División de Alta Tecnología

3

JavaScript, jQuery, JSON y Ajax

Visual Studio 2017 Web Developer

Objetivos

Al finalizar el capítulo, el alumno:

- Comprender el modelo DOM de una página HTML.
- Construir páginas básicas con HTML.
- Desarrollar funciones JavaScript para manipular los elementos HTML.
- Utilizar el patrón NameSpaces.
- Aplicar JQuery, Ajax y JSON para enviar información al servidor.

Agenda

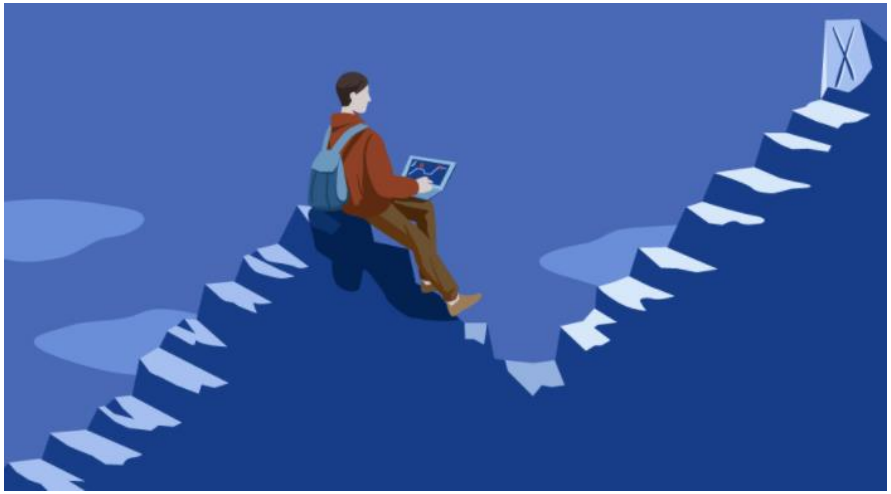
- JavaScript
  - Tipos de variables
  - Operadores
  - Instrucciones de control - condicionales
  - Instrucciones de control - bucles
  - Creación de objetos
  - Arrays
  - Document Object Model (DOM)
  - Manejo de eventos
  - Local y Session Storage

Agenda

- jQuery
  - Selectores jQuery
  - Eventos
  - Ajax con jQuery
- JSON creación y uso
- AJAX con jQuery

JavaScript

- JavaScript es un lenguaje de programación interpretado, dialecto del estándar ECMAScript.
- Permite interactuar con los documentos HTML, manipular el DOM y manejar eventos.



JavaScript

- Tipos de variables

TIPO	DESCRIPCIÓN	EJEMPLOS
Numéricas	Se utilizan para almacenar valores numéricos enteros o decimales.	var iva = 20; var total = 214.35;
Cadenas de Texto	Se utilizan para almacenar caracteres, palabras o frases de texto.	var mensaje = "Bienvenido a DAT"; var nombre = 'Juan Chavez';
Arrays	Se utilizan cuando se desea almacenar una colección de variables, que pueden ser todas del mismo tipo o cada una de un tipo diferente.	var dias = ["Lunes", "Martes", "Miércoles", "Jueves", "Viernes", "Sábado", "Domingo"];
Booleanos	Se utilizan para almacenar un tipo especial de valor que solamente puede tomar dos valores: true (verdadero) o false (falso).	var procesoTerminado=false; var igvIncluido=true;
Fecha	Para trabajar con fechas en javascript se utiliza la clase Date	miFecha = new Date() //Permite obtener la fecha y hora actual

JavaScript

- Operadores

TIPO	DESCRIPCIÓN
Igual (==)	Devuelve true si valor de las variables son iguales.
Distinto (!=)	Devuelve true si el valor de las variables no son iguales.
Igual estricto (===)	Devuelve true si las variables son iguales en valor y tipo.
Distinto estricto (!==)	Devuelve true si las variables no son iguales en valor y tipo.
Mayor que (>)	Devuelve true si la variable izquierda es mayor que la derecha.

