
GEEKSHUBS
Academy_
FULL STACK
DEVELOPER
BOOTCAMP
Desarrollamos
{ talento }

GEEKSHUBS Academy_

HTML

HyperText Markup Language

Lenguaje de marcas de hipertexto: hace referencia al lenguaje de marcado para la elaboración de páginas web.

- Extensión de archivo: html, htm, xhtml y xht
- Tipo de MIME: text/html y application/xhtml+xml
- Type code: TEXT
- Uniform Type Identifier: index.html
- Lanzamiento inicial: 1993
- Tipo de formato Lenguaje de marcado
- Extendido de SGML
- Extendido a XHTML
- Estándar(es) ISO/IEC 15445
  - W3C HTML 4.01
  - W3C HTML5
  - W3C HTML5.1
  - W3C HTML5.2

Un poco de história

1989: Primeros pasos

Se presenta una propuesta de gestión de la información internamente

CERN

1991: Nace la WWW

Nace la World Wide Web

Tim Berner-Lee

GEEKSHUBS Academy_

Desarrolló los pilares de la WWW

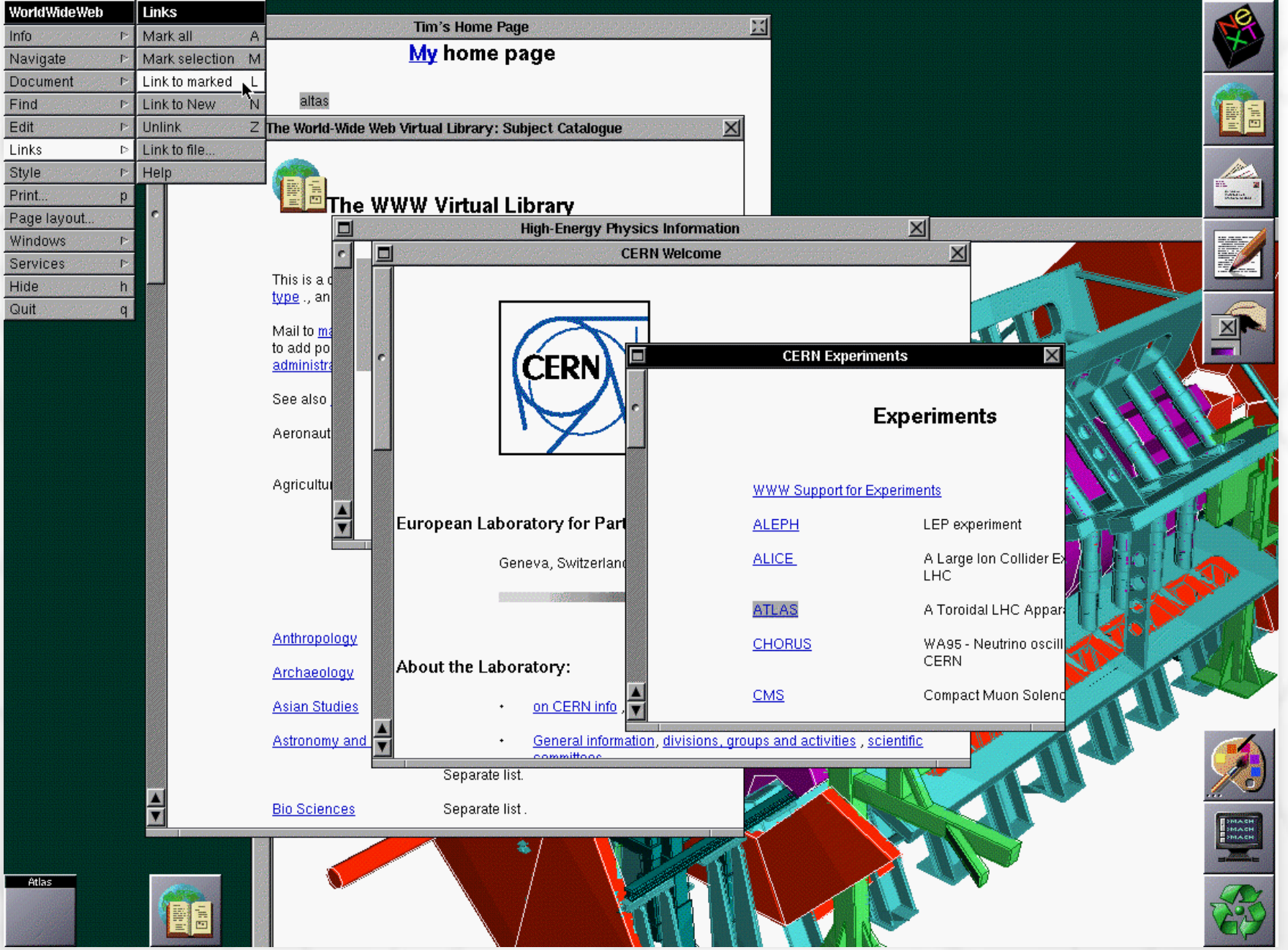
- El lenguaje HTML
- Las direcciones URI
- Protocolo de transferencia HTTP

