CONFECCION Y PUBLICACIÓN DE PÁGINAS WEB. MODULO 2

MODULO 2:

MF0951_2: Integrar componentes software en páginas web

UF1305: Programación con lenguajes de guión en páginas web. (90 horas).

UF1306: Pruebas de funcionalidades y optimización de páginas web. (90 horas).

MF0951_2: Integrar componentes software en páginas web

UF1305: Programación con lenguajes de guión en páginas web. (90 horas).

FUNCIONES

- También conocidas como subprogramas
- Bloques de código que realizan una tarea específica y que a veces pueden **retornar valores**
- Usar funciones es la forma más optima de desarrollar aplicaciones, dividiendo el programa principal en pequeños subprogramas que se llama entre si hasta dar con la solucion final.
- Las funciones pueden reutilizarse e invocarse dede cualquier punto de nuestro código tantas veces como sea necesario
- Sintaxis

  function nombreFunction (parámetros){
  instrucciones a ejecutar}

FUNCIONES

- Function Palabra clave que identifica el comienzo o la declaración de una función
- Nombre función: Es el nombre que le daremos a la función (se aconseja el método CamelCase para construir este nombre.
- Parámetros Son nombres de variables separados por comas. Estas variables contienen valores de entrada que recibe la función y que son imprescindibles para ejecutar las instrucciones que tiene definida la función.

  En caso de que nuestra función no necesite parámetros de entrada, la declaración sería con un paréntesis vacío.

FUNCIONES

- Recogemos información para la función a través del parámetro
- Utilizamos las funciones para mostrar un mensaje en una página, mover un objeto a través de la pantalla o validar los campos de un cuestionario.
- Otras veces querremos recuperar algo de una función. Una función que calcula el precio total de los productos de una lista de la compra no es muy útil a menos que la función permita conocer el total final.
  En el caso del prompt hemos visto como podemos almacenar el valor en una variable y hacer algo con el.
- En el caso de querer recuperar un valor de una función propia, se usa **return** seguido del valor que se quiere recuperar
- Ejemplo RETURN en CARPETA COMENTADOS

FUNCIONES

- INVOCAR UNA FUNCION

  Una función por si sola no hace nada, no tiene efecto ninguno, es necesario que se le llama o invoque desde algún punto de nuestro código para que ejecute las acciones que tiene definidas

  Cómo se invoca una función?

  nombreFunción(valores)

- Ejemplo

```
<script type="text/JS>
function mostrarMensaje (){
alert("prueba básica de funciones en JavaScript")
</script>
```

Se declara una función sin parámetros de entrada y cuyo código únicamente consiste en mostrar un mensaje por pantalla, ya fuera de la función se hace la llamada a la misma

FUNCIONES

- INVOCAR UNA FUNCIÓN (otra forma):
  Tras la modificación del código se ha incluido como parámetro de entrada en la función la variable mensaje, el valor de esta variable será el texto que se mostrará en pantalla.

```
<script type="text/javascript> function mostrarMensaje (texto){
alert(texto);}

mostrarMensaje("este es el mensaje pasado por parámetro"

                    //llamamos a la funcion declarada anteriormente
```

- Con este ejemplo podemos entender el funcionamiento básico de las funciones, pero es un claro ejemplo de lo que nunca deberíamos hacer: crear una función para una tarea existiendo ya otra función que hace exactamente lo mismo (hemos creado una función que muestre un mensaje en pantalla cuando ya conocemos la existencia de alert (función predefinida en js que tiene como finalidad mostrar mensaje al usuario)

FUNCIONES

- RETORNAR VALORES:
  Esta es una de las utilidades más importantes de las funciones en JavaScript
  Hasta ahora lo que hemos viso es como las funciones ejecutaban acciones, pero no retornaban ningún valor.
  La forma que tiene JS de devolver valores en una función es mediante la palabra clave **return** seguida de un valor o variable.
  ¿cómo recogemos el valor de salida de una función? Igual que se invoca a una función, pero asignándole el valor a una variable (ver ejemplo página siguiente)
- Importante:
  - Dentro de una función únicamente se puede retornar un valor (no es que solamente pueda existir un return por cada función. Se pueden añadir tantas instrucciones return como sean necesarios, pero teniendo en cuenta que el flujo del código solo nos llevará a definir una. Y posteriormente hacer el return de esa variable
  - y que el uso de return implica de manera automática el fin de la función
  -

FUNCIONES

- RETORNAR VALORES. EJEMPLO1(en la carpeta "comentados". 15. comentado_ejemplo_return1)

Definimos la función **calcular,** que tiene como parámetros de entrada los números que vamos a sumar.
Dentro de la función se realiza la suma y se almacena en la variable, **resultado**
con **return** devolvemos el valor de la suma que está almacenado en la variable, **resultado.**
Ya fuera de la función, declaramos una variable suma y la inicializamos con el valor devuelto por la función calcular. Es en este momento también cuando pasamos a la función los valores que debe sumar
Alert: muestra el mensaje con el resultado de la suma. Concatenando cadenas de texto con la variable suma

- RETORNAR VALORES. EJEMPLO2(en la carpeta "comentados". 16 comentado_ejemplo_return 2)**Enunciado**: Implementar una función en JS que dados dos números cualesquiera nos retorne el mayor de los dos, si los números son iguales devolverá cero.

