MF0951_2-Integración componentes software en páginas web

UF1305: Programación lenguajes de guión

Generalitat de Catalunya
Departament d'Empresa i Ocupació

MINISTERIO DE TRABAJO E INMIGRACIÓN
SERVICIO PÚBLICO DE EMPLEO ESTATAL

Unió Europea
Fons social europeu
L'FSE inverteix en el teu futur

Contenido

Java y Javascript

Java es un lenguaje de programación (como el VisualBASIC o el C y C++) que fue desarrollado por la empresa **Sun** fundamentalmente para crear aplicaciones para Internet. El lenguaje Java es complejo, es decir permite realizar cualquier operación sobre el ordenador (como por ejemplo borrar un archivo) y su aprendizaje es costoso.

Javascript es lo que se conoce como **lenguaje script**, es decir: se trata de código de programación que se inserta dentro de un documento. Javascript fue desarrollado por la empresa **Netscape** con la idea de potenciar la creación de páginas Web dinámicas para su navegador **Navigator.**

Javascript (en contra de lo que se podría suponer) es totalmente distinto de Java. Java crea programas totalmente independientes y operativos; Javascript es más sencillo porque lo único que permite es insertar código especial dentro del HTML de una página, su función es ampliar las posibilidades de HTML. Javascript no crea programas independientes, dependen por completo del código HTML de la página.

La ventaja fundamental de Javascript es que su aprendizaje y uso son más sencillos comparado con los lenguajes de programación para desarrollo de software o aplicaciones mobile y que permite realizar labores complejas en una página sin necesidad de aprender CGI.

Versiones de Javascript

Puesto que JavaScript fue desarrollado por Netscape, los navegadores de esta empresa lo incluyen desde la versión 2. Microsoft por su parte incluyó en la versión 3 una variante de este código llamado **JScript** que es casi idéntico al original JavaScript. Después se estandarizó el lenguaje, aunque ambas compañías poseen elementos que no son comunes con el estándar (aunque prácticamente todo el estándar es reconocido por ambas).

Así aparición el JavaScript 1.1 que es admitido por Navigator 3 y por Explorer 4. Y las versiones 1.2 y 1.3 que son reconocidas por las versiones 4 y posteriores de ambos navegadores.

La ECMA (asociación industrial para la normalización) definió un lenguaje estándar llamado **ECMAScript** que intentaba agrupar a los anteriores e incluía instrucciones nuevas

inclusión de Javascript en las páginas

Para hacer que un documento HTML incluya instrucciones en Javascript se debe hacer uso de la etiqueta **<SCRIPT>** de esta forma:

```
<script language="JavaScript">
código JavaScript
</script>
```

Si se quiere especificar qué versión de Javascript se utiliza, para evitar que navegadores que no soportan la versión decodifiquen el Javascript, entonces se usa, por ejemplo:

```
<script language="JavaScript1.3">
```

Navegadores no compatibles

Los navegadores que no soportan Javascript, no interpretarían las instrucciones Javascript sino que mostrarían el texto de las instrucciones en la página. Para evitar que estos navegadores lean el código en Javascript se hace:

```
<script language="Javascript">
<!-
código JavaScript
//-->
</script>
```

El signo "<!-" indica principio de comentario en HTML y el signo "-->" indica fin de comentario. A su vez el signo "//" indica comentario en Javascript (el intérprete de Javascript no tendrán en cuenta esa línea).

Uso de un archivo externo

También se puede utilizar el código JavaScript escrito en un archivo separado. Este archivo debe tener la extensión **js.** En el archivo se coloca sólo código en JavaScript. Después ese código se puede invocar desde la página web con el código:

```
<script language="Javascript" src="archivo.js">
```

Normas de escritura en Javascript

- Los comentarios deben empezar con el símbolo **//** si son de una sola línea o iniciarse con los símbolos /* y finalizar con */ si son de varias líneas.
- Las líneas de código terminan con el signo de punto y coma (;)
- Javascript distingue entre mayúsculas y minúsculas
- Las llaves ({ y }) permiten agrupar código.

Variables

Una variable es un elemento que tiene un determinado nombre y que permite almacenar valores.

nombre de las variables

Deben empezar con una letra la cual puede ir seguida de números, el signo "_" o más letras.

valores

Los valores que pueden asignarse a una variable pueden ser:

- Cadenas de texto: "esto es una prueba", 'prueba' o "esto es una 'prueba' de código". Siempre se encierran entre comillas dobles o simples. Una variable de texto que no tiene contenido, se dice que tiene valor null. La palabra null es un término reconocido por Javascript.
- Valores numéricos: 1, -100, 1.6, 2.0E2.
- Valores booleanos: true o false.

caracteres especiales

Los valores de tipo texto van entre comillas y dentro de ellos se pueden colocar caracteres especiales (caracteres que no se pueden ver, como el cambio de línea) los cuales son:

- \a: Alarma
- \b: Retroceso (cursor una posición hacia atrás).
- \f: Nueva página de impresora
- \n: Nueva línea
- \r: Retorno de carro
- \t: Tabulador

@ \\: Signo "\"

Declaración de una variable

Para declarar una variable se puede emplear:

```
var variable = valor;
```

O simplemente:

```
variable = valor;
```

De tal modo, que realmente en Javascript no hace falta declarar una variable antes de su uso. Ejemplos:

```
var testear = 0; testeaTexto = "Mi casa"; SeleccionarColor = true;
```

JavaScript permite que una variable pueda almacenar distintos tipos de datos en cada trozo de código. Es decir, una variable que ahora almacena texto, después puede almacenar números.

Tras declarar la variable, su valor puede cambiar mediante la asignación de un valor:

```
Testear = 12.3;
```

O mediante la asignación del resultado de una operación:

```
Testear = 12 * 3 + varX;
```

Conversión de datos

En muchas lenguajes si una variable toma valores de texto y luego se quiere hacer que tome números, resulta imposible hacerlo. No es el caso de JavaScript ya que realiza conversiones implícitas. Ejemplo:

```
var x="50" //x es una variable de texto
var y=30   //y es una variable numérica
z1=x+y     //z1 es variable de texto y vale "5010"
z2=y+x     //z2 es numérica y vale 60
/*dependiendo de cuál sea el primer operando, se determina el tipo del resultado*/
```

Naturalmente ocurrirá un error si pretende convertir a un número, un texto normal como "Hola" por ejemplo. En cualquier caso no conviene hacer conversiones de tipo en ningún caso.