1993 se inicia la expansión

El CERN libera su código

Aparecen las primeras páginas web

Eran sencillas y contenían hipervículos.


WorldWideWeb
Info
Navigate
Document
Find
Edit
Links
Style
Print... p
Page layout...
Windows
Services
Hide h
Quit q
Links
Mark all A
Mark selection M
Link to marked L
Link to New N
Unlink Z
Link to file...
Help
Tim's Home Page
My home page
altas
The World-Wide Web Virtual Library: Subject Catalogue
The WWW Virtual Library
High-Energy Physics Information
CERN Welcome
CERN
European Laboratory for Part
Geneva, Switzerlan
About the Laboratory:
on CERN info
General information, divisions, groups and activities, scientific
CERN Experiments
Experiments
WWW Support for Experiments
ALEPH
LEP experiment
ALICE
A Large Ion Collider Ex
LHC
ATLAS
A Toroidal LHC Appar
CHORUS
WA95 - Neutrino oscill
CERN
CMS
Compact Muon Solenc
Anthropology
Archaeology
Asian Studies
Astronomy and
Separate list.
Bio Sciences
Separate list .
Atlas

1997 Primer Estándar

HTML 4.0 se publicó como una recomendación del W3C.

2006 - 2007 HTML5

En 2006, el W3C se interesó en el desarrollo de HTML5, y en 2007 se unió al grupo de trabajo del WHATWG para unificar el proyecto.

GEEKSHUBS Academy_

Qué podemos hacer hoy en día

Lo que el entorno de ejecución y html5 nos permite:

https://whatwebcando.today/

Que tecnologías podemos aplicar.

- WebApps.
- Progressive Web Apps.
- Aplicaciones Híbridas.
- Aplicaciones Escritorio.

GEEKSHUBS Academy_

Sintaxis

Que es una etiqueta

Es el componente básico de un documento HTML

Se crean en forma de definiciones con un nombre

```
<etiqueta>Contenido</etiqueta>

<etiqueta/>
```

Que son los atributos

Son las características que definen el comportamiento especifico para una etiqueta dada

No siguen reglas para su ordenamiento y varían de acuerdo a la etiqueta que se aplican

```
<etiqueta atributo="valor">Contenido</etiqueta>
```

Estructura básica de un documento html

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Ejemplo1</title>
  </head>
  <body>
    <p>Párrafo de ejemplo</p>
  </body>
</html>
```

Estructura básica de un documento HTML

Todos los documentos Html tienen la estructura que se muestra a continuación.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8"/>
        <title>Título de la Página
        </title>
    </head>
    <body>
    Aquí va el contenido de la página
    </body>
</html>
```

Doctype

```
<!DOCTYPE html>
```

Sirve para identificar la versión de html a utilizar

HTML

```
<html>
```

Indica al navegador que se trata de un documento HTML

HEAD

```
<head>
```

En el head habitualmente se introduce todos los elementos de precarga

- title
- style
- base
- meta
- script

META CHARSET

```
<meta charset=UTF-8>
```

Índica el tipo de codificacicón

Uso de ñ, tildes, símbolos, etc.

TITLE

```
<title>
```

Índica el título de la página

Define un titulo que aparecerá en la barra del explorador

Será el título que aparecera en la barra de favoritos

Será el título que aparecerá en los buscadores y se indexará

BODY

```
<body>
```

Dentro de este tag se incluirá el cuerpo del documento

El mismo contendrá elementos como imagenes, listas, etc.

