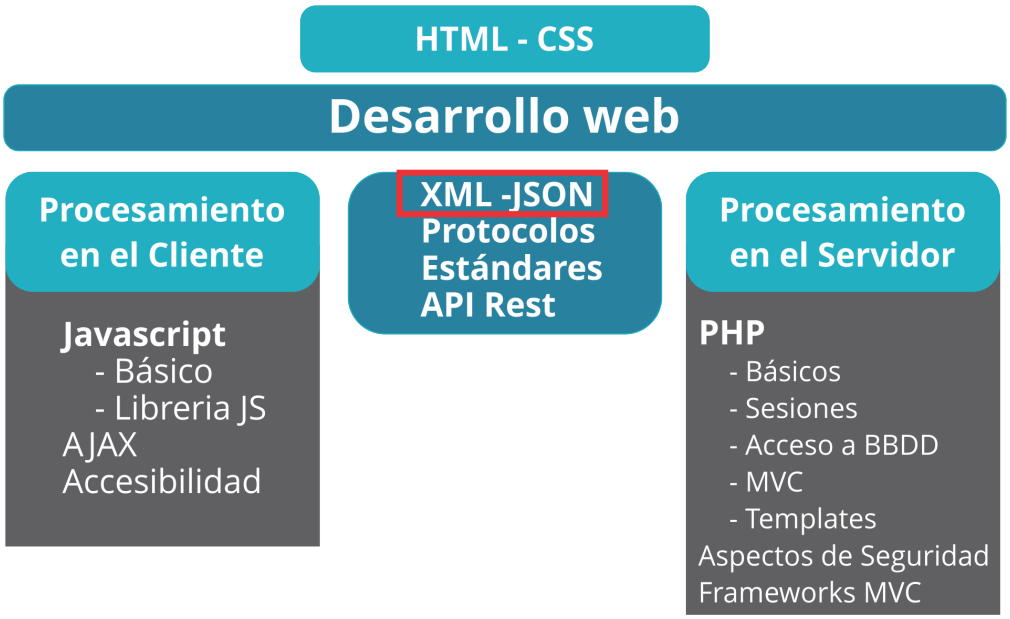
skip-help: true
title: Clase 5 Proyecto 2016
Author: Einar Lanfranco, Claudia Banchoff, Diego Vilches, Matías Pagano
description: Notaciones para descripción de datos
keywords: XML + JSON + YAML
css: proyecto.css

Proyecto de Software

Cursada 2016

Hoy seguimos con ...



Temario

- **Repaso Clase Anterior**
    - **Consultas a las BBDD**
        - SQL Injection
    - **MVC**
        - la vista con templates
- **Describiendo información**

    - XML
    - JSON
    - YAML
- **Procesamiento en el cliente**
    - Javascript

Repaso - Inyección SQL

- Una inyección SQL suele ocurrir cuando se arma en forma descuidada una consulta a la base de datos a partir de los datos ingresados por el usuario.
- Dentro de estos parámetros pueden venir el código malicioso.
- Ejemplos típicos:

```
select * from users where id='". "1' or '1=1" ."' and
                          pass='". "1' or '1=1" ."'";
```

- Lo que se se resuelve en:

```
select * from users where id='1' or '1=1'
                    and pass='1' or '1=1';
```

Repaso - Prepared Statement

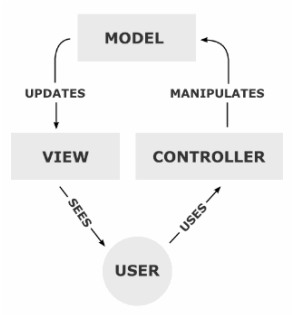
- Pueden definirse como un tipo de plantillas compiladas para SQL que las aplicaciones quieren ejecutar, pueden ser personalizadas usando parámetros de variables.
- Ejemplo:

```
<?php
   $db_host="127.0.0.1";
   $db_user="user";
   $db_pass="pass";
   $db_base="base";

   $cn = new PDO("mysql:dbname=$db_base;host=$db_host",$db_user,$db_pass);

   $query = $cn->prepare("SELECT * FROM usuarios where nombre=? and pass=?");
   $query->execute(array($_POST["email"],$_POST["pass"]));
?>
```

Repaso - MVC



- Reduce la complejidad, facilita la reutilización y acelera el proceso de comunicación entre capas.