JavaScript

- Instrucciones de control - Condicionales

TIPO	DESCRIPCIÓN	EJEMPLO

If …
else …
	Utilice la sentencia if para ejecutar una sentencia si la condición lógica es verdadera. Utilice la cláusula opcional else para ejecutar una sentencia si la condición es falsa.	If (edad >= 18) {
		alert ("Mayor de edad");
		}
		else {
		alert ("Menor de edad");
		}

switch	Permite evaluar una expresión e intenta compararla con el valor de la etiqueta de una expresión por casos. Si la coincidencia es encontrada, ejecuta la sentencia asociada.	switch (var1) {
		case 1:
		var2=10;break;
		case 2:
		var2=20;break;
		default:
		var=0;break; }

JavaScript
Creación de objetos

- Los objetos en JavaScript, al igual que en muchos otros lenguajes de programación, pueden ser comparados con objetos de la vida real.
- El concepto de objetos en JavaScript se puede entender como en la vida real, objetos tangibles.
- Una propiedad de un objeto puede ser explicada como una variable que se adjunta al objeto.

```
1  var miAuto = new Object();
2  miAuto.marca = "Ford";
3  miAuto.modelo = "Mustang";
4  miAuto.annio = 1969;
```

```
1  miAuto["marca"] = "Ford";
2  miAuto["modelo"] = "Mustang";
3  miAuto["annio"] = 1969;
```

JavaScript
Creación de objetos

- El uso de inicializadores de objeto

```
var miHonda = {color: "rojo", ruedas: 4, motor: {cilindros: 4, tamanio: 2.2}};
```

- Un método es una función asociada a un objeto, o, simplemente, un método es una propiedad de un objeto que representa una función.

```
nombreDelObjeto.nombreDelMetodo = nombre_de_la_funcion;

var miObjeto = {
  miMetodo: function(parametros) {
    // ...hacer algo
  }
};
```

```
object.nombreDelMetodo(parametros);
```

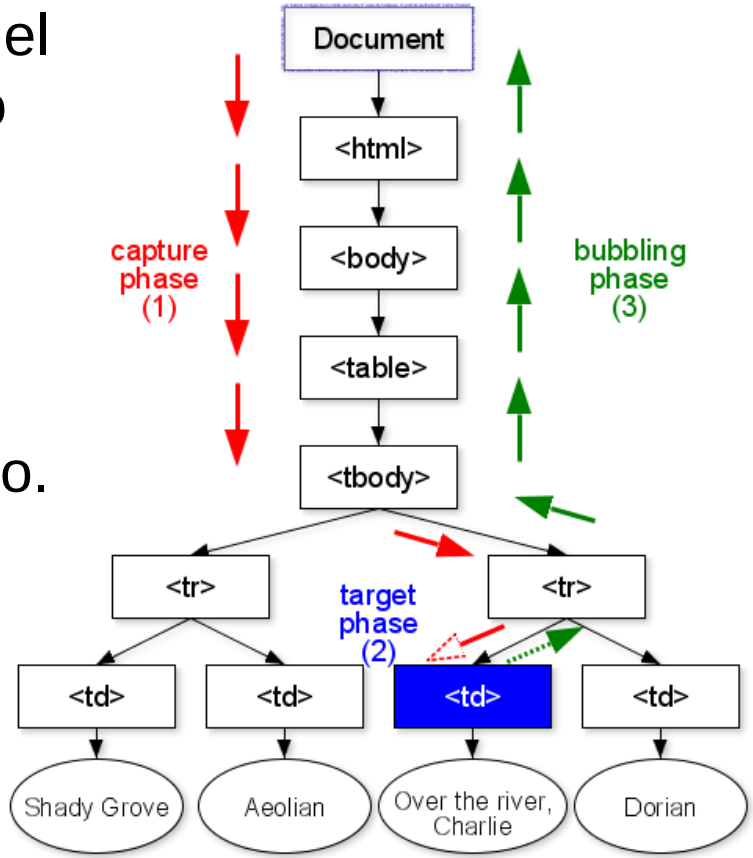
JavaScript
Creación de objetos

- Eliminando propiedades

```
//Crea un nuevo objeto, miobjeto, con dos propiedades, a y b.
var miobjeto = new Objecto;
miobjeto.a = 5;
miobjeto.b = 12;

//Elimina la propiedad, dejando miobjeto con sólo la propiedad b.
delete myobj.a;
console.log ("a" in myobj) // yields "false"
```