Titulos

Los titulos se describen con la etiqueta h

```
<h1>Título 1</h1> <h2>Título 2</h2> <h3>Título 3</h3> <h4>Título 4</h4> <h5
```

Parrafos

Los parrafos se escriben con la etiqueta p

```
<p>Este es nuestro primer parrafo</p>
<p>Este es nuestro segundo parrafo</p>
<p>Este es nuestro tercer parrafo</p>
```

Links

```
<a href="">

<a href="https://www.google.com">Vamos a Google</a>
<a href="./segunda.html">Segunda pagina</a>
<a href="mailto:">Enviar correo </a>
```

Atributo target

```
<a href="https://www.google.com" target="_blank">Vamos a Google</a>
<a href="https://www.google.com" target="_self">Vamos a Google</a>
```

Etiqueta de imagen

La etiqueta <img\> nos permite añadir imágenes a nuestra web

```
<img src="http://marvelipsum.com/200/200/random"/>
```

A la vez podemos utilizar los atributos <HEIGHT\> <WIDTH\>

para modificar la anchura y la altura de la imagen

Listas Desordenadas

Se utilizan la etiqueta `<ul><il></il></ul>`

```
<ul>
<il> 1° Elemento </il>
<il> <a href="www.google.es">2° Elemento a google</a> </il>
</ul>
```

Listas Ordenadas

Se utilizan la etiqueta `<ol><il></il></ol>`

```
<ol>
<il> 1° Elemento </il>
<il> <a href="www.google.es">2° Elemento a google</a> </il>
</ol>
```

Etiquetas y atributos para dar formato

```
<body bgcolor="#cccc" background="imagen">
<font color="red"><h1>Titulo</h1></font>
<p>Es una <strong>prueba</strong></p>
<p>Lanzando es un <em>prueba</em></p>
<p align="center">Es otro parrafo centrado</p>
```

Etiquetas propias HTML5

Etiqueta Header

```
<header>
```

Etiqueta para definir la cabecera de la estructura de la página. Aquí pondremos el nombre del sitio , logotipo etc.

Etiquetas Section y Article

```
<section>
   <article>
          Titulo
          parrafo.
```

Etiqueta Aside

```
<aside>
  <h3> Articulo
  <p> Parrafo
```

Etiqueta para definir las barras laterales.

Etiqueta Footer

```
<footer>
 <p>Creado por ...
```

Etiqueta para definir el footer de nuestra página.

Ejemplo HTML5

```
<!DOCTYPE html>

<html lang="es">

<head>
    <title>Titulo de la web</title>
    <meta charset="utf-8" />
</head>
<body>
    <header>
       <h1>Mi sitio web</h1>
       <p>Mi sitio web creado en html5</p>
    </header>
    <section>
       <article>
           <h2>Titulo de contenido<h2>
           <p> Contenido (ademas de imagenes, citas, videos etc.) </p>
       </article>
    </section>
    <aside>
       <h3>Titulo de contenido</h3>
           <p>contenido</p>
    </aside>
    <footer>
        Creado por mi el 2017
    </footer>
```

GEEKSHUBS Academy_

DOM y BOM

Que es el DOM

DOM o Document Object Model es un conjunto de utilidades específicamente diseñadas para manipular documentos XML. Por extensión, DOM también se puede utilizar para manipular documentos XHTML y HTML. Técnicamente, DOM es una API de funciones que se pueden utilizar para manipular las páginas XHTML de forma rápida y eficiente.


GEEKSHUBS
Academy_

Tipos de nodos

- **Document:** Es el nodo raíz de todos los documentos
- **DocumentType:** Es el nodo que contiene la representación del DTD
- **Element:** Representa el contendio defindio por los tags del html
- **Attr:** Representa el par nombre-de-atributo y valor
- **Text:** Almacena el contenido del texto ue se encuentra en una etiqueta
- **CDataSection:** es el nodo que representa una sección de tipo
- **Comment:** resepresenta un comentario

Acceso al DOM

Desde la consola

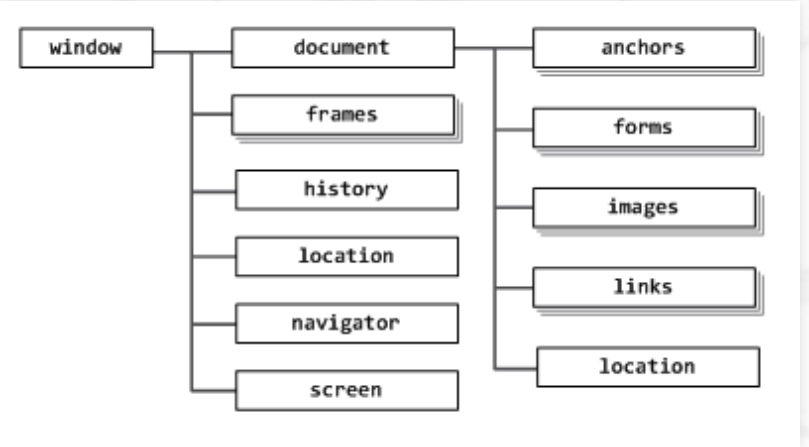
```
document.querySelector('title');
```