Operadores

Los operadores son los elementos que permiten realizar operaciones con los datos del código.

operadores aritméticos

Operador	Significado
+	Suma
-	Resta
*	Multiplicación
/	Dividir
%	Resto de la división
++	Incremento
--	Decremento

Ejemplo:

```
var valor1=50; var valor2=10; var valor3=20;
var suma, resta, producto, division, resto; var incremento, decremento;
suma=valor1+valor2; //suma vale 60
resta=valor1-valor2;//resta vale 40
producto=valor1*valor2;    //producto vale 5000
division=valor1/valor3;     //division vale 2,5
resto=valor1%valor3; //resto vale 10

valor1++; //valor1 vale 51 valor1--; //valor1 vale 50
```

incremento=valor1++; //incremento vale 50 y valor1 vale 51 decremento=valor1--; //decremento vale 51, valor1 vale 50

incremento=++valor1; //incremento vale 51 y valor1 también decremento=--valor1; //decremento y valor1 valen 50

Operadores lógicos

Trabajan con proposiciones matemáticas (valores boléanos) son:

Operador	Significado
&&	AND (Y lógico)
||	OR (O lógico)
!	NOT (NO lógico)

Operadores de comparación

Son:

Operador	Significado
==	Igual
!=	Distinto
>=	Mayor o igual
<=	Menor o igual
>	Mayor
<	Menor

Operadores de asignación

Son:

Operador	Ejemplo
+=	Suma y asignación
-=	Resta y asignación
*=	Producto y asignación
/=	División y asignación
%=	Resto y asignación

Mensajes navegador en JavaScript

¿qué son?

Se trata de ventanas que desde el código se lanzan al usuario para hacer que éste reaccione ante una situación o nos informe ante una duda. Realmente todos los mensajes se obtienen a través del objeto window (véase más adelante).

alert

Es el mensaje más usado. Saca un mensaje por la pantalla el cual sólo deja la posibilidad de aceptarle. Su uso es mostrar información al usuario pero resaltándola de la página. Su sintaxis es:

```
alert(texto_del_mensaje);
```

prompt

Actualmente en desuso. En este caso se trata de una ventana que pide entrar datos al usuario. De modo que esta función devuelve un valor que se puede usar en el código si es asignado a una variable. Su sintaxis es:

confirm

Saca un mensaje de confirmación el cual suele tener dos botones: Aceptar y Cancelar. Sintaxis:

```
confirm(texto_del_mensaje)
```

La ventana mostrará el texto elegido (normalmente es una pregunta) y el usuario elegirá si desea aceptar o no el contenido. Confirm devuelve un valor true en el caso de que el usuario acepte el mensaje, y false si no lo hace.

Estructuras condicionales

Introducción

Las estructuras de este tema son sentencias que permiten tomar decisiones dentro del código a fin de devolver un resultado u otro dependiendo de una determinada circunstancia que es la que se evalúa.

instrucción if

La instrucción IF realiza lo que se denomina un sí lógico. Su forma es:

```
if(condición) {
..código que se ejecuta si la condición es cierta
}
```

También admite esta otra:

```
if(condición) {
..código que se ejecuta si la condición es cierta
}else{
... ..código que se ejecuta si la condición es falsa
}
```

Se admite dentro de una instrucción IF, colocar otra instrucción IF. A esto se le llama “anidar” condiciones if.

bucle while

Un bucle es una estructura de programación que permite repetir sentencias hasta que se cumpla una determinada condición. Su forma es:

```
while(condicion){
... sentencias que se ejecutan mientras la condición se cumpla
}
```

Ejemplo:

```
var x=1; while(x<11)
{
document.write(x,” “); x++;
}
// Sale 1 2 3 4 5 6 7 8 9 10
```

bucle for

Su efecto es muy similar a la anterior estructura. Permite ejecutar una serie de sentencias hasta que se cumpla una determinada condición.. Su estructura es:

```
for(valor_inicial; condición; actualización)
{
..sentencias que se ejecutan mientras la condición se cumpla
}
```

Ejemplo:

```
for(x=1;x<11;x++)
{
document.write(x,” “);
```

```
}
// Sale 1 2 3 4 5 6 7 8 9 10
```

break

Es una instrucción que hace que el navegador que se la encuentra, abandone inmediatamente el bucle en el que está inmerso.

continue

Es parecida a la anterior, sólo que en lugar de abandonar el bucle, lo que hace el navegador es dejar de leer las siguientes instrucciones del bucle y saltar al principio del mismo.

Instrucción switch

Esta instrucción permite un mayor control sobre las condiciones, lo malo es que se incluyó en la versión 1.2 de JavaScript por lo que sólo los navegadores con versión 4 o posterior los pueden usar.

Su sintaxis es:

```
switch (objetodeanálisis) { case valor1: ..instrucciones case
valor2:...instrucciones

....

default: instrucciones

}
```

Switch funciona de esta forma: en los paréntesis se coloca una expresión a evaluar, y en cada apartado case, se coloca un posible valor de la expresión. Los valores que se cumplan harán que se ejecuten las instrucciones que les siguen. En caso de que no cumpla ninguna se ejecutarían las correspondientes al default (no es obligatorio poner este apartado). Ejemplo:

```
var x=12;
```

```
switch(x) {
case 4:document.write("Es cuatro");break; case
8:document.write("Es ocho");break; case 12:document.write("Es
doce");break;
default: document.write("No es ninguna de las anteriores")
}
```

Hace falta poner la instrucción break ya que de otro cuando se encuentra el valor que cumple la expresión, se ejecuta su "case" y los "case" siguientes.

Funciones y Objetos

Funciones

Como en otros muchos lenguajes, en Javascript se pueden utilizar funciones. Las funciones son una serie de instrucciones que realizan una determinada tarea. A las funciones se las pone un nombre que luego puede ser utilizado en el código de la página.

Definición de una función

Antes de poder usar una función en el código de la página, se la debe definir; es decir, se debe indicar qué operaciones son las que debe hacer la función. La definición de la función es:

```
function nombredelafunción(argumento1, argumento2,...)
{
instrucciones que debe realizar la función
}
```

El código que está encerrado entre llaves indica lo que realiza la función (por ejemplo mostrar un mensaje de ayuda), cada vez que

desde el código se llame a la función, ésta realizará sus instrucciones.

Por otro lado los argumentos son variables que algunas funciones necesitan para realizar su tarea

llamar a una función

Para usar (invocar) una función en el script, basta con poner su nombre seguido de los paréntesis.

Ejemplo:

```
function error() {
document.write("<b>Ocurrió un error</B><BR>");
}
```

En este caso se define una función que muestra texto en la posición actual del cursor. Para utilizar esta función desde el código sería:

```
error();
```

Lo cual mostraría el mensaje de error.

