FUNCIONES PERSONALIZADAS

Introducción

En términos generales, una función es un "subprograma" de alta reusabilidad que puede ser llamado mediante código externo de la misma (o interno, en caso de recursión). Al igual que el programa en sí mismo, una función se compone de una secuencia de instrucciones que conforman el llamado cuerpo de la función.

Etapas

Una función puede tener diferentes etapas dependiendo de su tipología:

- **Funciones nominales:** en la fase de declaración se la dota de un nombre específico, y en la de ejecución se invoca la función a través de ese mismo nombre seguido de un paréntesis, provocando la ejecución las instrucciones encapsuladas en la misma. Estas funciones pueden ser ejecutadas un número indeterminado de veces desde cualquier punto del script (ver función_nominal.html) Existen varias formas sintácticas para declarar una función nominal (ver declaraciones.html)

- **Funciones anónimas:** carecen de nombre y son declaradas en su propia ejecución: se invocan únicamente en el momento de ejecución del script, o al desencadenarse el evento al que estén asociadas (ver funciones_anonimas.html)

Características y funcionalidades

Las características base de las funciones son las siguientes:

- Se pueden pasar valores a una función como **argumentos** (ver argumentos.html)
- Cuando sea necesario podemos asignar valores por defecto a cualquier argumento, que será utilizado cuando no enviemos el valor de este en la invocación (ver argumentos2.html)
- Los denominados **parámetros REST** permiten enviar a la función una cantidad indeterminada de argumentos que son recibidos como un Array, anteponiendo tres puntos (...) al último argumento en su declaración (ver parámetros_REST.html)
- La función puede devolver un valor mediante una única sentencia **return** que especifique el valor a devolver, si no existe **return** retorna undefined (ver return.html)

Alcance de las variables

Cuando una variable es declarada fuera de una función, se denomina pública o global, estando disponible para la totalidad del script con un consumo mayor de memoria. Al ser declarada en su interior, se denomina privada, limitando su alcance al interior de la función en la que fue declarada (ver alcance.html)

Modo estricto

El modo estricto se aplica a un script por completo o a funciones individuales y permite asegurar el código contenido en una función y realizar comprobaciones adicionales. Puede ser activado mediante el string **"use strict"** tras la apertura de la misma, como primera instrucción

- Primero, elimina algunos errores silenciosos de JavaScript haciendo que lancen excepciones.
- Segundo, corrige errores que hacen que sea difícil para los motores de JavaScript realizar optimizaciones: a veces, el código escrito en modo estricto puede correr más rápido que el que no es estricto.
- Tercero, el modo estricto prohibe cierta sintaxis que es probable que sea definida en futuras versiones de ECMAScript.

Más detalles en referencia de Mozilla Developer Network

FUNCIONES PREDEFINIDAS / GLOBALES

Además de las funciones personalizadas, Javascript dispone de un juego de funciones predefinidas que forman parte del motor del lenguaje y recogen acciones usuales durante el desarrollo:

- **eval()** evalúa el código JavaScript representado como una cadena de caracteres
- **isFinite()** determina si el valor pasado es un número finito, si se necesita el parámetro primero es convertido a número.
- **isNaN()** evalúa un argumento para determinar si es "NaN" (no es un número).
- **parseInt**() y **parseFloat**(), devuelven un valor numérico cuando se les da una cadena como argumento.
- **Number**() y **String**() permiten convertir un objeto a un número o una cadena.

SELECTORES

Para acceder al DOM desde la aplicación y manipularlo o consultarlo se hace uso de los denominados **manejadores de eventos** (*event handlers*), también llamados **selectores.**

Un selector puede acceder a un objeto u objetos del DOM para:

- Aplicar sobre ese/esos objeto/s cualquiera de los métodos o propiedades que acepte
- Asociar al/los objeto/s un evento que sea reconocido por la aplicación

Los manejadores de eventos semánticos o selectores son los siguientes (ver selectores.html)

Selector	**Descripción**
document.getElementById();	**Devuelve un único objeto** del DOM que tenga como valor del ID el string indicado como argumento, null si no existe.
valorId	Funciona de forma idéntica al anterior, pero en vez de indicar el valor del ID como argumento, se menciona directamante
document.getElementsByClassName();	**Devuelve una colección (Array)** de elementos del DOM que tengan alguna de las clases indicadas como argumento, null si no existe. Cada clase se separa por un espacio. El orden en el que son almacenados es el orden en el que aparecen en el DOM.
document.getElementsByTagName();	**Devuelve una colección (Array)** de elementos del DOM que tengan la etiqueta de HTML indicada como argumento, null si no existe. El orden en el que son almacenados es el orden en el que aparecen en el DOM. Previo a realizar la búsqueda en el DOM, convierte el valor del argumento a lowercase.
document.querySelector();	Devuelve el primer objeto del selector indicado como argumento, null si no existe. Los selectores avanzados CSS3 son válidos. Los pseudo selectores no son válidos, devolviendo un NodeList vacío. Este método es menos eficiente que los tres primeros.
document.querySelectorAll();	Devuelve una objeto NodeList del selector o selectores indicados como argumento, null si no existe. Los selectores avanzados CSS3 son válidos, y se separan entre comas. Los pseudo selectores no son válidos, devolviendo un NodeList vacío. Este método es menos eficiente que los tres primeros.

EVENTOS

Un evento es una situación que puede ser detectada por la aplicación, abarcando tanto acciones realizadas por parte del usuario (hacer click en un elemento) como situaciones por parte del entorno (detectar cuándo la página se ha cargado completamente o cuándo se ha redimensionado la ventana de navegación).

Podemos asociar eventos a objetos del DOM de tres formas:

- Como manejadores de atributo en objetos de XHTM (ver eventos1.html)
- Como funciones externas (ver eventos2.html)
- **Como manejadores "semánticos" haciendo uso de un selector (Recomendado) (ver eventos3.html)**

En la Programación Orientada a Eventos, cada elemento o etiqueta XHTML (objeto) del DOM define su propia lista de posibles eventos que puede registrar.

Evento	Descripción	Elementos para los que está definido
onblur	Deseleccionar el elemento	<button>, <input>, <label>, <select>, <textarea>, <body>
onchange	Deseleccionar un elemento que se ha modificado	<input>, <select>, <textarea>
onclick	Pinchar y soltar el ratón	Todos los elementos
ondblclick	Pinchar dos veces seguidas con el ratón	Todos los elementos
onfocus	Seleccionar un elemento	<button>, <input>, <label>, <select>, <textarea>, <body>
onkeydown	Pulsar una tecla (sin soltar)	Elementos de formulario y <body>
onkeypress	Pulsar una tecla	Elementos de formulario y <body>
onkeyup	Soltar una tecla pulsada	Elementos de formulario y <body>
onload	La página se ha cargado completamente	<body>
onmousedown	Pulsar (sin soltar) un botón del ratón	Todos los elementos
onmousemove	Mover el ratón	Todos los elementos
onmouseout	El ratón "sale" del elemento (pasa por encima de otro elemento)	Todos los elementos
onmouseover	El ratón "entra" en el elemento (pasa por encima del elemento)	Todos los elementos
onmouseup	Soltar el botón que estaba pulsado en el ratón	Todos los elementos
onreset	Inicializar el formulario (borrar todos sus datos)	<form>
onresize	Se ha modificado el tamaño de la ventana del navegador	<body>
onselect	Seleccionar un texto	<input>, <textarea>
onsubmit	Enviar el formulario	<form>
onunload	Se abandona la página (por ejemplo al cerrar el navegador)	<body>