Vamos añadir un h1

```
var h1 = document.createElement('h1');
h1.appendChild(document.createTextNode("Prueba"));
document.body.appendChild(h1);
```

Acceso al BOM

Las versiones 3.0 de los navegadores Internet Explorer y Netscape Navigator introdujeron el concepto de Browser Object Model o BOM, que permite acceder y modificar las propiedades de las ventanas del propio navegador.



El objeto window representa la ventana completa del navegador, con el BOM podemos para manipular

```
myWindow = window.open('', '', 'width=200, height=100');
myWindow.document.write("Hola Mundo'");
myWindow.moveBy(250, 250);
myWindow.focus();
```

GEEKSHUBS Academy_

HTML5 API

GEEKSHUBS Academy_

LOCALSTORAGE

LOCALSTORAGE

El almacenamiento local es fundamental para poder disponer de información neesaria para la ejecución de nuestra App, antiguamente el almacenamiento sólo se podía hace en el servidor, o mediante coockies de sesión disponemos de la específicación de WebStorage

HTML5 no permite almacenar en cuatro espacios distintos.

- Web Storage
- Web SQL DataBase
- IndexedDB
- FileStorage

LocalStorage Web Storage

Es el sistema de almacenamiento mas simple ya que los datos son almacenados en parejas de clave/valor y es soportado por la mayoría de los navegadores.

Disponemos de dos tipos:

- localstorage
- sesionstorage

LocalStorage

El almacenamiento local por su parte, funciona de la misma forma que el almacenamiento de sesión, con la excepción de que son capaces de almacenar los datos por dominio y persistir más allá de la sesión actual, aunque el navegador se cierre o el dispositivo se reinicie.

SessionStorage

El almacenamiento local por su parte, funciona de la misma forma que el almacenamiento de sesión, con la excepción de que son capaces de almacenar los datos por dominio y persistir más allá de la sesión actual, aunque el navegador se cierre o el dispositivo se reinicie.

Almacenamiento

El proceso de almacenamiento tanto para LocalStorage como SessionStorage es identico símplemente hay que tener en cuenta que se alamcena un string

```
sessionStorage.setItem('twitter', '@starkyhach');
localStorage.setItem('LinkedIn','@jaroso78');
```

Almacenamiento

Como recuperar datos mediante **getItem** simplemente hay que tener en cuenta que el método nos devuelve siempre un STRING

```
let valor = localStorage.getItem('LinkedIn');
alert('Valor ->' + valor + ' tipo ' typeof(valor));
```

Almaenamiento

Podemos almacenar también objetos literales convertiéndolos en **JSON**

```
let persona ={
    nombre: 'Xavi',
    apellidos: 'Rodriguez'
}
//Almacenamos utilizando la conversión a texto el objeto JSON
localStorage.setItem('Datos', JSON.stringify(persona));
//Recuperamos realizando la conversión de texto a Objeto JSON.
let resultado = JSON.parse(localStorage.getItem('Datos');
alert('Resultado ->' + resultado.nombre + ' Tipo -> ' + typeof(resultado));
```

Almacenamiento

A la vez podemos eliminar la informacion almacenada mediante el uso de:

- removeItem
- Clear

```
//Eliminar un elemento determinado por su clave.
localStorage.removeItem('Datos');
//Para eliminar todo el localStorage o el sessionStorage.
localStorage.clear();
```

Almacenamiento

Web Storage dispone también de eventos que podemos detectarlos y que contienen toda la información asociado a los cambios de datos:

```
StorageEvent {
    readonly DOMString key;
    readonly any oldValue;
    readonly any newValue;
    readonly DOMString url;
    readonly Storage storageArea
}
```

Almacenamiento

Estos eventos se recogen dentro del objeto global Window

```
function handleStorage(event) {
    event = event || window.event; // support IE8
    if (event.newValue === null) { // it was removed
        // Do somthing
    } else {
        // Do somthing else
    }
}
```

GEEKSHUBS Academy_

DRAG & DROP

DRAG & DROP

Una de las novedades que incluye la API DE HTML5 es la funcionalidad de **DRAG & DROP** de forma nativa sin tener que utilizar bibliotecas como JQuery, Dojo u otros frameworks de front-end