Otro ejemplo (página HTML completa):

```
<HTML>
<HEAD>
<TITLE>Titulo</TITLE>
<SCRIPT LANGUAGE=”JavaScript”>
<!--
function doblar(valor) {
return valor * 2;
//-->
</SCRIPT>
<BODY>
<SCRIPT LANGUAGE=”JavaScript”>
document.write(“El doble de 100 es ”,doble(100));
</SCRIPT>
</BODY>
</HTML>
```

En este caso la función doblar(), posee un argumento (valor), el cual es necesario ya que esta función calcula el doble de ese número. Además esta función devuelve un valor que el código puede utilizar, eso lo realiza la instrucción return, la cual sirve para regresar un valor que el código puede mostrar o asignar a una variable.

Funciones predefinidas

JavaScript trae consigo muchas funciones predefinidas. Señalamos aquí algunas de las más importantes:

◉ eval(textoCódigo). La función eval tiene un único parámetro que es una cadena de texto. Esta función hace que el texto sea interpretado como si fuera código normal de JavaScript. Ejemplo:

eval("alert('Hola')");

◉ parseInt(textoNúmero, base). Convierte el texto (que debe tener cifras numéricas) a formato de número. El segundo parámetro es opcional y representa la base del número, ejemplo:

alert(parseInt("110011",2));//Sale 51

Si la conversión no es posible, devuelve el valor NaN (Not a Number) que indica que la variable numérica posee un valor inválido

◉ parseFloat(textoNúmero). Convierte el texto (que debe tener cifras numéricas) a formato de número con decimales.

// Escribe: ¡•

◉ isNaN(expresión). Devuelve true si la expresión tiene un contenido no numérico.

Objetos

Los objetos son una de las bases fundamentales de JavaScript. Un objeto es una agrupación de variables, que en ese caso se llaman propiedades, y de funciones, las cuales se llaman métodos. Las propiedades definen las características de los objetos y los métodos las operaciones que podemos hacer con él.

JavaScript posee muchos objetos predefinidos y también permite crear nuestros propios objetos.

propiedades

Como ya se comentó anteriormente, los objetos poseen propiedades asociadas, para acceder a ellas se utiliza el punto, en esta forma:

objeto.propiedad

Ejemplo, un objeto llamado miordenador que representa a un ordenador:

miordenador.marca=”HP”; miordenador.procesador=”pentium III 900 Mhz” miordenador.ram=64;

Las propiedades pueden ser de distintos tipos de datos; en el ejemplo la propiedad

ram es numérica, mientras que marca es de tipo texto.

métodos

Los métodos son funciones asociadas a los objetos. Su uso es:

objeto.metodo()

Donde el método además puede poseer parámetros.

operación new

La instrucción new sirve para crear nuevos objetos en el tiempo de ejecución del código JavaScript. Ejemplo:

```
miordenador = new ordenador(“HP”, “Pentium III”, 64);
```

En este caso mi ordenador es un nuevo objeto de tipo ordenador, al indicar el tipo de objeto que es, se le pueden pasar parámetros para rellenar sus propiedades.

jQuery Introducción

El propósito de jQuery es para que sea mucho más fácil de usar JavaScript en una página web.

¿Qué es jQuery?

jQuery es una biblioteca (Framework) de Javascript pensada para programar de forma más ligera, "escribir menos, hacer más".

El propósito de jQuery es para que sea mucho más fácil de usar JavaScript en una página web.

jQuery toma una gran cantidad de tareas comunes que requieren muchas líneas de código JavaScript para llevar a cabo, y los envuelve en los métodos que se pueden llamar con una sola línea de código.

jQuery también simplifica mucho las cosas complicadas de JavaScript, como las llamadas AJAX y la manipulación del DOM.

La librería jQuery contiene las siguientes características:

- HTML / DOM manipulación
- manipulación de CSS
- HTML
- Efectos y animaciones
- AJAX
- Utilidades

Consejo: Además, jQuery tiene plugins para casi cualquier tarea que hay.

¿Por qué jQuery?

Hay un montón de otros marcos de JavaScript por ahí, pero jQuery parece ser el más popular, y también la más extensible.

Muchas de las empresas más grandes en la Web utiliza jQuery, tales como Google, Microsoft o Netflix:

- Google
- MicrosoftNetflix

jQuery- declaración para su uso

Para agregar jQuery framework a sus páginas web

Hay varias formas de iniciar el uso de jQuery en su sitio web. Puede:

- Descargar la librería jQuery desde JQuery.com
- Incluir jQuery de un CDN, como Google

Descarga de jQuery

Hay dos versiones de jQuery disponible para descarga:

• Versión de producción – la más utilizada, ya que viene optimizada (comprimida y minificada) la biblioteca para cargar más rápido.

• Versión de desarrollo - esto es para pruebas y desarrollo (sin comprimir y código legible para el humano)

Ambas versiones se pueden descargar desde jQuery.com.

La librería jQuery es un solo archivo JavaScript, además de la referencia con el código HTML etiqueta

¿Se pregunta por qué no tenemos type = "text / javascript" dentro de la etiqueta <script>?

Esto no es necesario en HTML5. JavaScript es el lenguaje de scripts por defecto en HTML5 y en todos los navegadores modernos!

jQuery CDN

Si no desea descargar y albergar jQuery en tu proyecto, se puede incluir desde un servidor externo de tipo CDN (Content Delivery Network).

Tanto Google como Microsoft disponen de jQuery en sus CDN.

Para utilizar jQuery de Google o Microsoft, utilice uno de los siguientes:

Google CDN:

<head> <script src = "http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js "> </ scr ipt> </ head>

MicrosoftCDN: <head> <script src =" http://ajax.aspnetcdn.com/ajax / jQuery / jquery- 1.11.2.min.js "> </ script> </ head>

Una gran ventaja de utilizar el jQuery alojado desde Google o Microsoft:

Muchos usuarios ya han descargado jQuery de Google o Microsoft se encuentra en otro sitio. Como resultado, se cargará desde la memoria caché cuando visitan su sitio, lo que conduce a la descarga más rápida. Además, la mayoría de CDN se asegurará de que una vez que un usuario solicita un archivo de ella, se sirve desde el servidor más cercano a ellos, que también conduce a la descarga más rápida.

jQuery Sintaxis

Con jQuery selecciona (consulta) elementos HTML y realizar "acciones" en ellos.

La sintaxisLa jQuery está hecha a medida para la selección de los elementos HTML y realizar alguna acción sobre el elemento (s).

