



Cátedra: Diseño y Aplicaciones en la Web

Trabajo: Trabajo de Investigación – jQuery VS MooTools

Alumno: Ulises C. Ramirez

Objetivos de la Presentación

- Ver el contexto histórico para la creación de las herramientas.
- Entender a que tareas esta orientada cada una.
- Basada en la información recolectada definir cual es la mejor herramienta para cada caso.

Índice

- Preliminares
- JavaScript
- Librerías JavaScript
- Comparativa
- Conclusiones

Índice

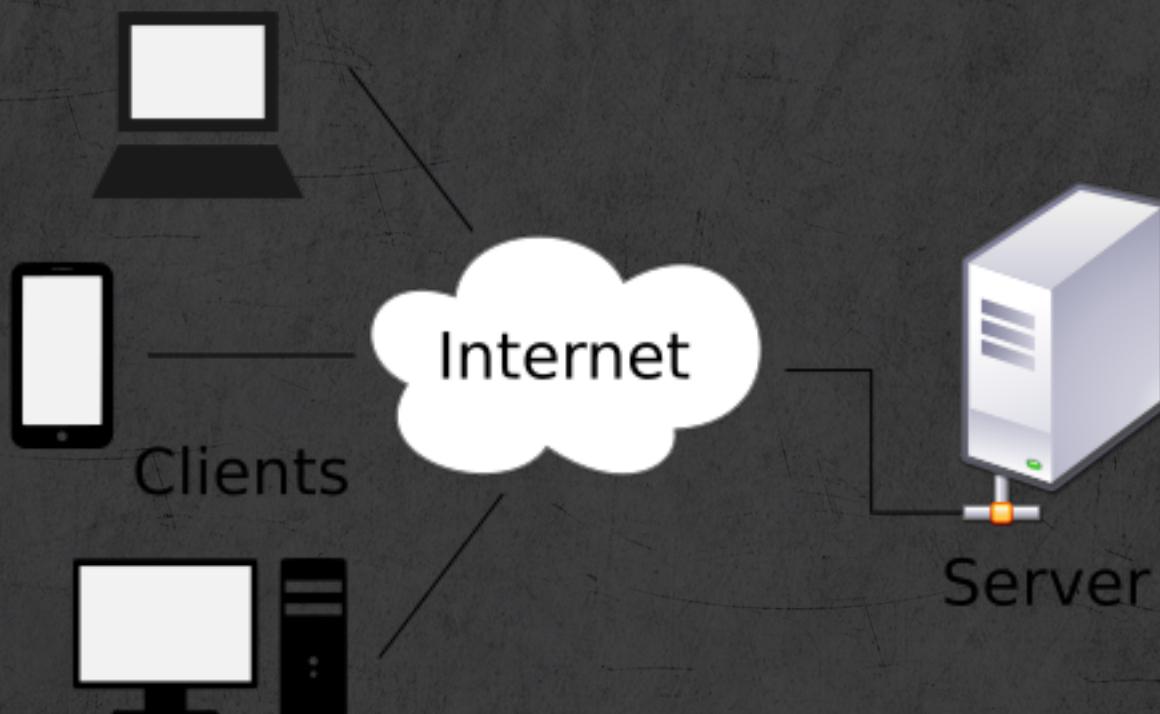
- •Preliminares
- JavaScript
- Librerías JavaScript
- Comparativa
- Conclusiones

Preliminares (1)

- Arquitectura C/S
- Motores de Navegador
 - Funcionamiento Conceptual
- Navegadores Web