Repaso - MVC

```php
<?php
//Conectamos a la Base
require_once("2-modelo.php");

//Recupero los usuarios
$usuarios=obtener_usuarios();

//Cargo la vista
require("2-vista.php");

?>
```

```php
<?php
function obtener_usuarios(){
    $db_host='127.0.0.1';
    $db_user='user';
    $db_pass='pass';
    $db_base='base';
    $link = mysqli_connect($db_host,$db_user,$db_pass,$db_base) o
    $resu=$link->query('select * from usuarios');

    while ($dato = mysqli_fetch_array($resu))  {
        $usuarios[]=$dato;
    }
    // Cierro la conexión
    mysqli_close($link);
    return $usuarios;
}
?>
```

```php
<html>
<head><title>Listados de Usuarios</title></head>
<body><h1>Listado de Usuarios Activos </h1>
<table>
<tr><th>Nombre</th><th>DNI</th></tr>

<?php foreach ($usuarios as $usuario){
    echo "<tr><td>".$usuario->nombre . "</td>";
    echo "<td>". $usuario->dni."</td></tr>";
};?>

</table>
</body>
</html>
```

---

class: destacado

Repaso - Templates

- El uso de templates o plantillas permite separar la aplicación de la presentación, pero ....

No asegura MVC. **Esa es NUESTRA responsabilidad**

Repaso - Twig

- Los templates **se utilizan para definir la vista**.
- Tienen un formato especial.
- No utiliza una extensión en particular (podría ser html, xml, twig, tpl, etc.).
- Son procesados por el sistema de plantillas.
- Contienen variables o expresiones que son reemplazadas cuando se procesa el template y tags que proveen la lógica de la presentación.
- **Ejemplo:**
    - template_twig_php
    - template_twig_tmpl

Describiendo información

XML, JSON, YAML ....

XML - eXtensible Markup Language

- Es un lenguaje de marcas.
- Es un metalenguaje.
- Surge de la necesidad de contar con un mecanismo para describir información estructurada.
- XML describe semántica.
- **Existe SGML – "Standarized General Markup Language", pero ...**
    - Es complejo de procesar.
- **Existe HTML, pero ...**
    - NO fue pensado para este fin.

XML - Sintaxis Básica

¿Nos suena conocido?

```
<?xml version="1.0" encoding="UTF-8"?>
<persona>
        <nombre>Pedro</nombre>
        <apellido>Perez</apellido>
        <fotoPerfil fotoTrabajo="carnet.jpeg"/>
</persona>
```

- Usamos etiquetas, aunque las definimos nosotros...

XML - Sintaxis Básica

Misma sintaxis de HTML, aunque con algunas consideraciones:

- Elementos: Vacíos y No vacíos.
- Existe sensibilidad a mayúsculas y minúsculas.
- Debe tener una **única** raíz.
- **Atributos asociados a la etiqueta de apertura.**
    - pares **nombreAtributo="valorAtributo"**
    - SIEMPRE entre **""**

Varios elementos

```
<?xml version="1.0" encoding="UTF-8"?>
<amigos>
        <persona>
                <nombre>Pedro</nombre>
                <apellido>Perez</apellido>
                <fotoPerfil fotoTrabajo="carnetPerez.jpeg"/>
        </persona>
        <persona>
                <nombre>Roberto</nombre>
                <apellido>Lopez</apellido>
                <fotoPerfil fotoTrabajo="carnetLopez.jpeg"/>
        </persona>
</amigos>
```