Sintaxis básica es: $ (selector) .action ()

- Un signo $ para definir / acceso jQuery
- Un (selector) para "consulta (o encontrar)" elementos HTML
- Una acción de jQuery () que se realizará en el elemento (s)

Ejemplos:

$ (this) .hide () - oculta el elemento actual (al que se le ha hecho clic, por ejemplo).

$ ("p") hide () -. oculta todo elementos <p>.

$ (". test") hide () -. oculta todos los elementos con class = "prueba".

$ ("# test") hide () -. oculta el elemento con id = "prueba".

¿Y con los CSS? jQuery utiliza la sintaxis CSS para seleccionar elementos.

El evento Documento Preparado

Muchos de los códigos de ejemplo que podemos encontrar por internet están declarados dentro del evento $(document).ready:

$ (document) ready (function ()

{//métodos jQuery van aquí

...});

Esto es para evitar cualquier código jQuery se ejecute antes de que el documento haya terminado de cargarse con todos los elementos Html, Css y Javascript princpalmente.

Es una buena práctica que esperar a que el documento esté completamente cargados listo antes de trabajar con él. Esto también le permite tener su código JavaScript(jQuery) antes del

cuerpo del documento, en la sección del “head”.

Estos son algunos ejemplos de acciones que pueden fallar si los métodos se ejecutan antes que el documento se ha cargado completamente:

• Tratando de ocultar un elemento que no se ha creado todavía

• Tratar de obtener el tamaño de una imagen que no está cargado todavía

Tip: El equipo de jQuery también se ha creado un método aún más corto para el evento ready documento:

```
$ (function ()
  {//métodos jQuery van aquí ...});
```

jQuery-selectores

Los selectores de jQuery son una de las partes más importantes de la biblioteca jQuery.

Los selectores de jQuery permiten seleccionar y manipular elemento (s) de HTML.

Los selectores de jQuery se utilizan para "encontrar" (o seleccionar) elementos HTML en función del ID, clases, tipos, atributos, valores de atributos y mucho más. Está basado en los selectores CSS existentes y, además, tiene algunos selectores personalizados propios.

Todos los selectores de jQuery comienzan con el signo de dólar y paréntesis: $ ().

El elemento Selector

El selector de elementos jQuery selecciona elementos basados en el nombre del elemento.

Puede seleccionar todos los elementos <p> en una página como esta:

```
$ ( "p")
```

Ejemplo

Cuando un usuario hace clic en un botón, todos los <p> se ocultarán: Ejemplo:

```
$ (document) ready (function () { .
   $ ( "botón") haga clic (function ()
  {
  $( "p") hide ();.});
});
```

El Selector de #id

El selector de jQuery #id utiliza el atributo id de una etiqueta HTML para encontrar el elemento específico.

Un identificador debe ser único dentro de una página, por lo que debe utilizar el selector #id cuando se quiere encontrar un solo elemento, único.

Para encontrar un elemento con un ID específico, escribe un carácter almohadilla, seguido por el id del elemento HTML:

```
$ ( "# test")
```

Ejemplo

Cuando un usuario hace clic en un botón, el elemento con id = "prueba" se ocultará Ejemplo:

```
$ (document) ready (function ()  $ ( "button").click(function ()
  {..{$( "# test") hide ();});});
```

El Selector .class

El selector de clase jQuery encuentra elementos con una clase específica.

Para encontrar elementos con una clase específica, escribe un carácter de punto, seguido del nombre de la clase:

```
$ (".test")
```

Ejemplo

Cuando un usuario hace clic en un botón, los elementos con class =

"test" se ocultarán: Ejemplo

$ (document) ready (function () {$ ("botón") haga clic (function () ();..});});

{$(".test"). hide()

Más ejemplos de jQuery Selectores

Sintaxis Descripción Ejemplo

$("*") selecciona todos los elementos.

$(this) Selecciona el elemento HTML actual.

$("p.intro") Selecciona todos los elementos <p> con class = "intro".

$ ("p: first") Selecciona el primer elemento <p>.

$ ("ul li:first") Selecciona la primera <li> elemento de la primera <ul>.

$ (" ul li:first-child ") Selecciona la primera <li> de cada <ul>.

$ (" [href] ") Selecciona todos los elementos con un atributo href.

$ (" a [target = '_ blank'] ") selecciona todos <a> elementos con un objetivo atribuyen valor igual a "_blank"

$ ("a [target! = '_ blank']") selecciona todos los elementos <a> con atributo target NO es igual a "_blank"

$ (":button") Selecciona todos los <button> y elementos <input> de type button.

$ ("tr:even") Selecciona todos los pares <tr>.

$ ("tr:odd") Selecciona todos los impares <tr>.

Código JavaScript/jQuery en archivo externo

Si un sitio web contiene una gran cantidad de páginas, y deseas que sus funciones de jQuery sean fáciles de mantener/actualizar, se puede declarar las funciones jQuery en un archivo .js aparte.

Es preferible colocar todo el código jQuery en un archivo separado, con el atributo src en la etiqueta script se declara la: Ejemplo

<head>

<script src = "http://ajax.googleapis.com/ajax

```
/libs/jquery/1.11.2/jquery.min.js "> </ script> <script src ="
my_jquery_functions.js "> </ script>
</ head>
```

Eventos JQuery

jQuery está hecho a medida para responder a eventos en una página HTML.

¿Cuáles son eventos?

Todas las acciones de los diferentes visitantes que una página web puede responder a eventos que se llaman.

Un evento representa el momento preciso en que algo sucede.

Ejemplos:

- mover el ratón sobre un elemento
- seleccionar un botón de radio
- Al hacer clic sobre un elemento

El término "dispara" a menudo se utiliza con los eventos. Ejemplo: "El evento de pulsación de tecla dispara el momento en que se pulsa una tecla".

Aquí están algunos eventos DOM comunes:

Eventos con el ratón.

Eventos de teclado.

Eventos de formulario

Eventos de Documento / Ventanas.

Eventos de clic, pulsación de tecla

Evento de al cargar (onload)

jQuery sintaxis para Métodos de Evento

En jQuery, la mayoría de los eventos DOM tienen un método jQuery equivalente.

Para asignar un evento de clic a todos los párrafos de una página, puede hacer esto:

```
$("p").click();
```

El siguiente paso es definir lo que debe ocurrir cuando se activa el evento. Se debe pasar una función al evento:

```
$("p").click(function(){
 // el código de lo que debe hacer va aquí!!
 });
```

JQuery Métodos de Evento más usado comúnmente

- **$ (document) .ready()**

El método $ (document) ready () nos permite ejecutar una función cuando el documento se ha cargado completamente. Este evento ya se ha explicado en el capítulo jQuery Sintaxis.