JavaScript
Creación de objetos

- Document Object Model (DOM)
  - Permite acceder y modificar el contenido, estructura y estilo de los documentos HTML.
  - Define la manera en que los objetos y elementos se relacionan entre sí, en el navegador y en el documento.



JavaScript

- Manejo de eventos

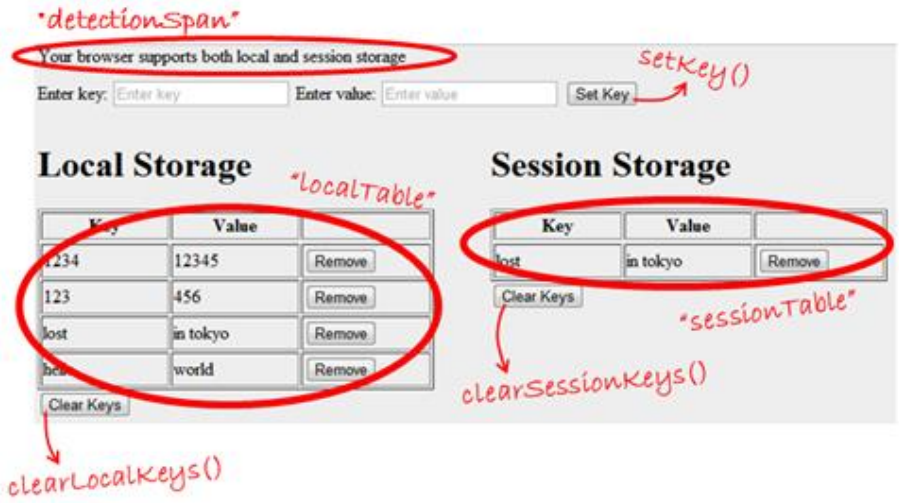
EVENTO	DESCRIPCIÓN	DEFINIDO PARA
onblur	Deseleccionar el elemento.	<button>, <input>, <label>, <select>, <textarea>, <body>
onchange	Deseleccionar un elemento que se ha modificado.	<input>, <select>, <textarea>
onclick	Al hacer clic.	Todos los elementos
ondblclick	Hacer clic dos veces seguidas.	Todos 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>

JavaScript
Local Session y Session Storage

- HTTP no tiene estado, o sea, es stateless.
- Los cookies permiten mantener el estado entre peticiones (guardar información en el browser).
- Las cookies siempre viajan entre el cliente y el servidor.
- En HTML5 se tienen dos mecanismos para guardar información en el cliente sin tener que ser enviado en cada momento al servidor, estos son:
  - Local Storage
  - Session Storage

JavaScript
Local Session y Session Storage

- LocalStorage: permite guardar información en el cliente y no tiene un tiempo de expiración.
- SessionStorage: permite guardar información en el cliente y la información se borra cuando cierra el browser.



JavaScript
Local Session y Session Storage

- Métodos comunes
  - getItem(key) permite obtener un valor del storage.
  - setItem(key, value) permite establecer valor en storage.
  - removeItem(key) permite remover elemento de storage.

