Funciones

Antonio Espín Herranz

Funciones (1)

- Una función JavaScript contiene código que será ejecutado sólo por un evento o por una llamada a esa función
- Una función puede ser llamada desde cualquier sitio en la página
- Se pueden definir tanto en la sección `<head>` como en el `<body>`
  - Para asegurar que la función es leída por el navegador antes de ser llamada, mejor colocarla en el `<head>`

Funciones (2)

- Ejemplo:

```
<html>
<head>
    <script type="text/javascript">
        function displayMessage(){
            alert("Hola mundo!");
        }
    </script>
</head>
<body>
    <form>
        <input type="button" value="Click me!"
               onclick="displayMessage();">
    </form>
</body>
</html>
```

  - La función se ejecuta al hacer *click* en el botón (evento `onclick`)
  - Si el código `alert("Hola mundo!");` no estuviera dentro de una función, sería ejecutado tan pronto se cargase en la página (antes incluso de "pintarse" el botón)

Funciones (3)

- Definición de una función:
  - Con parámetros

```
function nombreFuncion(var1,var2,...,varX) {
   ...
   codigo
   ...
}
```

  - Sin parámetros

```
function nombreFuncion() {
   ...
   codigo
   ...
}
```

Funciones (4)

- Sentencia de retorno:
  - Cuando la función devuelve algún valor, se utiliza **`return`**
  - Ejemplo:

```
function producto(op1, op2) {
   return op1 * op2;
}
```

```
tresPorDos = producto(3,2);
```

Funciones (5)

- Una función puede tener un número variable de argumentos
  - En el siguiente ejemplo, se puede llamar tanto a `miFuncion()` como a `miFuncion(20)`

```
function miFuncion(valor){
    if (valor){
        this.area=valor;
    }
    return this.area;
}
```

Funciones anidadas

```
function generarListaOrdenada(vec)
 {
  function comienzo()
  {
   s='<ol>';
  }

  function fin()
  {
   s=s+'</ol>';
  }

  var s='';
  comienzo();
  var f;
  for(f=0;f<vec.length;f++)
  {
   s=s+'<li>'+vec[f]+'</li>';
  }
  fin();
  return s;
 }
```

```
var s='';
  comienzo();
  var f;
  for(f=0;f<vec.length;f++)
  {
   s=s+'<li>'+vec[f]+'</li>';
  }
  fin();
  return s;
 }
```

Múltiples parámetros

```
// Múltiples parámetros en una función de JS

function imprimir(p1, p2, ...params){
        console.log("p1: "+p1)
        console.log("p2: "+p2)
        console.log("...params: "+params)
        console.log('\n')

        params.forEach( p => console.log(p))
}


imprimir(1,2,3,4,5,6)
```

Funciones JS predefinidas

- **eval(exp)**:
  - La función eval tiene como argumento una expresión y devuelve el valor de la misma. Esta función resulta útil para evaluar una cadena de caracteres que representa una expresión numérica.
- **escape / unescape: (deprecated)**
  - Estas dos funciones permiten codificar cadenas de caracteres en formato URL (ISO Latin 1). Esta codificación es necesaria en la creación automática de enlaces de hipertexto o en la definición de propiedades persistentes como los Cookies.
  - *escape("He aquí")="He%20aquí"*
    unescape("He%20aquí")="He aquí"
  - *escape("#"); // devuelve %23*
    unescape("%23"); // devuelve #
- **encodeURIComponent() / decodeURIComponent()**
  - console.log(`?x=${encodeURIComponent('test?')}`);
  - // expected output: "?x=test%3F"

Funciones JS predefinidas

- **parseFloat(str)**
  - Convierte un string a un número en punto flotante. Si se encuentra otros caracteres que no sean números, el signo '+', el '-' o un exponente, devuelve el valor encontrado hasta ese punto. Del mismo modo, si el primer caracter no se puede convertir a número devolverá cero.
- **NaN(exp)**
  - Comprueba si el valor pasado por parámetros es númerico o no. El resultado de esta función es un booleano. Es decir, evalúa un argumento para ver si es NaN: Not Number.
- **parseInt(str)**
  - Convierte una cadena de caracteres de entrada a un número entero con una base especificada. La base puede ser 8, 10 ó 16. Si se encuentra otros caracteres que no sean números, el signo '+', el '-' o un exponente, devuelve el valor encontrado hasta ese punto. Del mismo modo, si el primer carácter no se puede convertir a número devolverá cero.

Ejemplo

```
function Comprueba(form){
    var number = parseFloat(form.valor.value);
    if (isNaN(number)==true)
        alert("No es numérico");
    else {
        form.valor.value = number;
        alert("Es numérico");
    }
}
```