- Ver: XML_simple

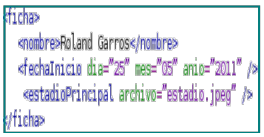
Ejemplo de XSL (hojas de estilo para documentos XML)

```
<?xml version="1.0" encoding="UTF-8"?>
<html xsl:version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<body style="font-family:Arial;font-size:12pt;background-color:#EEEEEE">
        <h1>Saludos!!!</h1>
        <p> Mis Amigos hasta ahora...</p>
        <ul>
          <xsl:for-each select="amigos/persona">
                <li><xsl:value-of select="nombre"/></li>
          </xsl:for-each>
        </ul>
</body>
</html>
```

- Ver: XSL_simple

¿Cómo validar un documento XML?

Representan lo mismo, pero...



DTD – Document Type Definition

- Describe la “gramática” del documento.
- **Define los elementos del documento XML:**
    - Qué elementos.
    - Qué atributos.

Ejemplo de DTD

```
<?xml version="1.0"?>
<!DOCTYPE ficha [
<!ELEMENT  ficha (nombre+, lugar+, estadioprincipal?)>
<!ELEMENT  nombre (#PCDATA)>
<!ELEMENT  lugar (#PCDATA)>
<!ELEMENT  estadioprincipal EMPTY> ]>
<ficha>
        <nombre>Roland Garros</nombre>
        <lugar>Paris</lugar>
        <estadioprincipal archivo="estadio.jpeg"/>
</ficha>
```

- +: uso obligatorio y múltiple (uno o más)
- *: opcional y múltiple (cero o más)
- ?: opcional pero singular (cero o uno)
- |: or (opciones)
- #PCDATA

Existen dos tipos de documentos XML

Documentos bien formados:

- Respetan la sintaxis básica
- Un XML mal formado: ver-xml-mal.xml

Documentos válidos:

- Respetan un DTD
- Un XML no válido: ver-xml-invalido.xml
- Usamos un validador-XML

¿Qué pasa si quiero ...?

- Especificar que un elemento dado es de tipo fecha o un número.
- **Asegurarme que un elemento únicamente puede aparecer 3 veces?**
    - Y si quiero indicar que aparezca un mínimo de 3 veces y un máximo de 100?.
- Agregar algún nuevo tipo de elemento o atributo.

Cuando el DTD no alcanza ...

Usamos Schemas

Schemas

- También permiten definir la estructura de un documento XML
- A diferencia de los DTD, utiliza sintaxis XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element name="ficha">
<xsd:complexType>
        <xsd:sequence>
          <xsd:element name="nombre" type="xsd:string"/>
          <xsd:element name="lugar" type="xsd:string"/>
          <xsd:element name="fechaInicio"  type="xsd:date"/>
        </xsd:sequence>
        <xsd:attribute name="tipo" type="xsd:string" use="required" />
  </xsd:complexType>
......
</xsd:schema>
```

Otras notaciones

JSON - YAML

¿De qué estábamos hablando?

- Intercambio de información entre aplicaciones.
- Definición de datos estructurados.
- Procesamiento en el cliente.
- Procesamiento en el servidor.

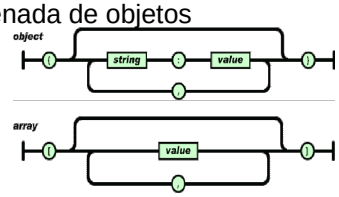
JSON – JavaScript Object Notation

- Formato alternativo para el envío y recepción de datos.

- **Es un subconjunto de la notación literal de objetos de Javascript.**
    - Si bien aún no vimos JS, veremos cómo es la notación.
- Se lo conoce también como LJS.
- Es un formato ligero de intercambio de datos.
- Muy popular.

Sintaxis JSON

- **JSON está constituído por dos estructuras:**
    - Objetos: Una colección de pares de nombre/valor
    - Arreglos: Una lista ordenada de objetos



Ejemplo sencillo