En la nueva HMTL5 se define como un mecanismo basado en eventos, donde identificamos los elementos que deseamos arrastrar con el atributo **draggable**

DRAG & DROP

Por defecto, son arrastables los siguientes elementos:

- Enlaces
- Imágenes
- Nodos de texto
- Seleccion de textos

DRAG & DROP

Para hacer un elemento que sea draggable sólo hay que establecer el atributo **draggable=true**

```
<img src="" draggable="true"\>
```

DRAG & DROP

Existen diversos eventos que son lanzados tanto por los elementos de origen como los elementos de destino

- ondragstart: Se dispara cuando empieza el evento (al arrastrar el elemento)
- ondrag: En todo momento mientras el elemento se está moviendo
- ondragenter: Se dispara cuando un elemento que está siendo arrastrado entra en un contendor receptor. Quien dispara dicho evento es el contenedor receptor
- ondragleave: Se dispara en el contenedor receptor cuando un elemento draggable sale de él
- ondragover: Se dispara cuando un draggable está encima, se dispara en el receptor. Como el comportamiento por defecto es denegar el drop, la función debe retornar el valor fale(false) o llamar al metodo prevendefault() para poder soltar el elemento
- ondrop: El elemento arrastrado se acaba soltando en un receptor. Se dispara en el receptor
- ondragend: Nos indica si el elemento draggable se ha soltado, se llama al finalizar el evento

DRAG & DROP

Ahora debemos añadir a los elementos establecidos como draggable un gestor de eventos mediante JavaScript, para ello creamos una función que recoja el evento:

```
function drag(ev){
    ev.dataTransfer.setData("text", ev.target.id);
}
```

DRAG & DROP

Ahora añadimos el evento en el elemento al que se lo queramos aplicar, mediante el atributo **ondragstart**.

```
<img id="imagen1" ondragstart="drag(event)" draggable="true" src="126.jpg">
```

DRAG & DROP

Debemos establecer el contenedor que va a recibir el elemento que queremos mover y añadimos el atributo **ondrop**

```
<div ondrop="drop(event)" ondragover="allowDrop(event)" id="contenedor1"></
```

DRAG & DROP

Ahora para poder dejar la imágen debemos evitar el comportamiento normal del objeto mediante **ondragover**

```
function allowDrop(ev) {
      ev.preventDefault();
  }
```

DRAG & DROP

Añadimos la imágen en el contenedor utilizando javascript.

```
function drop(ev){
    ev.preventDefault();
    var data = ev.dataTransfer.getData("text");
    ev.target.appendChild(document.getElementById(data));
  }
```

Geolocalización.

Geolocalización.

Con html5 podemos utilizar la **Api de Geolocalización** que permite al usuario compartir su ubicación a las aplicaciones web. Simplemente por motivos de privacidad, al usuario se le pide que confirme el permiso para proporcionar información de ubicación.

Geolocalización.

En la actualidad existen diversos métodos para establecer la geolocalización:

- Vía IP: Todo dispositivo conectado a la red, tiene asignado una dirección IP pública que actúa, y esto nos da una ligera idea de dónde se encuentra.
- Redes GSM: Cualquier dispositivo que se conecte a una red telefónica. Es un método más preciso que mediante la dirección IP.
- GPS: Es el método más preciso, pudiendo concretar la posición del usuario con un margen de error de escasos metros.

Geolocalizaión.

Lo primero que tenemos que tener en cuenta es si el navegador soporta la Gelocalización.

```
if("geolocation" in navigator){
  console.log("Existe geolocation");
}else{
  console.log("No existe geolocation");
}
```

Geolocalización

Una vez comprobamos que realmente la API está activa, podemos posicionar el equipo mediante **getCurrentPosition()**

```
navigator.geolocation.getCurrentPosition(function(posicion){
    console.log("Posición Latitude->"+ posicion.coords.latitude+" Posición
},function(error){
   console.log("Error ->"+ error);
})
```

Geolocalización

Objeto de opciones:

- enableHighAccuracy: Utiliza una fuente de información exacta [bool]
- timeout: Tiempo máximo de espera para obtener las coordenadas en milisegundos. (Infinity por defecto)
- maximumAge: Indica la edad máxima de una geolocalización utilizable en milisegundos. Puede ser:
  - 0, lo que significa que no se pueden utilizar posiciones del caché.
  - Infinity, puede utilizar cualquier posición almacenada en caché.
  - Un número, indicando la antiguedad máxima de la localización en milisegundos.