JavaScript
Local Session y Session Storage

- LocalStorage

```
if (window.localStorage) {

  localStorage.setItem("nombre", "pepe");

  var nombre = localStorage.getItem("nombre");

  localStorage.removeItem("nombre");
}
else {
  throw new Error('Tu Browser no soporta LocalStorage!');
}
```

JavaScript
Local Session y Session Storage

- SessionStorage

```
// Almacena la información en sessionStorage
sessionStorage.setItem('key', 'value');


// Obtiene la información almacenada desde sessionStorage
var data = sessionStorage.getItem('key');
```

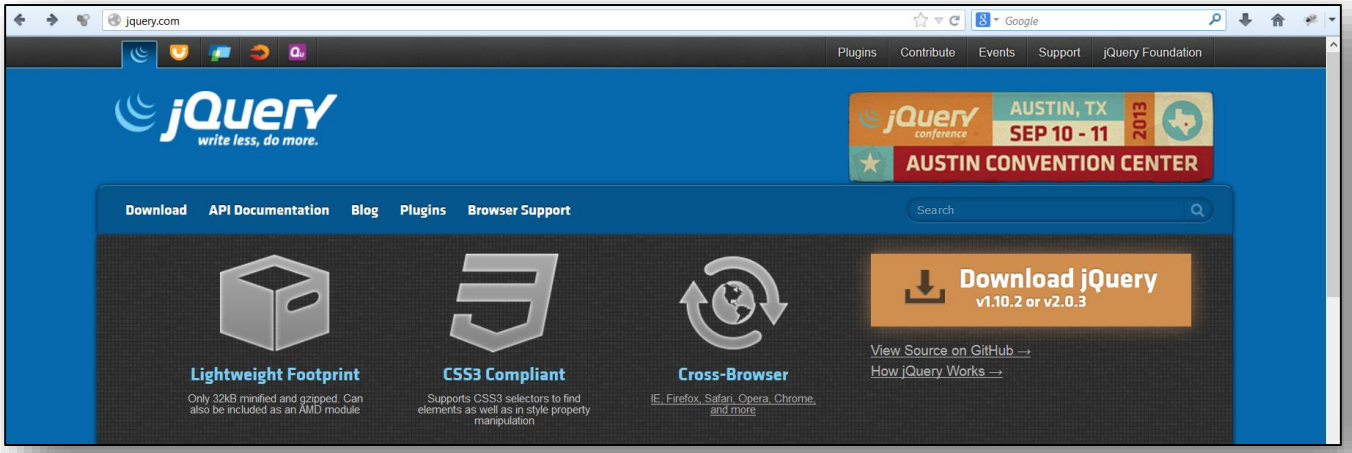
JavaScript
Local Session y Session Storage

- Guardar un objeto
  - Primero se tiene que serializar (convertirlo en cadena) el objeto con JSON.stringify
  - Para deserializarlo (convertirlo nuevamente a objeto) se debe usar JSON.parse

```
var personaAGuardar = JSON.stringify(persona);

localStorage.setItem("persona", personaAGuardar);

var personaGuardada = localStorage.getItem("persona");

console.log(typeof persona); //object

console.log(typeof personaGuardada); //string

var personaGuardada = JSON.parse(personaGuardada);

console.log(personaGuardada.locura); //true
```