```
libro= {
        "titulo": "La casa de los espiritus",
        "precio": "21.90",
        "autor": "Isabel Allende",
        "paginas": "350"
        }

coleccion= [libro1, libro2]
```

Libros en XML

```xml
<?xml version="1.0" encoding="ISO8859-1" ?>
<Libros>
        <libro>
                <titulo>La casa de los espiritus</titulo>
                <precio>21.90</precio>
                <autor>Isabel Allende</autor>
                <paginas>350</paginas>
        </libro>
        <libro>
                <titulo>El socio </titulo>
                <precio>21.90</precio>
                <autor>John Grisham</autor>
                <paginas>504</paginas>
        </libro>
</Libros>
```

Libros en JSON

```json
"Libros": {
        "libro":[
                {"titulo": "La casa de los espiritus",
                "precio": "21.90",
                "autor": "Isabel Allende",
                "paginas": "350"
                },
                {"titulo": "El socio",
                "precio": "21.90",
                "autor": "John Grisham",
                "paginas": "504"
                },
        ]
}
```

¿Cómo proceso?

- **En el servidor: usando PHP.**
    - PHP tiene una extensión para JSON: http://php.net/manual/es/book.json.php
- En el cliente: usando Javascript.

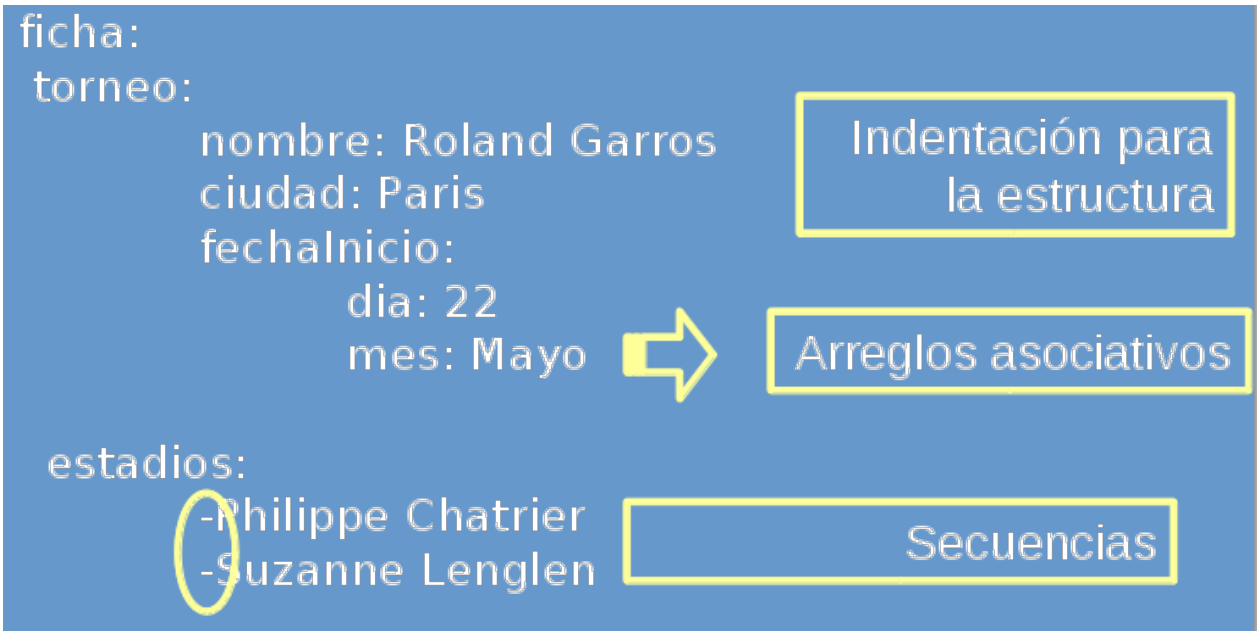
Ejemplo