- **Hacer click ()**

El método click () atribuye una función de controlador de eventos para un elemento HTML.

La función se ejecuta cuando el usuario hace clic en el elemento HTML.

El siguiente ejemplo se dice: Cuando un evento de clic incendios en un elemento <p>; ocultar el elemento actual <p>:

Ejemplo

```
$("p").click(function(){
  $(this).hide(); });
```

- **dblclick ()**

El método dblclick () atribuye una función de controlador de eventos para un elemento HTML.

La función se ejecuta cuando el usuario hace doble clic en el elemento

Ejemplo

```
$("p").dblclick(function(){
  $(this).hide(); });
```

- **MouseEnter ()**

El método MouseEnter () atribuye una función de controlador de eventos para un elemento HTML.

La función se ejecuta cuando el puntero del ratón entra en el elemento HTML:

Ejemplo

```
$("#p1").mouseenter(function(){
  alert("You entered p1!"); });
```

- **mouseleave ()**

El método mouseleave () atribuye una función de controlador de eventos para un elemento HTML.

La función se ejecuta cuando el puntero del ratón sale del elemento HTML:

```
$("#p1").mouseleave(function(){
  alert("Adiós! Acabas de salir del elemento p1!"); });
```

- **mousedown ()**

El método mousedown () atribuye una función de controlador de eventos para un elemento HTML.

La función se ejecuta, cuando se pulsa el botón izquierdo del ratón, mientras que el ratón está sobre el elemento HTML:

Ejemplo

```
$("#p1").mousedown(function(){
  alert("Mouse down over p1!"); });
```

- **mouseup ()**

El método mouseup () atribuye una función de controlador de eventos para un elemento HTML.

La función se ejecuta, cuando se suelta el botón izquierdo del ratón, mientras que el ratón está sobre el elemento HTML:

Ejemplo

```
$("#p1").mouseup(function(){
  alert("Mouse up over p1!"); });
```

- **hover ()**

El método de la libración () realiza dos funciones y es una

combinación de los métodos () MouseEnter () y mouseleave.

La primera función se ejecuta cuando el ratón entra en el elemento HTML, y se ejecuta la segunda función cuando el ratón sale del elemento

Ejemplo

```
$("#p1").hover(function(){
  alert("You entered p1!"); }, function(){
  alert("Bye! You now leave p1!"); });
```

- **focus ()**

El método de enfoque () atribuye una función de controlador de eventos a un campo de formulario HTML.

La función se ejecuta cuando el campo de formulario se centrará:

Ejemplo

```
$ ( "input") se centran (function () "#cccccc");});.
  {$(this) .css ( "background-color",
```

- **blur ()**

El método de blur () atribuye una función de controlador de eventos a un campo de formulario HTML.

La función se ejecuta cuando el campo de formulario pierde el foco:

Ejemplo

```
$("input").focus(function(){
  $(this).css("background-color", "#cccccc"); });
```

jQuery Efectos - ocultar y mostrar

jQuery hide () y show ()

Con jQuery, puede ocultar y mostrar los elementos HTML con la hide () y show ()

Ejemplo:

```
$("#hide").click(function(){$("p").hide(); });
$("#show").click(function(){$("p").show(); });
```

Sintaxis:

$ (selector) .hide (velocidad, devolución de llamada);

$ (selector) .show (velocidad, devolución de llamada);

El parámetro de velocidad opcional especifica la velocidad de la Ocultación / muestra, y puede tomar los siguientes valores: "lento", "rápido", o milisegundos.

El parámetro callback opcional es una función a ejecutar después de que el método de ocultar () o Mostrar () completa (aprenderá más sobre las funciones de devolución de llamada en un capítulo posterior).

jQuery toggle ()

En jQuery, cuando se cambia de la hide () y show () es el método de toggle ().

Elementos que se muestran están ocultos y se muestran los elementos ocultos:

Ejemplo

```
$("button").click(function(){
$("p").toggle(); });
```

Más parámetros:

$ (selector) .toggle (speed, callback);

El parámetro de velocidad (speed) es opcional puede tomar los siguientes valores: "lento", "rápido", o valor en milisegundos.

El parámetro callback opcional es una función a ejecutar después de toggle () se ha completado.

Ejemplo:

```
<!DOCTYPE html>
<html>
<head>
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
<script>
$(document).ready(function(){
$("#hide").click(function(){
$("p").hide();
});
$("#show").click(function(){
$("p").show();
});
});
</script>
</head>
<body>
<p>Si haces clic en el botón ocultar, voy a desaparecer</p>
<button id="hide">Ocultar</button>
<button id="show">Mostrar</button>
</body>
</html>
```

jQuery Efectos - Fading

Fading jQuery

Con jQuery se pueden declarar métodos de fundido con animaciones que pueden hacer desaparecer/aparecer un elemento.

jQuery tiene los siguientes métodos de fundido:

- fadeIn ()
- FadeOut ()
- fadeToggle ()
- fadeTo ()

jQuery fadeIn () Método

El jQuery fadeIn () se utiliza para hacer aparecer un elemento oculto.

Sintaxis:

$(selector).fadeIn(speed,callback);

El parámetro de velocidad opcional especifica la duración del efecto. Puede tomar los siguientes valores: lento (slow), rápido (fast), o milisegundos.

El parámetro callback opcional es una función a ejecutar después de la decoloración completa.

El siguiente ejemplo demuestra el método fadeIn () con diferentes parámetros:

Ejemplo

```
$("button").click(function(){
$("#div1").fadeIn(); $("#div2").fadeIn("slow"); $("#div3").fadeIn(3000); });
```

jQuery FadeOut () Método

El jQuery fadeIn () se utiliza para hacer desaparecer un elemento oculto.

Sintaxis:

$(selector).fadeOut(speed,callback);

Con el mismo funcionamiento que el fadeIn, se le puede enviar opcionalmente un parámetro de velocidad y/o la duración del efecto. Puede tomar los siguientes valores: "lento", "rápido", o milisegundos.

El parámetro callback opcional es una función a ejecutar después de la decoloración completa.

El siguiente ejemplo demuestra el método FadeOut () con diferentes parámetros:

Ejemplo

```
$("button").click(function(){ $("#div1").fadeOut(); $("#div2").fadeOut("slow"); $("#div3").fadeOut(3000); });
```

jQuery fadeToggle () Método

El método jQuery fadeToggle () alterna entre el () métodos fadeIn () y fadeOut().