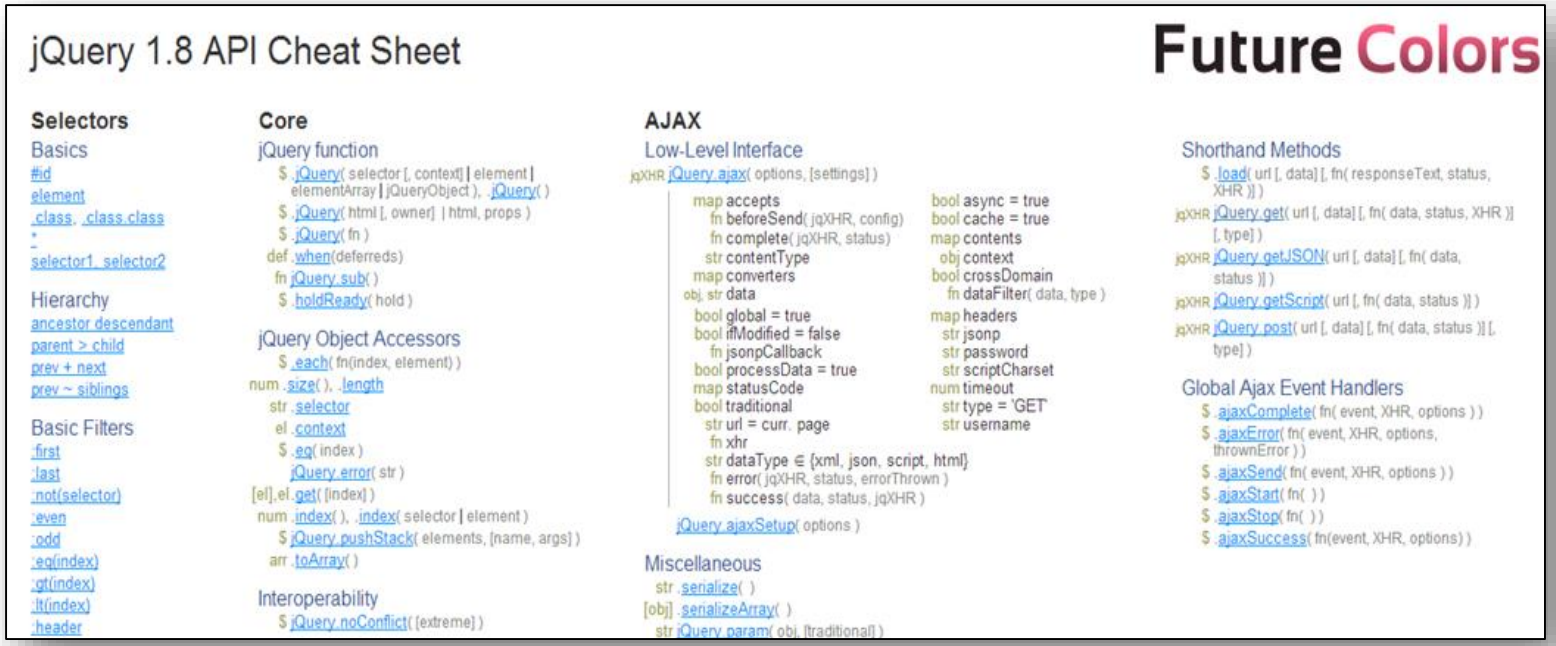
jQuery

- Es un JavaScript library que permite simplificar la manera de interactuar con los documentos HTML, manipular los DOM, manejar los eventos, desarrollar animaciones y agregar las interacciones con AJAX en aplicaciones web.



http://jquery.com/

jQuery



jQuery 1.8 API Cheat Sheet

Future Colors

Selectors

Basics
#id
element
.class, .class.class
*
selector1, selector2

Hierarchy
ancestor descendant
parent > child
prev + next
prev ~ siblings

Basic Filters
:first
:last
:not(selector)
:even
:odd
:eq(index)
:gt(index)
:lt(index)
:header

Core

jQuery function
$ .jQuery(selector [, context] | element | elementArray | jQueryObject), .jQuery( )
$ .jQuery(html [, owner] | html, props)
$ .jQuery(fn)
def .when(deferreds)
fn jQuery.sub( )
$ .holdReady(hold)

jQuery Object Accessors
$ .each(fn(index, element))
num .size( ), .length
str .selector
el .context
$ .eq(index)
jQuery.error(str)
[el],el.get([index])
num .index( ), .index(selector | element)
$ jQuery.pushStack(elements, [name, args])
arr .toArray( )

Interoperability
$ jQuery.noConflict([extreme])

AJAX

Low-Level Interface
jqXHR jQuery.ajax(options, [settings])
map accepts
fn beforeSend(jqXHR, config)
fn complete(jqXHR, status)
str contentType
map converters
obj, str data
bool global = true
bool ifModified = false
fn jsonpCallback
bool processData = true
map statusCode
bool traditional
str url = curr. page
fn xhr
str dataType ∈ {xml, json, script, html}
fn error(jqXHR, status, errorThrown)
fn success(data, status, jqXHR)
bool async = true
bool cache = true
map contents
obj context
bool crossDomain
fn dataFilter(data, type)
map headers
str jsonp
str password
str scriptCharset
num timeout
str type = 'GET'
str username
jQuery.ajaxSetup(options)