```php
<?php
$informatica = array("nombre"=>'Licenciatura en Informática', "duracion"=>'5');
$sistemas = array("nombre"=>'Licenciatura en Sistemas', "duracion"=>'5');
$ingenieria=array("nombre"=>'Ingeniería en Computación', "duracion"=>'5');
$apu=array("nombre"=>'Analista Programador Universitario',"duracion"=> '3');
$carreras = array(
        "informatica" => $informatica,
        "ingenieria" => $ingenieria,
        "sistemas" => $sistemas,
        "apu" =>$apu);
?>
<div>
<p> El objeto JSON desde PHP:
<p> <?php print json_encode($carreras);?>
</div>
```

- Ver JSON.php

YAML – YAML Ain't Markup Language

- Es un superconjunto de JSON que trata de superar algunas de las limitaciones de éste.

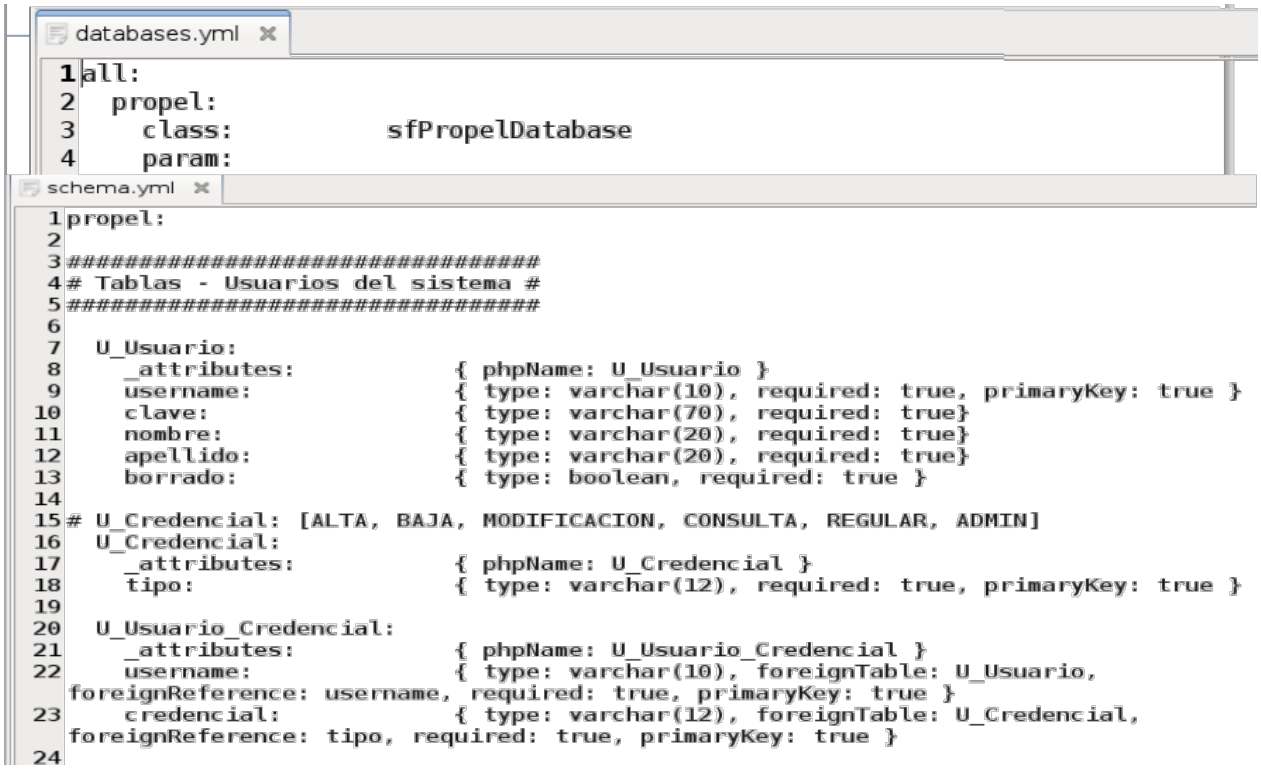


YAML - Notación resumida

¿Nos suena conocido?

