado (como consultar o modificar el DOM).

- unload → Ocurre cuando el documento es destruido, por ejemplo, después de cerrar la pestaña donde la página estaba cargada.

- beforeunload → Ocurre justo antes de cerrar la página. Por defecto, un mensaje pregunta al usuario si quiere realmente salir de la página, pero hay otras acciones que pueden ser ejecutadas.

- resize → Este evento se lanza cuando el tamaño del documento cambia (normalmente se usa si la ventana se redimensiona)

iesperemariaorts

Eventos del teclado

- keydown → El usuario presiona una tecla. Si la tecla se mantiene pulsada durante un tiempo, este evento se generará de forma repetida.
- keyup → Se lanza cuando el usuario deja de presionar la tecla
- keypress → Más o menos lo mismo que keydown. Acción de pulsar y levantar.

Eventos del ratón

- click → Este evento ocurre cuando el usuario pulsa un elemento (presiona y levanta el dedo del botón → mousedown + mouseup). También normalmente se lanza cuando un evento táctil de toque (tap) es recibido.
- dblclick → Se lanza cuando se hace un doble click sobre el elemento
- mousedown → Este evento ocurre cuando el usuario presiona un botón del ratón
- mouseup → Este evento ocurre cuando el usuario levanta el dedo del botón del ratón
- mouseenter → Se lanza cuando el puntero del ratón entra en un elemento
- mouseleave → Se lanza cuando el puntero del ratón sale de un elemento
- mousemove → Este evento se llama repetidamente cuando el puntero de un ratón se mueve mientras está dentro de un elemento

iesperemariaorts

Eventos

Eventos de formulario

- focus → Este evento se ejecuta cuando un elemento (no sólo un elemento de un formulario) tiene el foco (es seleccionado o está activo).
- blur → Se ejecuta cuando un elemento pierde el foco.
- change → Se ejecuta cuando el contenido, selección o estado del checkbox de un elemento cambia (sólo <input>, <select>, y <textarea>)
- input → Este evento se produce cuando el valor de un elemento <input> o <textarea> cambia.
- select → Este evento se lanza cuando el usuario selecciona un texto de un <input> o <textarea>.
- submit → Se ejecuta cuando un formulario es enviado (el envío puede ser cancelado).

iesperemariaorts

Eventos: Manejo

- Hay muchas formas de asignar un código o función a un determinado evento.

- Vamos a ver las dos formas posibles, pero el recomendado (y la forma válida para este curso) es usar event listeners.

- Ejemplo clásico (no recomendado):

Archivo: ejemplo1.html

```
<input type="text" id="input1" onclick="inputClick(this, event)" />
```

Archivo: ejemplo1.js

```
function inputClick(element, event) {
   // Mostrará "Un evento click ha sido detectado en #input1"
   alert("Un evento" + event.type + " ha sido detectado en #" + element.id);
}
```

iesperemariaorts

Eventos: Manejo

- Event listeners (recomendado)

- Añadimos método addEventListener sobre el elemento. Este método recibe al menos dos parámetros. El nombre del evento (una cadena) y un manejador (función anónima o nombre de una función existente).

Archivo: ejemplo1.html

```
<input type="text" id="input1" />
```

Archivo: ejemplo1.js

```
let input = document.getElementById("input1");
input.addEventListener('click', function(event) {
   alert("Un evento " + event.type + " ha sido detectado en " + this.id);
});
```

- Podemos añadir tantos manejadores como queramos.

```
input.addEventListener('click', inputClick);
input.addEventListener('click', inputClick2);
```

iesperemariaorts

Eventos: Objeto del evento

- Creado por JavaScript y pasado al manejador como parámetro.

- Tiene algunas propiedades generales:

- target → El elemento que lanza el evento (si fue pulsado, etc...).
- type → El nombre del evento: 'click', 'keypress', ...
- cancelable → Devuelve true o false. Si el evento se puede cancelar significa que llamando a event.preventDefault() se puede anular la acción por defecto (El envío de un formulario, el click de un link, etc...).
- bubbles → Devuelve cierto o falso dependiendo de si el evento se está propagando (Lo veremos más adelante).
- preventDefault() → Este método previene el comportamiento por defecto (cargar una página cuando se pulsa un enlace, el envío de un formulario, etc.)
- stopPropagation() → Previene la propagación del evento.

iesperemariaorts

Eventos: Objeto del evento

- Propiedades específicas (dependiendo del tipo de método):