-La función 'mayorMenor' va calculando el mayor de los dos números que son pasados por parámetro y almacenado el resultado en la variable 'mayor' Al final de la función se hace el retorno de la variable mayor que almacena el resultado de las comparaciones.
ya fuera de la función se utiliza la variable numeroMayor para guardar el resultado de la función.

Con un if se analiza el valor de la variable numeroMayor y se muestra por pantalla el resultado final.

FUNCIONES

- RETORNAR VALORES. OTRO EJEMPLO: Carpeta Comentados: 17 comentados Segundo ejemplo return. Un return para cada caso
  La diferencia de este código respecto al código anterior radica en la forma en la que se gestionan los diferentes casos dentro de la función, es decir , ya no existe la variable mayor donde se almacena el resultado, en su lugar se lanza directamente un return con la solución de cada caso

Las funciones en JS van definidas y agrupadas en ficheros con extensión js que posteriormente son enlazadas en los documentos HTML.
Es recomendable agrupar en diferentes ficheros las funciones que comparten funcionalidades similares; de este modo, podríamos tener por ejemplo un fichero denominado validacion.js con las funciones destinadas a validar formularios

FUNCIONES PREDEFINIDAS

- EVA1(CODIGO)
- Escape(cadena)
- Unescape (cadena)
- encodeURI(cadena_url)
- decodeURL(cadena_url)
- encodeURLComponent(cadena_url)
- decodeURLComponent(cadena_url)
- parseInt (cadena_numero,base)
- parseFloat (cadena_numero,base)
- isNaN(expresion)

matriz

- Las variables también pueden almacenar varios valores a la vez en una estructura llamada **matriz..**
- Las matrices son variables multidimensionales que se crean utilizando una sencilla notación de corchetes separando cada valor con comas.
- Los valores se identifican más tarde utilizando un indice a partir de 0.
- Tanto las matrices como las variables pueden tomar cualquier tipo de valor: numeros, textos , valores booleanos o incluso declararar el valor como null o undefined.
- Ambos representan ausencia de valor, pero tambien se devuelve undefined cuando la propiedad de un objeto o un elemento de matriz no ha sido definida

matriz

- También puede realizar operaciones sobre los valores de una matriz y almacenar el resultado, como lo hicimos antes con las variables simples.(ejercicio comentado 6)
- En este ejercicio se muestra cómo modificar el valor de una matriz y concatenar texto.
- Al final de la última secuencia de comandos , el método alert() muestra ambos valore de la matriz separados por un espacio. Los valore y el espacio están unidos por el operador +. Este operador puedes ser utilizado no solo para realizar operaciones sino también para concatenar valores y cadenas con fines de estilo (los números se convierten en cadenas de forma automática

matriz

- Podemos extraer o agregar valores mediante los siquientes métodos:
  - **Push (valor)**: método que agrega un nuevo valor al final de la matriz. El atributo valor indica el valor que se añade
  - **Shift()**: Método que elimina y devuelve el primer valor de la matriz
  - **Pop():** Método que elimina y devuelve el último valor de la matriz

SINTAXIS JS

- **Ejercicio carpeta ejercicios**.(El llamado "operadores" del libro de ejercicios). Muestra un ejemplo que realiza distintas operaciones sobre dos variables
  la instrucción document.write será analizada posteriormente, por ahora es suficiente con saber que escribirá el contenido de cada variable en nuestra página HTML.

Modifica el ejemplo anterior, probando diferentes operadores aritméticos y comprobando en el navegador el resultado obtenido

- ESTRUCTURAS DE CONTROL
- Las estructuras para resolver cualquier algoritmo, que vimos en el capítulo anterior, tienen su representación en el lenguaje de script
  - **Sentencia condicional:** Si las condición es verdadera se ejecuta la instrucción o instrucciones del primer bloque, si la condición es es falsa, se ejecutan las condiciones del segundo bloque
    if (condicion)
    bloque de instrucciones 1
    else
    bloque de instrucciones 2

- FUNCIONES
- OBJETOS
- EVENTOS

SINTAXIS JS

Ejercicio carpeta ejercicios : CONDICIONAL .Este ejemplo, al igual que los siguientes, puede ser incluido en el body de cualquier documento HTML, además de este existen otros dos formatos de estructura condicional que veremos más adelante