Si los elementos se ocultaron, fadeToggle () hará un fadeIn().

Si los elementos se mostraron, fadeToggle () realizará un fadeOut();.

Sintaxis:

$(selector).fadeToggle(speed,callback);

El parámetro de velocidad opcional especifica la duración del efecto. Puede tomar los siguientes valores: lento (slow), rápido (fast), o milisegundos.

El parámetro callback opcional es una función a ejecutar después de la decoloración completa.

El siguiente ejemplo demuestra el método fadeToggle () con diferentes parámetros:

Ejemplo

```
$("button").click(function(){
$("#div1").fadeToggle(); $("#div2").fadeToggle("slow"); $("#div3").fadeToggle(3000); });
```

jQuery fadeTo () Método

El método jQuery fadeTo () permite la decoloración de una opacidad dada (valor entre 0 y 1).

Sintaxis:

$(selector).fadeTo(speed,opacity,callback);

El parámetro de velocidad requerida especifica la duración del efecto. Puede tomar los siguientes valores: "lento", "rápido", o milisegundos.

El parámetro de opacidad se requiere en el método fadeTo () especifica la decoloración de una opacidad dada (valor entre 0 y 1).

El parámetro callback opcional es una función a ejecutar después de la función completa.

El siguiente ejemplo demuestra el método fadeTo () con diferentes parámetros:

Ejemplo

```
$("button").click(function(){
$("#div1").fadeTo("slow", 0.15);
$("#div2").fadeTo("slow", 0.4);
$("#div3").fadeTo("slow", 0.7);
 });
```

Ejemplo:

```
<!DOCTYPE html>
<html>
<head>
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
<script>
$(document).ready(function(){
$("button").click(function(){
$("#div1").fadeIn();
$("#div2").fadeIn("slow");
$("#div3").fadeIn(3000);
});
});
</script>
</head>
<body>
<p>Demonstrate fadeIn() with different parameters.</p>
<button>Click to fade in boxes</button><br><br>
<div id="div1" style="width:80px;height:80px;display:none;background-color:red;"></div><br>
<div id="div2" style="width:80px;height:80px;display:none;background-color:green;"></div><br>
<div id="div3" style="width:80px;height:80px;display:none;background-color:blue;"></div>
</body>
</html>
```

jQuery - Efectos Sliding

Con jQuery se puede crear un efecto deslizante en elementos.
jQuery tiene los siguientes métodos de sliding:

- slideDown ()
- slideUp ()
- slideToggle ()

jQuery slideDown () Método

El jQuery slideDown () se utiliza para deslizar hacia abajo por un elemento.

Sintaxis:

$(selector).slideDown(speed,callback);

El parámetro de velocidad opcional especifica la duración del efecto. Puede tomar los siguientes valores: lento (slow), rápido (fast), o milisegundos.

El parámetro callback opcional es una función a ejecutar después de que se complete el deslizamiento.

El siguiente ejemplo demuestra el método slideDown ():

Ejemplo

```
$("#flip").click(function(){ $("#panel").slideDown(); })
```

jQuery slideUp () Método

El jQuery slideUp () se utiliza para deslizar hacia arriba un elemento.

Sintaxis:

$ (selector) .slideUp (velocidad, devolución de llamada);

El parámetro de velocidad opcional especifica la duración del efecto. Puede tomar los siguientes valores: lento (slow), rápido (fast), o milisegundos.

El parámetro callback opcional es una función a ejecutar después de que se complete el deslizamiento.

El siguiente ejemplo demuestra el método slideUp (): Ejemplo

```
$("#flip").click(function(){
$("#panel").slideUp(); });
```

jQuery slideToggle ()

El método jQuery slideToggle () alterna entre el () métodos slideDown () y slideUp.

Si los elementos se han deslizado hacia abajo, slideToggle () se deslizará hacia arriba.

Si los elementos se han deslizado hacia arriba, slideToggle () se deslizará hacia abajo.

$(selector).slideToggle(speed,callback);

El parámetro de velocidad opcional especifica la duración del efecto. Puede tomar los siguientes valores: lento (slow), rápido (fast), o milisegundos.

El parámetro callback opcional es una función a ejecutar después de que se complete el deslizamiento.

El siguiente ejemplo demuestra el método slideToggle (): Ejemplo

```
$("#flip").click(function(){
$("#panel").slideToggle(); });
```

Ejemplo:

```
<!DOCTYPE html>
 <html>
 <head>
<script
src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></scri
pt>
<script>
 $(document).ready(function(){
$("#flip").click(function(){
$("#panel").slideDown("slow"); }); });
</script>
<style>
#panel, #flip {
padding: 5px; text-align: center; background-color: #e5eecc; border: solid 1px
#c3c3c3; }
#panel {
padding: 50px; display: none; }
</style>
</head>
<body>
<div id="flip">Click to slide down panel</div>
<div id="panel">Hello world!</div>
</body>
</html>
```

jQuery - Obtener contenido y atributos

jQuery contiene métodos poderosos para cambiar y manipular elementos y atributos HTML.

jQuery DOM Manipulación

Una parte muy importante de jQuery es la posibilidad de manipular el DOM.

jQuery viene con un montón de métodos DOM relacionados que hacen que sea fácil de acceder y manipular los elementos y atributos.

DOM = Documento Modelo de objetos

El DOM define un estándar para acceder a documentos HTML y XML:

"El Modelo W3C objetos de documento (DOM) es una interfaz de plataforma y lenguaje neutro que permite a los programas y scripts acceder y actualizar el contenido, estructura dinámica, y estilo de un documento ".

Obtener contenido - texto (), html (), y val ()

Tres, métodos sencillos, pero útiles jQuery para la manipulación DOM son:

• text () - Establece o devuelve el contenido de texto de los elementosseleccionados

• html ()- Establece o devuelve el contenido de los elementos seleccionados (incluyendo el formato HTML)

• val () - Establece o devuelve el valor de los campos de formulario

El siguiente ejemplo muestra cómo obtener el contenido con el texto de jQuery () y html () métodos: ejemplo

```
$("#btn1").click(function(){
$("#test1").text("Hello world!"); }); $("#btn2").click(function(){
$("#test2").html("<b>Hello world!</b>"); }); $("#btn3").click(function(){
    $("#test3").val("Dolly Duck"); });
```

El siguiente ejemplo muestra cómo obtener el valor de un campo de entrada con el método jQuery val (). Ejemplo

```
$("#btn1").click(function(){
alert("Value: " + $("#test").val()); });
```

Obtener Atributos - attr ()

El método attr jQuery () se utiliza para obtener los valores de atributos.