```
ficha:
        torneo:{nombre: Roland Garros, ciudad: Paris,
        fechaInicio: { dia: 22, mes: Mayo },
        estadios: [Philippe Chatrier, Suzanne Lenglen]
```

Ejemplos de usos



databases.yml

```
all:
  propel:
    class:          sfPropelDatabase
    param:
```

schema.yml

```
propel:

###################################
# Tablas - Usuarios del sistema #
###################################

  U_Usuario:
    _attributes:            { phpName: U_Usuario }
    username:               { type: varchar(10), required: true, primaryKey: true }
    clave:                  { type: varchar(70), required: true}
    nombre:                 { type: varchar(20), required: true}
    apellido:               { type: varchar(20), required: true}
    borrado:                { type: boolean, required: true }

# U_Credencial: [ALTA, BAJA, MODIFICACION, CONSULTA, REGULAR, ADMIN]
  U_Credencial:
    _attributes:            { phpName: U_Credencial }
    tipo:                   { type: varchar(12), required: true, primaryKey: true }

  U_Usuario_Credencial:
    _attributes:            { phpName: U_Usuario_Credencial }
    username:               { type: varchar(10), foreignTable: U_Usuario,
foreignReference: username, required: true, primaryKey: true }
    credencial:             { type: varchar(12), foreignTable: U_Credencial,
foreignReference: tipo, required: true, primaryKey: true }

```

Libros en JSON

```
"Libros": {
        "libro":[
                {"titulo": "La casa de los espiritus",
                "precio": "21.90",
                "autor": "Isabel Allende",
                "paginas": "350"
                },
                {"titulo": "El socio",
                "precio": "21.90",
                "autor": "John Grisham",
                "paginas": "504"
                },
        ]
}
```

Libros en YAML

```
libros:
        -titulo   : "La casa de los espiritus"
         precio   : 21.90
         autor    : "Isabel Allende"
         paginas  : 350
        -titulo   : "El socio"
         precio   : 21.90
         autor    : "John Grisham"
         paginas  : 504
```

En resumen ....

- Importancia del intercambio de datos en un formato estándar.
- Procesamiento eficiente.
- Legibilidad para el desarrollador.
- Depende del uso.

---

class: tabla

Algunas comparativas

- Un análisis con algunos números
- Generación de 2.000.000 de usuarios con id y name.
- XML: <user><id>%d</id><name>%s</name></user>
- JSON: {id:%d,name:"%s"}
- Resultados:

	Text	Gzip	Zip duration
XML	91.78M	18.74M	3.38s
JSON	49.78M	17.09M	2.78s
XML overhead	84.38%	9.62%	21.3%

Y entonces... ¿qué usamos?

Como siempre ... depende...

- **XML es más que un formato para describir información**
    - Existen: DTDs y Schemas, XSL, XPath, etc.
- Simplicidad vs. legibilidad vs. performance

Referencias

XML http://www.w3.org/XML/ http://www.w3.org/TR/REC-xml/

Schemas http://www.xml.com/pub/a/2000/11/29/schemas/part1.html http://www.w3.org/XML/Schema.html

JSON http://www.json.org

YAML http://www.yaml.org

Procesamiento en el cliente

Programando scripts

- Los scripts son pequeños programas que se incluyen en la página web.
- Se delimitan entre las etiquetas **<script>** y **</script>**.
- Si el navegador NO puede ejecutar scripts, existen las etiquetas **<noscript>** y **</noscript>**.

Programando scripts (cont.)

- Es posible escribir el código en un archivo externo al documento HTML.

```
<script type="text/javascript"
                src="misScripts.js">
</script>
```

- O, en HTML 5, si no indico el atributo type, asume Javascript.

```
<script  src="misScripts.js">
</script>
```

Programando scripts (cont.)