- **Sentencia repetitiva.** Hay muchos formatos..los dos fundamentales son:
  - El primero que se basa en la repetición del bloque de instrucciones partiendo de un valor inicial para una variable , que se irá modificando en cada iteración y se repetirá mientras se cumpla la condición especificada
    for (inicio variable; condición; modificación de variable) {
    Bloque de instrucciones}

Ejercicio carpeta ejercicios SENTENCIA REPETIITVA1

Aquí la variable num" se inicializa con el valor 1, en cada iteración de bucle se aumenta de 1 en 1, de forma que va tomando los valores, 1, 2, 3….La condición que debe cumplir para que continue en el bucle es que sea menor o igual que 10, por lo que después de 10 veces, la variable valdría 11, no cumplirá la condición por tanto, saldría del bucle

***Modificar el ejemplo, probando diferentes valores para el valor incial de la variable num, el valor de la condición y el incremento, por ejemplo: que comience en 100, que la condición sea <120 y el incremento num+-3

- El segundo es cuando el formato de estructura repetitiva no va ligado al valor de una variable, simplemente se ejecuta el bucle mientras se cumpla la condición descrita

Ejercicio carpeta ejercicios SENTENCIA REPETITIVA2
se solicita que introduzcamos un numero mediante la instrucción parseInt(prompt("…") y mientras ese numero sea distinto de 0 lo irá acumulando en otra variable y solicitará que introduzcamos un nuevo numero. Finalmente nos muestra la suma final alcanzada

SINTAXIs JS

- FUNCIONES
  imprescindibles para que nuestros programas sean modulares, entendibles y reutilizables.
  Una función es un bloque de instrucciones identificado por un nombre y que puede ser invocado de forma directa o asociada a un evento.
  Por ejemplo, la suma del ejemplo anterior, podría ser estructurado en una función, de forma que podría ser invocado en más de una ocasión, por ejemplo asociado a que el usuario pulsase un botón. Verlo en el **Ejercicio carpeta ejercicios** FUNCIONES

  Proceso: al ejecutarse el script, se encuentra con la función sumar() el navegador no ejecuta ese código, lo ejecutará cuando se invoque desde otra parte del código. Esa invocación o llamada a la función se hace escribiendo simplemente el nombre de la función con los paréntesis, en nuestro caso asociada a un evento.
  La función parseInt()toma un texto (el que introduce el usuario mediante otra función alert) y lo convierte en un numero entero con el que poder realizar operaciones aritméticas.

SINTAXIS JS

•OBJETOS

al ser JS un lenguaje orientado a objetos podremos crear nuestras propias clases de objetos aunque la funcionalidad más útil del lenguaje en este aspecto viene dada por los objetos que tiene creados por defecto.

Sobre estos objetos ya creados podremos

- acceder a los atributos de los objetos, tanto para obtener sus valores como para modificarlos y
- también invocar a sus métodos para realizar tareas imprescindibles en nuestros scripts

Objetos más utilizados:

- Document: atributos y métodos referentes a la página web que contiene nuestro script
- Window: atributos y métodos referentes a nuestra ventana de navegador
- Math: mantiene funcionalidad matemática
- Date: atributos y métodos referentes
- Array: utilidades para almacenar datos de matrices
- String: las cadenas de caracteres son objetos de esta clase. Nos ofrece multitud de métodos para su tratamiento

Ejercicio carpeta ejercicios OBJETOS

Al pulsar el botón se cambian dos propiedades de la página web, mediante el objeto document

La clase Date ya la hemos empleado en ejemplos anteriores

la instanciamos con la instrucción new. Se crea un objeto con la fecha y hora actuales. Una vez creado podemos llamar a los métodos que nos devuelven el año, el mes el dia de la semana…

•

•EVENTOS

SINTAXIS JS

•EVENTOS

La interactividad se logra mediante los eventos que se pueden añadir a cualquier elemento de la página

Hay muchos, pero ahora nos centraremos en las diferentes categorías de eventos disponibles:

- Eventos de Ratón: onclick, onmauseover…
- Eventos de teclado: Onkeypress Al pulsar una tecla
- Eventos de formularios: Onsubmit. Al enviar el formulario al servidor, onblur: al perder el foco el elemento que tiene asociado el evento
- Otros tipos de eventos: por ejemplo: onabort, cuando el usuario interrumpe la carga de la imagen

Los eventos se añaden a los elementos HTML como cualquier otro atributo, dentro de la etiqueta de apertura. El valor asignado a este atributo será la tarea a realizar por JS, generalmente representada por el nomber de una función que es invocada:

<etiqueta HTML evento=“funcion=

A un mismo elemento podemos asignarle distintos eventos, llamando a distintas funciones en cada uno de ellos

UBICACIÓN DEL SCRIPT EN HTML

Podemos situar el código JS tanto en HEAD como en BODY. El ubicarlo en uno u otro lugar no afecta al rendimiento del programa, ni a los tiempos de carga de la página; Es simplemente una cuestión de organización y de claridad de cara a la codificación, prueba y mantenimiento