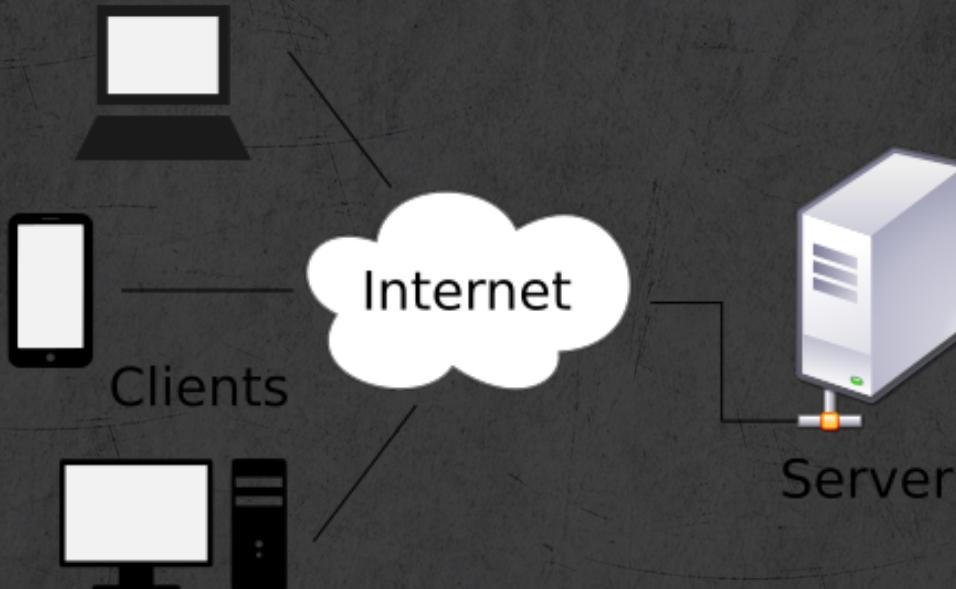
Preliminares (2)

Arquitectura Cliente-Servidor (Servidor web)