Geolocalización

También podemos rastrear la posición actual mediante la función **watchPosition()** que recibe los mismo parámetros que **getCurrentPosition()**, el cual nos devuelve una ID del rastreador

```
var watchID = navigator.geolocation.watchPosition(function(position) {
  console.log("Funciones" + position.coords.latitude, position.coords.long
});
console.log("watchID " + watchID);
```

Geolocalización.

Para eliminar el rastreo debemo utilizar el metodo **clearWatch**

```
navigator.geolocation.clearWatch(watchID);
```

GEEKSHUBS Academy_

WEBWORKERS

WEBWORKERS

Los WebWorkers nos permiten ejecutar threads en el navegador. Al final nos permite crear un entorno en el que un bloque de código JavaScript puede ejecutase de forma paralela sin afectar al thread principal del navegador

WEBWORKERS

Los Webworker se ejecutan en un subproceso aislado, por ello es necesario que el código que se ejecuta se encuentre en un archivo aislado. Para crearlo instanciamos nuevo Worker con la dirección donde se encuentra el archivo .js

```
var worker = new Woker('./js/tarea.js');
```

WEBWORKERS

El **WebWorker** no se pondrá en funcionamiento hasta que se haya cargado el archivo js, y este se instancia en el navegador de forma asíncrona.

WEBWORKERS

Los webworkers utilizan una API de transferencias de mensajes para comunicarse entre los diferentes "threads" (hilos de procesamiento)

La comunicación entre un Worker y su página principal se realiza mediante un modelo de evento y el método postMessage

```
    //Archivo de Worker (tarea.js)
postMessage('es una prueba');
    //Arhivo Principal (main.js)

var worker = new Worker('./js/tarea.js');
worker.addEventListener('message', function (e){
  alert(e.data);
}, false);
```

WEBWORKERS

No sólo podemos enviar cadenas de texto, también podemos remitir json como objetos mediante las propiedades de **JSON**

```
    //Archivo Worker (tarea.js)
let message1 = JSON.stringify({
  name: {first: 'Ivan', second: 'Ruiz'},
});
postMessage(message1); // se envia como string

    //Archivo Principal. (main.js)
var worker = new Worker('./js/tarea.js');

worker.addEventListener('message', function (e){
  alert(e.data); // Se recibe como string
  console.log(JSON.parse(e.data)); // lo pasamos a objeto
}, false);
```

WEBWORKERS

Para finalizar el WebWorker podemos utilizar le método **terminate()**

```
Start Worker
Stop Worker
```

WEBWORKERS

Modificación de la aplicación **app.js**

```
var worker;
function startWorker() {
  worker = new Worker('./js/tarea.js');
  // worker.terminate() // demostracion terminate.
  worker.onmessage = function (e) {
    alert(e.data);
    console.log(JSON.parse(e.data));
  }
}

function stopWorker(){
  worker.terminate();
}
```

GEEKSHUBS Academy_

WEBWORKERS

Como detectar si el navegador esta habilitado para utilizar **WEBWORKERS**

```
var worker;
if(typeof(Worker) !== "undefined") {
  if(typeof (worker) === 'undefined'){
    worker = new Worker ('./js/tarea.js');
  }

  worker.onmessage = function (e) {
    alert(e.data);
  }
}else {
  alert('Error, no tienes soporte para ejecutar WebWorker');
};
```

WEBWORKERS

Que no podemos hacer con los WebWorkers acceder las objetos propios del navegador es decir:

- DOM
- Objeto Window
- Objeto Document
- Objeto Parent

WEBWORKERS

Además podemos gestionar Errores, mediante la captura del evento erro, que nos devuelve la siguiente información.

- filename -> El nombre de la secuencia que genera el error
- lineno -> El número de línea donde se produjo el error
- message -> Una descripción significativa del error

WEBWORKERS

GEEKSHUBS Academy_

Ejemplo de un worker con error

```
// en main.js
function onError(e){
document.getElementById('error').textContent= [
    'ERROR: Line ', e.lineno, ' in ',  e.filename, ': ', e.message].join('
}

function onMsg(e){
document.getElementById('result').textContent = e.data;
}
worker = new Worker('./js/tarea.js');
worker.onerror = function(err) { onError(err); }
worker.onmessage = function (data) { onMsg(data); }
worker.postMessage('');
// en tarea.js
self.onmessage = function (message) {
message1.message = message;
postMessage(message1;
}

var message1 = JSON.stringify({
name: {first: 'Ivan', second: 'Ruiz'}
});
```