•SCRIPT EN EL CUERPO DE HTML

En este caso hay que diferenciar entre un script no asociado a ningún evento y el asociado a un evento

- SCRIPT NO ASOCIADO A NINGÚN EVENTO
  El código no asociado a ningún evento se va resolviendo a medida que se va cargando la página y podrá añadir contenido a la misma como el propio HTML , pero de forma dinámica
  Ej de este caso: cálculo de fecha y hora actual
  El script realizará una serie de cálculos y finalmente escribirá en el código HTML,
  texto y etiquetas en función de dichos cálculos
  Para realizar esta acción de generar código HTML empleamos el método document.write al que le podemos pasar como parámetros cadena de caracteres entre comillas o bien valores de variables, todas ellas unidas con el simbolo +

  Ejercicio carpeta ejercicios de SCRIPT NO ASOCIADO A NINGUN EVENTO. Muestra una página que incluye mezclado con HTML, código generado dinámicamente y no asociado a ningún evento: con un encabezado <h1> escribirá "hoy es dia D del M de A" siendo D el día actual, M el mes actual y A el año actual.

  El único requisito que debe cumplir el código es que comience con <script languaje "javascript" y finalice con </script> y podremos situarlo en cualquier punto del body y entre cualquier etiqueta del lenguaje de marcas

  SCRIPT ASOCIADO A UN EVENTO

- En este caso depende de que el usuario realice una determinada operación.
  Ejercicio carpeta ejercicios SCRIPT ASOCIADO A UN EVENTO
  en este ejercicio lo que realmente nos interesa es que el texto llamado *nombre* tiene asociados dos eventos: onmauseover y onmausout e invocar a sendas instrucciones en JS:una cuando el ratón pasa por encima de dicho campo y otra cuando el ratón sale de él

UBICACIÓN DEL SCRIPT EN HTML

- SCRIPT EN EL ENCABEZADO DEL HTML

  la forma más clara de construir aplicativos con javascript se basa en llamadas a funciones desde la seccion body (tanto asociadas a un evento como directamente sin evento asociado) y en la sección <head> la definición de esas funciones

  Ejercicio carpeta ejercicios SCRIPT EN EL ENCABEZADO DEL HTML

  En este ejemplo nos encontramos una llamada no asociada a ningún evento a la función ecribir_fecha ()y una llamada a la función cambiar_moneda() que se ejecuta cuando se pulsa el hipervínculo. La primera función escribe el dia y año actuales y la segunda calcula para los euros introducidos en la primera caja de texto, el importe equivalente en dólares, suponiendo un cambio fijo de 1,35 dolores por euro.

En el ejemplo, el cuerpo del documento queda limpio con el propio lenguaje de marcas y simples llamadas a funciones que nos ocultan las complejidad de las operaciones realizadas

Además, al centrarnos en el lenguaje de script no tenemos etiquetas HTML mezcladas que dificulten su análisis.

En este ejercicio introducimos una nueva forma de llamar a funciones mediante eventos en este caso el evento que llama a la funcion es el clic en una etiqueta de hipervínculo:

texto del enlace

Obviamente no es un hipervínculo como tal , ya que no viaja a otra página, simplemente ejecuta la función javascrippt como si de un botón se tratase

UBICACIÓN DEL SCRIPT EN HTML

- SCRIPT EN UN ARCHIVO EXTERNO

  En el caso de que el código sea muy amplio se crea un archivo externo y por otra parte en el documento HTML incluir una referencia en la sección head para que ese archivo js sea cargado como si fuese parte del propio documento HTML

  - <script type="text/javascript" src"funciones.js"></script>

Ejercicio carpeta ejercicios SCRIPT EN UN ARCHIVO EXTERNO

EJECUCIÓN DE UN SCRIPT

Para ejecutar cualquier código JS en nuestro ordenador no es necesario ningún software especial, puesto que se sitúa **enbebido** en diferentes secciones de un documento HTML, o bien en un archivo aparte, pero que es referenciado desde un documento HTML.

•Los documentos web tienen que estar alojados en un servidor web y cuando el usuario solicite dicho documento escribiendo su URL en el navegador, el servidor le servirá la página con el JS asociado
ojo!! Podríamos ejecutar un documento con JS que tengamos almacenado en nuestro ordenador sin que este sea un servidor web; al hacer doble clic sobre el documento, será abierto en nuestro navegador web predeterminado. Será el navegador el que ejecute línea a línea todas las instrucciones del script.