El siguiente ejemplo muestra cómo obtener el valor del atributo href en un selector de ID

Ejemplo

```
$("button").click(function(){
alert($("#w3s").attr("href")); });
```

Ejemplo

```
<!DOCTYPE html>
<html>
<head>
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js" ></script> <script>
$(document).ready(function(){
$("button").click(function(){
alert("Value: " + $("#test").val()); }); });
</script>
</head>
<body>
<p>Name: <input type="text" id="test" value="Mickey Mouse"></p>
<button>Show Value</button>
</body>
</html>
```

jQuery - Establecer contenido y atributos

Establecer Contendio - text(), html(), and val()

vamos a utilizar los mismos tres métodos de la página anterior para ajustar el contenido:

- text () - Establece o devuelve el contenido de texto de los elementos seleccionados
- html () - Establece o devuelve el contenido de los elementos seleccionados (incluyendo el formato HTML)
- val () - Establece o devuelve el valor de los campos de formulario

El siguiente ejemplo muestra cómo establecer el contenido con el texto de jQuery (), html (), y Val) métodos

ejemplo

```
$("#btn1").click(function(){
$("#test1").text("Hello world!"); });
$("#btn2").click(function(){
$("#test2").html("<b>Hello world!</b>");});
$("#btn3").click(function(){
$("#test3").val("Dolly Duck"); });
```

Una llamada de retorno (callback) para el text (), html (), y val ()

Todos los tres métodos anteriores jQuery: text(), html (), y val (), también vienen con una función de devolución de llamada. La función de devolución de llamada (callback) tiene dos parámetros: el índice del elemento actual de la lista de elementos seleccionados y el valor original (antigua). A continuación, retornar la cadena que desea utilizar como el nuevo valor de la función.

El siguiente ejemplo muestra el text () y html () con una función de devolución de llamada.

Ejemplo

```
$("#btn1").click(function(){
$("#test1").text(function(i, origText){
return "Old text: " + origText + " New text: Hello world! (index: " + i + ")"; });
 });
$("#btn2").click(function(){
$("#test2").html(function(i, origText){
return "Old html: " + origText + " New html: Hello <b>world!</b> (index: " + i + ")"; }); });
```

Obtener/Establecer Attributos - attr()

El método attr jQuery () se utiliza para obtener los valores de atributos.

El siguiente ejemplo muestra cómo obtener el valor del atributo href en un enlace.

```
$("button").click(function(){
alert($("#w3s").attr("href")); });
```

El método jQuery attr () también se utiliza para ajustar / cambiar los valores de atributos.

El siguiente ejemplo muestra cómo cambiar (set) el valor del atributo href en un enlace.

```
$("button").click(function(){
$("#w3s").attr("href", "http://www.w3schools.com/jquery"); });
```

El método attr () también le permite configurar múltiples atributos al mismo tiempo.

El siguiente ejemplo muestra cómo establecer tanto el href y el título atributos al mismo tiempo:

Ejemplo

```
$("button").click(function(){
$("#w3s").attr({
"href" : "http://www.w3schools.com/jquery", "title" : "W3Schools jQuery Tutorial" }); });
```

Una llamada de retorno (callback) para attr ()

El método attr jQuery (), también vienen con una función de devolución de llamada (callback). La función de devolución de llamada tiene dos parámetros: el índice del elemento actual de la lista de elementos seleccionados y el original (antiguo) el valor de atributo. A continuación, retornar la cadena que desea utilizar como el nuevo valor del atributo de la función.

El siguiente ejemplo demuestra attr () con una función de devolución de llamada.

Ejemplo

```
$("button").click(function(){
$("#w3s").attr("href", function(i, origValue){
return origValue + "/jquery"; }); });
```

jQuery - Añadir elementos

Con jQuery, es fácil de añadir nuevos elementos / contenido.

Añadir nuevo contenido HTML

Vamos a ver cuatro métodos de jQuery que se utilizan para añadir nuevos contenidos:

- append() - Añade contenido al final del elemento seleccionado.
- prepend() - Añade contenido al principio del elemento seleccionado.
- after() - Inserta el contenido después de los elementos seleccionados
- before () - Inserta el contenido antes de los elementos seleccionados

jQuery append ()método

El método jQuery append () inserta contenido al final de los elementos HTML seleccionados.

Ejemplo

```
$("p").append("Some appended text.");
```

jQuery prepend () Método

métodoEl jQuery prepend () inserta contenido al principio de los elementos HTML seleccionados. Ejemplo

```
$("p").prepend("Some prepended text.");
```

Añadir varios nuevos elementos con append () y PREPEND ()

En los dos ejemplos anteriores, sólo hemos insertado un texto / HTML al principio / final de los elementos HTML seleccionados.

Sin embargo, tanto en el modo de append () y prepend () pueden tomar un número infinito de nuevos elementos como parámetros. Los nuevos elementos se pueden generar conl texto / HTML (como lo hemos hecho en los ejemplos anteriores), con jQuery, o con código JavaScript y elementos DOM.

En el siguiente ejemplo, creamos varios elementos nuevos. Los elementos se crean con el texto / HTML, jQuery y JavaScript / DOM. A continuación, añadimos los nuevos elementos al texto con el método append () (esto funcionaría igual con el preprend () también):

Ejemplo

```
function appendText() {
var txt1 = "<p>Text.</p>"; // Create element with HTML
var txt2 = $("<p></p>").text("Text."); // Create with jQuery
var txt3 = document.createElement("p"); // Create with DOM
txt3.innerHTML = "Text."; $("p").append(txt1, txt2, txt3); // Append the new elements }
```

jQuery after() (antes) and before() (después)

El método jQuery after () inserta contenido después de los elementos HTML seleccionados.

El método jQuery before () inserta el contenido antes de los elementos HTML seleccionados. Ejemplo

```
$("img").after("Some text after");
$("img").before("Some text before");
```

Añadir varios nuevos elementos con after () y before ()

Además, tanto el before () y after () pueden tener un número infinito de nuevos elementos como parámetros. Los nuevos elementos se pueden generar con el texto / HTML (como lo hemos hecho en el ejemplo anterior), con jQuery, o con código JavaScript y elementos DOM.