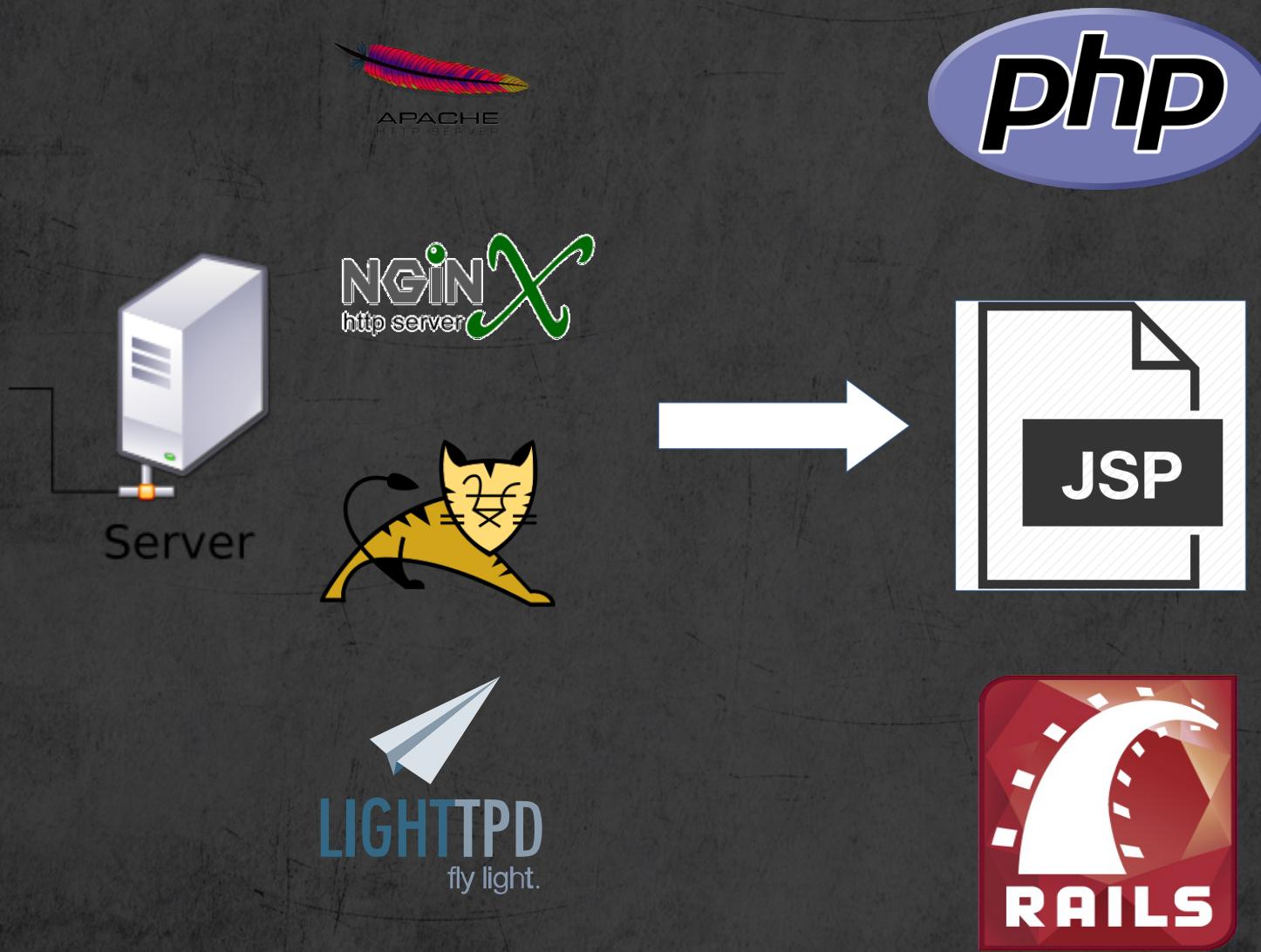


Preliminares (2)

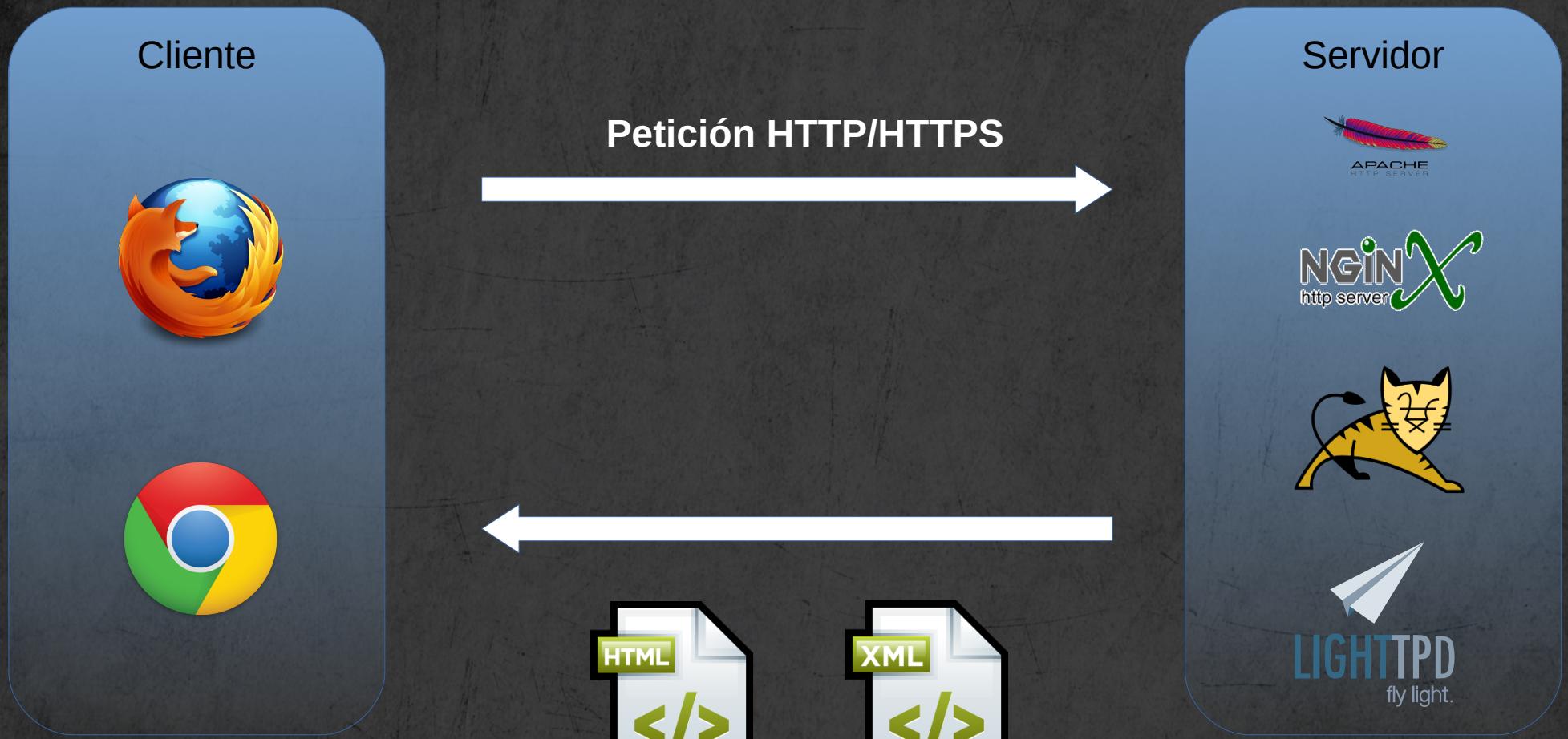
Arquitectura Cliente-Servidor (Servidor web)