```
<html> <head>
<title>Scripts – Proyecto de Software</title>
<meta http-equiv="content-script-type"
                content="text/tcl">

<script type="text/vbscript"
                src="http://someplace.com/progs/vbcalc">
</script>
</head> <body>

<script type="text/javascript">
...some javascript...
</script>
</body></html>
```

---

class: destacado

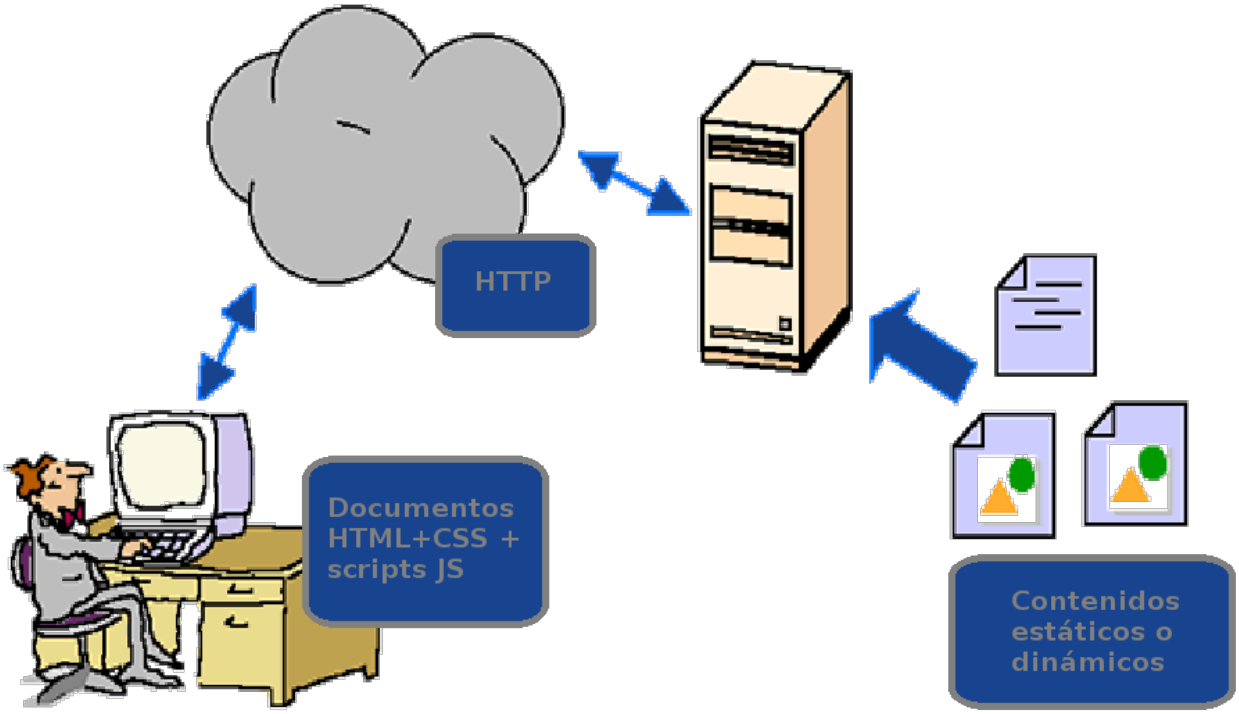
Javascript

- JavaScript (abreviado comúnmente "JS") es un lenguaje de programación interpretado.
- Dialecto del estándar ECMAScript.
- Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.
- Se utiliza principalmente en su forma del lado del cliente (client-side), implementado como parte de un navegador web.
- Existe JavaScript del lado del servidor y se usa en algunas aplicaciones no web, como el acrobat reader.
- Sintaxis similar al C, aunque adopta algunas caracteristicas de Java.

No tiene relación con Java, el nombre a veces se presta a confusión.

Javascript

- Es interpretado: el intérprete de Javascript está contenido en el navegador.



Javascript

- Es multiplataforma.
- Surgió como Livescript (creado por la empresa Netscape) y luego, junto con la empresa Sun, se convirtió en Javascript.
- El estándar es el **ECMA 262**
- Javascript es una marca registrada de Oracle
- La implementación de Microsoft se llama JSCRIPT, aunque se tratan de manera indiferente generalmente.
- Es case-sensitive, la sintaxis es similar a la del Lenguaje C o Java.
- Hay varios engines: V8 (Google), Nitro(Apple), SpiderMonkey (Mozilla Firefox) y Chakra (Internet Explorer)

Hola Mundo!!!