•El código es interpretado por lo que se irá traduciendo a código binario a medida que se ya ejecutando…esto puede hacer que haya líneas que no se ejecuten(ej las de una rama de una estructura condicional)

•Esto dificulta detectar errores, porque las instrucciones que no se ejecuten no serán traducidas y por lo tanto no se mostrará ningun sintoma si hay algún error en ellas.
Por este motivo es importante hacer unas pruebas muy estrictas garantizando que toda la líneas de código se han ejecutado para evitar que produzcan errores cuando el programa ye está en funcionamiento

EJECUCIÓN DE JS EN LOS NAVEGADORES
Algunos navegadores protegen la ejecución de JS asumiendo que puede contener código dañino para nuestro equipo.
Cómo configurar el navegador para que acepte los Scripts:

MOZILLA: Herramientas>opciones>pestaña contenido: marcar : Activar JS
Dependiendo de la versión puede no aparecer la barra de menús **HERRAMIENTAS> Opciones de Internet>Seguridad**> Con el icono **INTERNET** seleccionado pulsamos el botón **Nivel personalizado>automatización** y en **Active Scripting** marcamos **Habilitar.** A continuación también marcamos en la pestaña **Avanzado Permitir Contenido Activo en mi ordenador**

CHROME: Icono de Personalización y Configuración (llave inglesa en la esquina superior derecha)>**OPCIONES**> **AVANZADA**> **PRIVACIDAD** y pulsamos el botón **CONFIGUACION DEL CONTENIDO>PERMITIR QUETODOS LOS SITIOS EJECUTEN JS**

EJECUCIÓN DE UN SCRIPT

TIEMPOS DE EJECUCIÓN

Según lo visto hasta ahora, la ejecución del código de Script puede estar asociada a un evento o no, independientemente de su ubicación en la cabecera o en el cuerpo del documento. Así pues tendríamos ya los 3 casos ya comentados, que son los más típicos:

- **Durante la carga del documento:** En este caso el código no aparece asociado a ningún evento, se introduce el código JS en el body del lenguaje de marcas dentro de el elemento <SCRIPT></SCRIPT>. EN ESTE CASO LA EJECUCIÓN SE REALIZA SIEMPRE INCONDICIONALMENTE
- **Asociada a un evento** el código se ejecuta al producirse un evento en un elemento HTML: Es decir cabe la posibilidad de que no pueda llegar a ejecutarse (si no se produce el evento) y también que se ejecute varias veces en el caso de el evento se produzca varias veces.
- **Al cargar el documento.** En realidad es una ocurrencia concreta del caso anterior. Al comenzar la carga del documento se activa el evento onload (asociado a la etiqueta BODY.
A diferencia del caso anterior es que este evento es lo primero que se hace antes de la carga de la página, mientras que el primero puede ser en cualquier momento de la carga,

Pero además de estas situaciones típicas, JS nos ofrece otras funciones para modificar los tiempos de ejecución, en las que profundizaremos más adelante:

setTimeout: ejecuta el código tras un determinado periodo de tiempo. Existe una función complementaria: ClearTimeout que cancela la anterior
SetInterval:ejecuta el código cada cierto intervalo de tiempo, de forma repetida.

EJECUCIÓN DE UN SCRIPT

ERRORES DE EJECUCIÓN

JS, a diferencia de otros lenguajes, no posee un entorno de desarrollo o prueba potente que nos permita detectar errores de programación.

Además al ser un lenguaje interpretado, los errores pueden no detectarse en las primeras ejecuciones, ya que puede que haya código que no se ejetute (por ejemplo en sentencias condicionales) y por lo tanto no se procese por el Navegador, con lo que no sabremos si hay algún error o no.

En la configuración por defecto de los navegadores, un error de programación en la mayoría de los casos puede hacer simplemente que no se ejecute el código y se muestre el HTML sin más, sin mostrar ningún síntoma adicional

Podemos distinguir 2 tipos diferentes de errores:

En tiempo de carga: Son errores que e detectan en el momento de la carga sin que estén relacionados con la ejecución del proceso, generalmente son errores sintácticos (se estudiaran con profundidad más adelante)

En tiempo de ejecución. Ej. Si se trata de suma de números y el usuario introduce letras (estos errores no se corrigen, sino que se previenen con un buen estilo de programación, modificando el código para evitar que el usuario introduzca valores que produzcan errores..También podemos incluir instrucciones de tipo try…catch que se adelantan al problema: Si la operación a realizar va a producir un error, lo capturan y realizan una acción

Ejercicios repaso

- **Ejercicio carpeta ejercicios** repaso 1. Sustituye el evento onclicK por el evento ondblclick y también por el evento onmouseover para ver las diferencias de comportamientos.
- Luego introduce algún error a propósito y compruebe la respuesta del navegador
- **Ejercicio carpeta ejercicios** repaso 2. Separe la función a un archivo aparte. En el documento HTML deberá introducir una referencia a este nuevo archivo para poder utilizar la función. Compruebe que sigue funcionando como el programa original

Herramientas de desarrollo

Qué herramientas necesitamos para desarrollar este lenguaje JS:

Lo más básico sería un bloc de notas y un navegador web

1. Un editor de texto para codificar nuestros programas
2. Un sistema de dección /corrección de errores
3. Un entorno en el que probar y ejecutar el código desarrollado (un navegador web)

Herramientas de desarrollo

A CONTINUACIÓN VAMOS A VER LOS TIPOS DE EDITORES

•EDITORES DE TEXTO PLANO (ej notepad++, 1st JS editor….)
Herramientas que nos permiten escribir texto. Se diferencian de los procesadores de texto porque estos editores de texto de programación no tienen formato de texto interno. Es decir no tiene sentido hablar de negrita, subrayado, alineación justificada….ya que el resultado final deben ser caracteres simples

ventajas de un editor JS:

1. Resaltar con colores los diferentes elementos de lenguaje
2. Marcar los comienzos y finales de cada bloque para encontrar pincipios o finales de bloque huerfanos
3. Detección de errores sintácticos
4. Gestion de las tabulaciones de código para una mayor claridad a la hora de depurarlo
5. Lanzar el código en un navegador web para probarlo
6. Funcion de autocompletado que nos ayude a identificar las palabras reservadas , eventos, funciones….y los complete sin errores de sintaxis

•EDITORES DE APLICACIÓN WEB (ej Dreamweaver qu ecada vez incluye más comportamientos y efectos visuales generados con JS y AJAX)
estos además de las carácterísticas de editores planos, añaden la propiedad WYSIWYG (lo que se ve es lo que se obtiene). Se crea la web sin tener conocimientos de HTML CSS o JS (con comportamientos)

•ENTORNOS INTEGRADOS DE DESARROLLO o IDE (Netbeans, Visual Studio, Eclipse, WebStorm….)
Los editores vistos hasta ahora son útiles para desarrollar pequeños scrpts, sobretodo cuando estamos centrados en el desarrollo del lado del cliente, pero cuando nuestra misión es realizar aplicativos web más complejos, y en los que utilizaremos tecnologías de cliente y de servidor, necesitamos herramientas más potentes que nos proporcionen:

- Las funciones de edición
- Un entorno que de soporte a todas estas tecnologías trabajando conjuntamente. Herramientas que nos proporcionen en un solo entorno las funcionalidades para el desarrollo web sin necesidade de cambiar de herramienta:
  - SERVIDOR WEB CAPAZ DE PROCESAR LENGUAJE DE SERVIDOR (JAVA, PHP, ASP, NET)
  - utilidades para la construcción de HTML y CSS
  - y programacion del lado del cliente con JS

Herramientas de desarrollo

- RECURSOS WEB PARA LA CREACION DE SCRIPTS
  - A veces los módulos de código con funcionalidad similar o complementaria se agrupan en un único archivo al que denominamos librería
    - Las librerías deben informarnos de las funciones disponibles, su objetivo o funcionalidad, y forma de utilizarlas; es decir su sintaxis, qué parámetros requiere….
    - La forma de utilizar una librería es incorporarla a nuestro script con si se tratase de cualquier archivo de JS externo. <script languaje="javascirpt" src="libreria,.js"></script>
    - Podemos destacar las siguientes:
      - Jquery. Una de las librerías más potentes. Nos permite realizar multitud de tareas de forma simple, desde efectos gráficos a comunicación AJAX con el servidor, pasando por validación de campos de formularios
      - Mootools: Conjunto de librarías con una jerarquía de objetos que abarca multitud de áreas de programación similar a la anterior. Se suele utilizar en webs con alto rendimiento gráfico, como galerías de imágenes, gráficos en 3D….
      - Prototype: igual que las anteriores abarca diversos ámbitos del lenguaje con una orientación especial hacia AJAX
      - JSValidate: incluye funciones de validación de campos de formularios, permitiendo distintos tipos de validación (campo alfanumérico, una dirección de correo electrónico….)

Ejercicio carpeta ejercicios 1 Jquery. Para programar una imagen que se desvanezca al pulsar un botón y vuelva a reaparecer suavemente al pulsar otro

Ejercicio carpeta ejercicios 2 JSValidate y PROTOTYPE
basada en la primera y utiliza recursos de la segunda

DEPURACION DE ERRORES

La validación y prueba de aplicaciones, para detectar y corregir errores es la fase que más tiempo y recursos consume.

- TIPOS DE ERRORES. Básicamente existen 2:
  - **De edición**: los que se producen al escribir el código (errores de sintaxis, ej una llave en una posición errónea puede hacer que nuestra aplicación funcione de una forma totalmente diferente)
  - **De ejecución** Debidos a una situación que se produce durante la ejecución del script ej valores erroneos introducidos por el usuario.