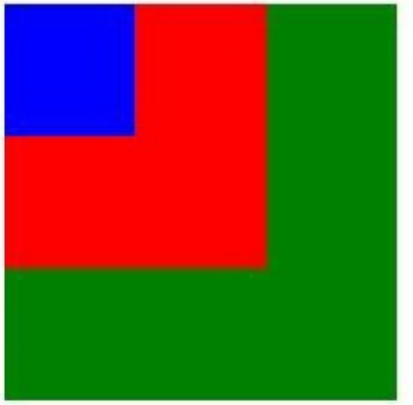
MouseEvent

- button → Devuelve el botón del ratón que lo ha pulsado (0: botón izquierdo, 1: la rueda del ratón, 2: botón derecho).
- clientX, clientY → Coordenadas relativas del ratón en la ventana del navegador cuando el evento fue lanzado.
- pageX, pageY → Coordenadas relativas del documento HTML, si se ha realizado algún tipo de desplazamiento (scroll), este será añadido (usando clientX y clientY no se añade).
- screenX, screenY → Coordenadas absolutas del ratón en la pantalla.
- detail → Indica cuántas veces el botón del ratón ha sido pulsado (un click, doble, o triple click).

Eventos: Propagación de eventos

Archivo: ejemplo1.html

```
<div id="div1" style="background-color: green; width: 150px; height: 150px;">
      <div id="div2" style="background-color: red; width: 100px; height: 100px;">
          <div id="div3" style="background-color: blue; width: 50px; height: 50px;"></div>
      </div>
 </div>
```

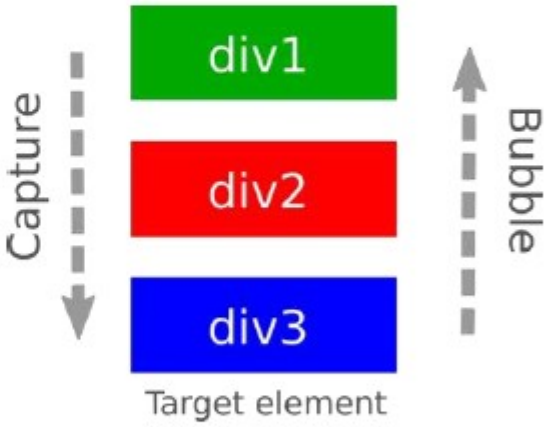


Archivo: ejemplo1.js

```
let divClick = function(event) {
   // eventPhase: 1 -> capture, 2 -> target (objetivo), 3 -> bubble
   console.log("Has pulsado: " + this.id + ". Fase: " + event.eventPhase);
};

let div1 = document.getElementById("div1");
let div2 = document.getElementById("div2");
let div3 = document.getElementById("div3");

div1.addEventListener('click', divClick);
div2.addEventListener('click', divClick);
div3.addEventListener('click', divClick);
```



Por defecto, cuando pulsamos el azul (div3), imprime:

Has pulsado: div3. Fase: 2 → Target element
Has pulsado: div2. Fase: 3 → Bubbling
Has pulsado: div1. Fase: 3 → Bubbling

iesperemariaorts

Formularios: obtener valores

- Debemos capturar el evento submit del mismo (más recomendable que capturar click del botón).

```
const form = document.getElementById('formulario');

form.addEventListener('submit', e => {
    e.preventDefault();
    ...
});
```

iesperemariaorts

Formularios: obtener valores

- Obtener valores de un campo de texto:

```
<input type="text" id="nombre" name="nombre">

form.nombre.value
```

iesperemariaorts

Formularios: obtener valores

- Obtener valores de un checkbox, tendremos colección de inputs:

```
<input type="checkbox" id="deporte" name="aficiones" value="deporte">
<label for="deporte">Deporte</label>
<input type="checkbox" id="viajar" name="aficiones" value="viajar">
<label for="viajar">Viajar</label>
<input type="checkbox" id="comer" name="aficiones" value="comer">
<label for="comer">Comer</label>
```

```
const aficiones = Array.from(form.aficiones)
    .filter(input => input.checked)
    .map(input => input.value);
```

iesperemariaorts

Formularios: obtener valores

- Obtener valores de un radio, tendremos colección de inputs y elegir solo el seleccionado (value):

```
<input type="radio" id="rojo" name="color" value="rojo" checked>
<label for="rojo">Rojo</label>
<input type="radio" id="verde" name="color" value="verde">
<label for="verde">Verde</label>
<input type="radio" id="azul" name="color" value="azul">
<label for="azul">Azul</label>

form.color.value
```