En el siguiente ejemplo, creamos varios elementos nuevos. Los elementos se crean con el texto / HTML, jQuery y JavaScript / DOM. A continuación insertamos los nuevos elementos al texto con el método después de () (esto habría trabajado para antes () también):

Ejemplo

```
function afterText() {
var txt1 = "<b>I </b>"; // Create element with HTML
var txt2 = $("<i></i>").text("love "); // Create with jQuery
var txt3 = document.createElement("b"); // Create with DOM
txt3.innerHTML = "jQuery!"; $("img").after(txt1, txt2, txt3); // Insert new elements after img
 }
```

Ejemplo Completo

```
<!DOCTYPE html>
<html>
<head>
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
<script>
$(document).ready(function(){
$("#btn1").click(function(){
$("p").prepend("<b>Prepended text</b>. "); });
$("#btn2").click(function(){
$("ol").prepend("<li>Prepended item</li>"); }); });
</script>
</head>
<body>
<p>This is a paragraph.</p> <p>This is another paragraph.</p>
<ol>
<li>List item 1</li>
<li>List item 2</li>
<li>List item 3</li>
</ol>
<button id="btn1">Prepend text</button>
<button id="btn2">Prepend list item</button>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"
></script> <script>
$(document).ready(function(){
$("#btn1").click(function(){
$("img").before("<b>Before</b>"); });
$("#btn2").click(function(){
$("img").after("<i>After</i>"); }); });
</script>
</head>
<body>
<img src="/images/w3jquery.gif" alt="jQuery" width="100" height="140"><br><br>
<button id="btn1">Insert before</button> <button id="btn2">Insert
after</button>
</body>
</html>
```

jQuery - eliminar elementos

Con jQuery, es fácil de quitar elementos HTML existentes.

eliminar elementos / contenido

Para eliminar los elementos y contenidos, existen principalmente dos métodos de jQuery:

- remove() - Elimina el elemento seleccionado (y sus elementos secundarios)
- empty() - Elimina los elementos secundarios del elemento seleccionado

jQuery método remove ()

El método jQuery remove () elimina el elemento (s) seleccionado y sus elementos secundarios.

Ejemplo

```
$ ( "#div1") remove ().;
```

jQuery vacía () Método

El método empty() de jQuery elimina los elementos secundarios del elemento (s) seleccionado.

Ejemplo

```
$("#div1").empty();
```

Filtrar los elementos que van a ser eliminados

El método jQuery remove () también acepta un parámetro, que le permite filtrar los elementos que se deben eliminar.

El parámetro puede ser cualquiera de las sintaxis del selector de jQuery.

El siguiente ejemplo elimina todos los elementos <p> con class = "italic":

Ejemplo

```
$("p").remove(".italic");
```

Ejemplo completo

```
<!DOCTYPE html>
<html>
<head>
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
<script>
$(document).ready(function(){
$("button").click(function(){
$("#div1").remove(); }); });
</script>
</head>
<body>
<div id="div1" style="height:100px;width:300px;border:1px solid black;background-color:yellow;"> This is some text in the div. <p>This is a paragraph in the div.</p>
<p>This is another paragraph in the div.</p>
</div>
<br>
<button>Remove div element</button>
</body>
</html>
```

jQuery - Obtener y establecer clases CSS

jQuery manipulación de CSS

jQuery tiene varios métodos para la manipulación de CSS. Vamos a ver los métodos siguientes:

- addClass () - Añade una o más clases de los elementos seleccionados
- removeClass () - Elimina una o más clases de los elementos seleccionados
- toggleClass () - Cambia entre la adición / eliminación de clases de la seleccionadaelementos
- css () - Establece o devuelve el estilo de atributos

jQuery addClass()

El ejemplo siguiente muestra cómo agregar atributos de clase a los diferentes elementos. Por supuesto, puede seleccionar varios elementos, al agregar

Ejemplo

```
$("button").click(function(){
$("h1, h2, p").addClass("blue"); $("div").addClass("important");
});
```

También puede especificar varias clases dentro de la addClass () Método:

Ejemplo

```
$("button").click(function(){
$("#div1").addClass("important blue"); });
```

jQuery removeClass () Método

El siguiente ejemplo muestra cómo eliminar un atributo de clase específica de diferentes elementos:

Ejemplo

```
$("button").click(function(){
$("h1, h2, p").removeClass("blue"); });
```

jQuery toggleClass () Método

El siguiente ejemplo mostrará cómo utilizar el método jQuery toggleClass (). Este método se conmuta entre añadir / eliminar las clases de los elementos seleccionados:

Ejemplo

```
$("button").click(function(){
    $("h1, h2, p").toggleClass("blue"); });
```

Ejemplo

```
<! DOCTYPE html>
<html>
<head>
<script src = "http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"> </ script>
<script > $ (document) ready (function ()
{$( "botón") haga clic (function () "azul");..});});
{$( "h1, h2, p") toggleClass (
</ script>
<style>
.Blue
{color:blue; }
 </ Style>
</ head>
<body>
<h1> 1 </ h1> <h2> Rúbrica 2 </ h2>
<p> Este es un párrafo. </ P> <p> Este es otro párrafo. </ p>
<botón> clase Toggle botón </>
</ body>
</html>
```

jQuery - css () Método

El método css () establece o devuelve una o más propiedades de estilo para los elementos seleccionados.

Devolver una propiedad CSS

Para devolver el valor de una propiedad CSS especificado, utilice la siguiente sintaxis:

css("propertyname");

El siguiente ejemplo devolverá el valor de color de fondo del primer elemento encontrado que coincida, en este caso, el primer párrafo declarado.

Ejemplo

```
$ ( "p") CSS ( "background-color.");
```

Establecer una propiedad CSS

Para establecer una propiedad CSS especificado, utilice la siguiente sintaxis:

css ("propertyname", "valor");

El siguiente ejemplo se establecerá el valor de color de fondo para todos los elementos coincidentes:

Ejemplo

```
$("p").css("background-color", "yellow");
```

Establecer propiedades CSS Múltiples

Para establecer múltiples propiedades CSS, utilice la siguiente sintaxis:

css ({ "propertyname": "valor", "propertyname": "valor", ...});

El siguiente ejemplo creará un fondo de color y un tamaño de fuente para todos los elementos coincidentes:

Ejemplo

```
$("p").css({"background-color": "yellow", "font-size": "200%"});
```

Ejemplo

```
<!DOCTYPE html>
 <html>
 <head>
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js" ></script> <script>
 $(document).ready(function(){
$("button").click(function(){
$("p").css({"background-color": "yellow", "font-size": "200%"});
}); });
 </script>
</head>
 <body>
<h2>This is a heading</h2>
<p style="background-color:#ff0000">This is a paragraph.</p> <p style="background-color:#00ff00">This is a paragraph.</p> <p style="background-color:#0000ff">This is a paragraph.</p>
<p>This is a paragraph.</p>
<button>Set multiple styles for p</button>
</body>
</html>
```