DEPURACION DE ERRORES

- DETECCION DE ERRORES EN LA EDICIÓN
  Sabemos que , en el caso de JS, es el navegador el que traduce cada instrucción del código fuente a código binario en el momento en que la va ejecutar.
  A veces una instrucción situada en una sentencia secuancial puede no ejecutarse frecuentemente si el programa circula con la otra rama de la sentencia y los posibles errores de esa primera instrucción podrían pasar inadvertido.
- Algunos editores de código JS incluyen una utilidad para revisar este tipo de errores sin necesidad de ejeutar una a una las instrucciones

DEPURACION DE ERRORES

- DETECCION DE ERRORES EN LA EJECUCIÓN

Ejercicio carpeta ejercicios 3 1 DETECCION DE ERROR

Abrimos este ejemplo y sustituimos la sentencia de la rama else por la siguiente dddocumento.write (no estamos en el siglo XXI")

Abrirlo con el navegador y explica qué ha ocurrido (a diferencia de otros lenguaje de programación, cuando ejecutamos un script con errores sobre un navegador web en su configuración por defecto, no nos proporciona ningún mensaje o pista para detectarlo)

- Para solucionar este problema algunos navegadores web nos ofrecen las posibilidad de detectar este tipo de errores. Vamos a ver el caso se MOZILLA Y CHROME
  MOZILLA con la extensión Firebug. Una vez instalada nos permite detectar cualquier error
  1. Para instalarla en mozilla: menú complementos> Obtener complementos>en la casilla de búsqueda tecleamos Firebug (icono cucaracha)>Instalar> reiniciar navegador
  2. Activamos la extensión clicando sobre su icono. Se nos abrirá en la parte de abajo una ventana con el código HTML de la página que estemos visitando en ese momento.
  3. Abrimos un scriipt cualquiera de los ejemplos anteriores pero modificado para que tenga un error de sintaxis en una línea de las que se vaya a ejecutar (no vale el error del ej 31 porque esta en una rama que no se va a ejecutar

En el ejercicio que hicisteis ayer MODIFICAMOS LA FUNCION ALERT CON UNA ERRATA EJ AAAALERT.

Al ejecutarlo en firefox con la extensión activada, si clicamos en la pestaña consola veremos un mensaje descriptivo del error

DEPURACION DE ERRORES

- **EN CHROME**
  de forma predeterminada y si instalar ningún complemento, G C permite mostrar los errores de forma parecida al firebug
- Pasos: Con el documento abierto selecciona el icono (llave inglesa) (Herramientas > herramientas desarrolladores/o cntrl+may+I)
- Además de la detección de errores, este navegador dispone de una potente herramienta de traza de programas, (debugger) Consiste en la ejecución de un programa paso a paso de forma que el programador controla la evolución del script
- Funciones principales de un debugger:
  - Ejecutar instrucción a instrucción, con opción de pausar el script y volverlo a reiniciar
  - Ejecutar directamente hasta un punto determinado del script, al que se denomina breakpoint (saltar a)
  - Ejecutar sin entrar en el detalle de las funciones, de forma que vemos la ejecución del cuerpo principal del script sin los detalles
  - Mostrar el valor que van tomando las variables a medida que van avanzando el programa

Mediante estos botones, se controlan la ejecución paso a paso, la pausa, el salto hasta el breakpoint….
En el apartado Watch Wxpression s indicamos las variables a las que queremos seguir su valor y finalmente, en los número de linea de script podemos clicar para fijar los breakpointes que queramos…Volviendo a clicar sobre ellos los haremos desaparecer

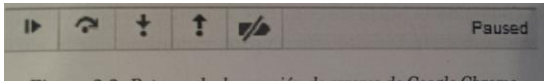

Figura 3.9. Botones de depuración de errores de Google Chrome

CONTROL DE ERRORES EN EL SCRIPT

Las pruebas anteriores son interesantes en tiempo de codificación y prueba, pero una vez que el script está funcionando en el mundo real a disposición de los usuarios, lo que se conoce como "entorno de producción" estos sistema ya no son válidos pues no son interpretables por el usuario.

Para los errores que no hayamos descubierto y también para aquellos no controlables porque dependan de la interacción del usuario con el programa, JS nos ofrece maneras de asegurarnos que la ejecución de la aplicación sea correcta:

•Mediante el bloque de instrucciones try(código conflictivo…catch
se trata de una estructura capaz de capturar un error y ejecutar el código que deseemos en vez de producirse el comportamiento por defecto del navegador en caso de errores.
Sintaxis

```
try{ (código conflictivo)
}…
catch (err) { (codigo conflictivo si se produce un error
}
```

En el caso de que tengamos una serie de instrucciones que puedan producir errores de ejecución, las situamos en la parte try del bloque. Así en el caso de que el error se produzca no se para la ejecución del script, ni tampoco se mostraran mensajes de error del sistema, sino que se ejecutarán las instrucciones situadas en la parte **cath.**

Además la variable **err** incluida en la instrucción ***cath*** es un objeto que captura las características de error

Ejercicio carpeta ejercicios 3.2

En este ejemplo hemos añadido voluntariamente un error sintáctico al código incluido en la parte try.com, con lo que obligamos la ejecución del código de la parte catch.