```
function saludar(){
        alert("Hola Mundo!!")
}
```

- Veamos el holaMundo

Objetos en Javascript

- Paradigma **"basado en objetos" (no en clases)**
- Cada objeto tiene asociado propiedades y métodos.
- **¿Cómo se referencian? Mediante calificación:**
    - objeto.propiedad
    - objeto.método
- La herencia se realiza a través de **prototipos**.
- Las propiedades y los métodos pueden agregarse al objeto en forma dinámica.

Aspectos básicos

- Maneja excepciones: mecanismo **try - catch**
- **Funciones útiles**
    - **eval(unaCadena):** ejecuta la expresión o sentencia contenida en unaCadena.
    - **parseInt(unaCadena, base):** convierte unaCadena al valor numérico asociado usando como base el parámetro base.
    - **parseFloat(unaCadena):** convierte una Cadena en un número real.
    - **isNaN(unaCadena):** retorna true si una Cadena No es un Número.

Probemos en la consola

```
parseInt("1234")
parseInt("11",2)
parseInt("0x10")
parseInt("hola")
Math.sqrt(9)
Number.MAX_VALUE

var  x=  false;
if (x) alert("hola")
```

Seguimos probando...

```
"proyecto".length
"proyecto".charAt(2)
"proyecto de desarrollo".replace("desarrollo", "software")
"proyecto".toUpperCase()

var arreglo = new Array("uno", 2, "tres")
arreglo[3]="3333"
var frutas= new Array()
frutas["citricos"]=new Array("naranja", "pomelo")
frutas["otros"]=new Array("manzana", "pera")
frutas["citricos"]
frutas.citricos
```

Seguimos probando...

```
fecha = new Date()
hoy=new Date(2014, 8, 30)

dias=new Array("dom","lun","mar","mier","jue","vie","sab")
alert(dias[hoy.getDay()])
```

Más ejemplos en https://developer.mozilla.org/en-US/docs/Web/JavaScript/A_re-introduction_to_JavaScript

---

Ejemplo objetos en Javascript

- Ejemplo completo objetos

```
<script>
function Libro(titulo, precio, autor, paginas){
        this.titulo=titulo;
        this.precio=precio;
        this.autor=autor;
        this.paginas=paginas;
                }

var libros= new Array();
libros[0]= new Libro("PHP for dummies",
        30,"Autor desconocido",10);
libros[1]= new Libro("ASP for dummies",
        35.90,"Otro desconocido",1000);
</script>
```

¿Volvemos a JSON?

- Usando Objetos JSON
- Otro ejemplo pero con una función en el objeto complejo
- Usando eval
- Usando JSON.parse
- Usando JSON.php+javascript

Clases de PHP y JSON

```
<?php
class PruebaJSON{
        public $variablePublica;
        public $variablePublica2;
        private $variablePrivada;
        public function __construct($a="hola",$b="mundo",$c="este es el secreto")
                {       $this->variablePublica=$a;
                        $this->variablePublica2=$b;
                        $this->variablePrivada=$a; }}?>
<div><p> La clase en JSON:
<p> <?php $test=new PruebaJson(); print json_encode($test);?>
<script type="text/javascript">
var test = <?php echo json_encode($test) ?>;
alert(test.variablePublica);
</script>
```

- Ver: JSON.PHP.clases

+Info en http://www.json.org/js.html