Miscellaneous
str .serialize( )
[obj] .serializeArray( )
str jQuery.param(obj, [traditional])

Shorthand Methods
$ .load(url [, data] [, fn(responseText, status, XHR)])
jqXHR jQuery.get(url [, data] [, fn(data, status, XHR)] [, type])
jqXHR jQuery.getJSON(url [, data] [, fn(data, status)])
jqXHR jQuery.getScript(url [, fn(data, status)])
jqXHR jQuery.post(url [, data] [, fn(data, status)] [, type])

Global Ajax Event Handlers
$ .ajaxComplete(fn(event, XHR, options))
$ .ajaxError(fn(event, XHR, options, thrownError))
$ .ajaxSend(fn(event, XHR, options))
$ .ajaxStart(fn( ))
$ .ajaxStop(fn( ))
$ .ajaxSuccess(fn(event, XHR, options))

http://jqapi.ru/

jQuery
Selectores

- Permiten acceder de una manera rápida y sencilla a un elemento o grupo de elementos del DOM y luego poder aplicar sobre los mismos cualquier tipo de instrucción, evento, animación, etc.

```
//jQuery("#id")
$("#myDiv");

//jQuery("ElementType")
$("div");

//jQuery(".class")
$(".myClass");

//jQuery("selector1, selector2, selectorN")
$("span,#myDiv,p.myclass");

//jQuery(":ControlType")
$(":button");
```

jQuery
Eventos

- Lista de eventos

EVENTO	DESCRIPCIÓN
click()	Ocurre cuando se produce un clic en un elemento de la página
dblclick()	Ocurre cuando se produce un doble click sobre un elemento de la página
mousedown()	Ocurre cuando se hace clic, en el momento que se presiona el botón.
mouseup()	Ocurre cuando se hace clic y luego se suelta un botón del mouse.
mouseenter()	Ocurre al situar el mouse encima de un elemento de la página.
mouseleave()	Ocurre cuando el mouse sale de encima de un elemento de la página.
mousemove()	Ocurre al mover el mouse sobre un elemento de la página.
mouseout()	Ocurre cuando el usuario sale con el mouse de la superficie de un elemento de la página.
mouseover()	Ocurre cuando el mouse está sobre un elemento de la página.
toogle()	Sirve para indicar dos o más funciones para ejecutar acciones al hacer clic.
hover()	Sirve para manejar dos eventos: cuando el mouse entra y sale de encima de un elemento.

jQuery
Eventos

- Lista de eventos

EVENTO	DESCRIPCIÓN
keydown()	Este evento se produce en el momento que se presiona una tecla. Se produce una única vez en el momento exacto de la presión, ocurre después de escribir el carácter en pantalla.
keypress()	Este evento se produce en el momento que se presiona una tecla. Es parecido al evento keydown, pero keypress se lanza antes de escribir el carácter en pantalla. Se ejecuta una vez como respuesta a una pulsación e inmediata liberación de la tecla, o varias veces si se pulsa una tecla y se mantiene pulsada.
keyup()	Ocurre en el momento de liberar una tecla, es decir, al dejar de presionar una tecla que se tenía pulsada.

JSON
Creación y uso

- Es un formato ligero de intercambio de datos.
- Entenderlo es simple para los humanos, mientras que para las máquinas es fácil interpretarlo y generarlo.
- Basado en un subconjunto del lenguaje de programación JavaScript, Standard ECMA-262 3rd Edition 1999.
- Significa JavaScript Object Notation (Notación de Objetos de JavaScript)

JSON
Creación y uso

- JSON está constituido por dos estructuras:

  - Una colección de pares de nombre/valor

```
"firstName":"John"
```

  - Una lista ordenada de valores

```
"employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter","lastName":"Jones"}
]
```

JSON: creación y uso

- Objeto complejo:

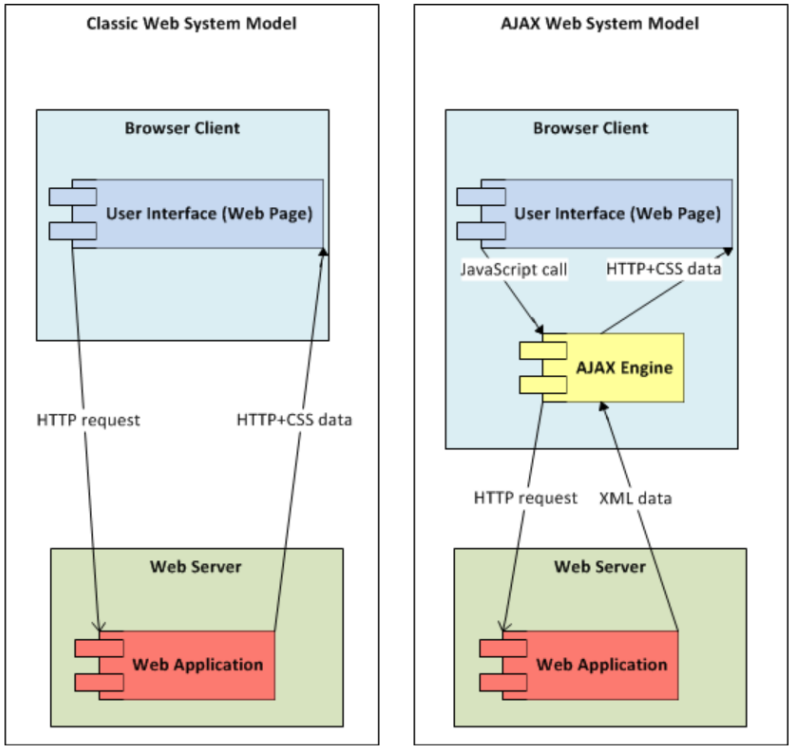
```
var persona = {
    'nombre': 'Cesar',
    'edad': 25,
    'estudios': ['primaria', 'secundaria','universidad']
};
```

AJAX con jquery

- Es una técnica de desarrollo web para crear aplicaciones interactivas combinando tecnologías ya existentes: HTML, CSS, XML, JavaScript y algún lenguaje de servidor.
- Consiste en realizar una comunicación asíncrona con el servidor en segundo plano, de esta forma, realiza cambios sobre la misma página sin necesidad de refrescarla.
- Permite aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

AJAX con jquery

- Para trabajar con Ajax es necesario manipular el objeto XMLHttpRequest del navegador o Ajax Engine.



AJAX con jquery

- jQuery facilita el trabajo y provee las siguientes funciones:

```
//Utilizando $.get():
$.get("/Demo/Mantenimiento", { nombre: "Cesar", edad: "25" },
  function (data) {
      alert("Información Cargada: " + data);
  });

//Utilizando $.post():
$.post("/Demo/Mantenimiento", { nombre: "Juan", edad: "18" },
  function (data) {
      alert("Información Cargada: " + data);
  });

//Utilizando $.getJSON():
$.getJSON('/Demo/ProfesoresListar', { busqueda: filtro }, function (dataLista) {

    alert(dataLista[1].nombre);
});
```

Ejercicio Nº 3.1: Implementar NameSpaces pattern

Implementar NameSpaces pattern de una aplicación

Al finalizar el laboratorio, el alumno logrará:

- Entender el uso del NameSpaces pattern.

Ejercicio Nº 3.2: Uso de controles personalizados de Video en HTML5

Personalizar los controles de reproducción en HTML5.

Al finalizar el laboratorio, el alumno logrará:

- Implementar controles personalizados de reproducción de video con JavaScript.

Ejercicio Nº 3.3: Implementar geolocalización con JavaScript y Google Map

Crear una página HTML que muestre la ubicación actual en el Google Map.

Al finalizar el laboratorio, el alumno logrará:

- Implementar geolocalización con JavaScript y Google Map.

Tarea Nº 3: JavaScript, jQuery, JSON y Ajax

- En el módulo administrador implementar la pantalla para gestionar las ventas, cómo cambiar el estado de la venta y verificar los productos vendidos.