Al ejecutarse esta última parte se mostrará una ventana de diálogo con la palabra error y una descripción del mismo proporcionada por la variable err

•Mediante el evento ONERROR

Este evento captura cualquier error que se produzca en el script y permite tomar una acción común sea cual sea el tipo de error.

Se suele incluir entre unas etiquetas <script></script> solo, y al principio del <head> para que su ámbito de acción se lo mayor posible.

Como la operación a realizar será común para cualquier tipo de error, es típico que la acción a tomar sea una redirección hacia una página de error genérica que simplemente informa de que se ha producido un error. Para realizar esta operación empleamos el objeto document, con su propiedad location.href

Ejercicio carpeta ejercicios 5 se incluye un primer script simplemente con el evento **onerror** y la redirección a la página de error, y un segundo script ya con el programa que deseamos que tenga el control de errores

test

1 Indique qué afirmación es falsa:

a) El código JavaScript puede ser editado en cualquier editor de texto plano.

b) Para codificar en JavaScript necesitamos un editor de texto orientado a este lenguaje.

c) Hay editores de texto que facilitan la codificación resaltando con colores ciertas palabras reservadas, variables, etc.

d) JavaScript no necesita compilador ya que es un lenguaje interpretado.

2 Indique qué afirmación es falsa:

a) El editor gratuito Notepad++ ofrece autocompletado de términos para varios lenguajes.

b) El editor 1st JavaScript Editor ofrece autocompletado de términos para JavaScript.

c) Los editores WYSIWYG permiten crear aplicaciones web cliente sin codificar.

d) Kompozer es un editor WYSIWYG gratuito.

3 Indique qué afirmación es falsa:

a) Las librerías son un conjunto de funciones que podemos utilizar en los scripts.

b) Las librerías deben incorporarse al *script* para utilizar sus funciones.

c) Librerías como JQuery nos facilitan incorporar efectos visuales a los sitios web.

d) El código de la librería se ejecuta remotamente, en la web donde la descargamos.

4 Indique qué afirmación es falsa:

a) Firebug es una extensión de Mozilla Firefox que detecta errores en el código JavaScript.

b) Google Chrome permite ejecutar sentencia por sentencia el código JavaScript.

c) El editor 1st JavaScript Editor permite validar sintácticamente el código sin ejecutarlo.

d) Netbeans es un IDE gratuito para codificar programas en Java, no en JavaScript.

1. Metodología de la programación
– Lógica de programación.
– Descripción y utilización de operaciones lógicas.
– Secuencias y partes de un programa.
– Ordinogramas.
– Descripción de un ordinograma.
– Elementos de un ordinograma.
– Operaciones en un programa.
– Implementación de elementos y operaciones en un ordinograma.
– Pseudocódigos.
– Descripción de pseudocódigo.
– Creación del pseudocódigo.
– Objetos.
– Descripción de objetos.
– Funciones de los objetos.
– Comportamientos de los objetos.
– Atributos de los objetos.
– Creación de objetos.
– Ejemplos de códigos en diferentes lenguajes.
– Códigos en lenguajes estructurales.
– Códigos en lenguajes scripts.
– Códigos en lenguajes orientados a objetos.

¿Qué necesito para crear un sitio web dinamico?

- Son lenguajes para ser interpretados, no para ser compilados, por lo que no generan ningún archivo ejecutable.
- Por tanto, es necesario contar con un intérprete para ejecutar código Javascript, y el intérprete que se utiliza una frecuencia: se incluye en tu navegador de internet.
- Cada navegador tiene un intérprete Javascript, que varía en función del navegador.
  - Si está utilizando Internet Explorer, el intérprete es llamado JScript (versión 9 intérprete llamado Chakra),
  - en Mozilla Firefox se llama SpiderMonkey y el motor V8 es el de Google Chrome.

PASOS

- Los lenguajes de script, al igual que el resto de lenguajes de programación, disponen de variables, métodos y objetos predefinidos, pero un menor control sobre los tipos de variables (enteros, cadenas de caracteres, etc.), por lo que es posible asignar valores de distintos tipos a una misma variable, lo que puede producir errores y dificultar la depuración de los programas.

- Si utilizáramos un editor visual, como Dreamweaver, podríamos insertar algunos comportamientos de este tipo sin la necesidad de escribir ni una sola línea de código JavaScript. Esta aplicación permite insertar comportamientos a través de menús y paneles, generando automáticamente el código JavaScript necesario.
- Para insertar funciones JavaScript en un documento, es necesario insertar las etiquetas **<script>** y **</script>** dentro de la cabecera