Preliminares (3)



Preliminares (2)

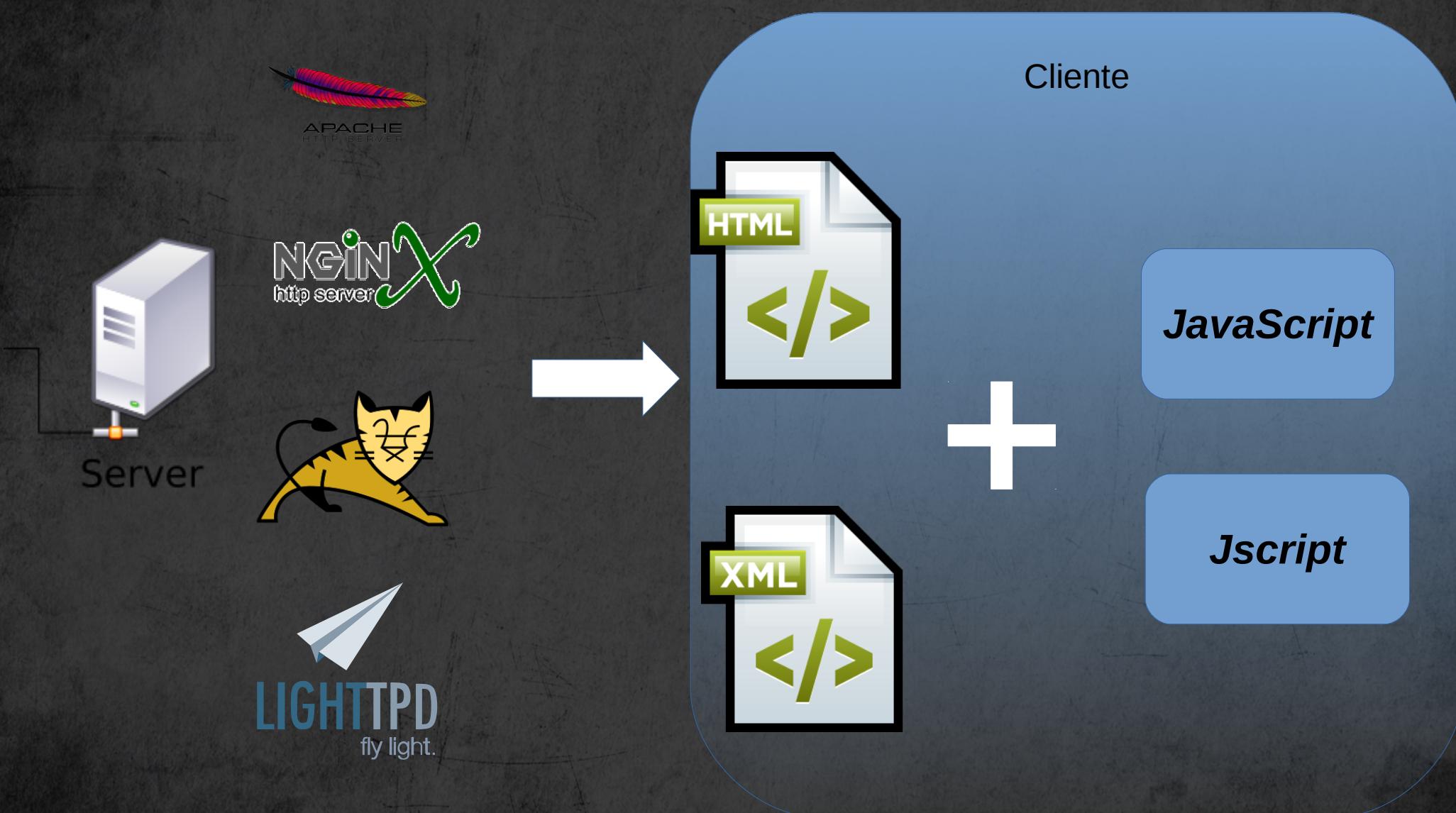


Ejemplo 1

Lenguaje: PHP, HTML

Objetivo: Demostrar el paso de parámetros mediante los métodos de HTTP, Además de visibilizar como el servidor cambia al usuario de directorio con el fin de brindarle la respuesta que deseada

Preliminares (3)



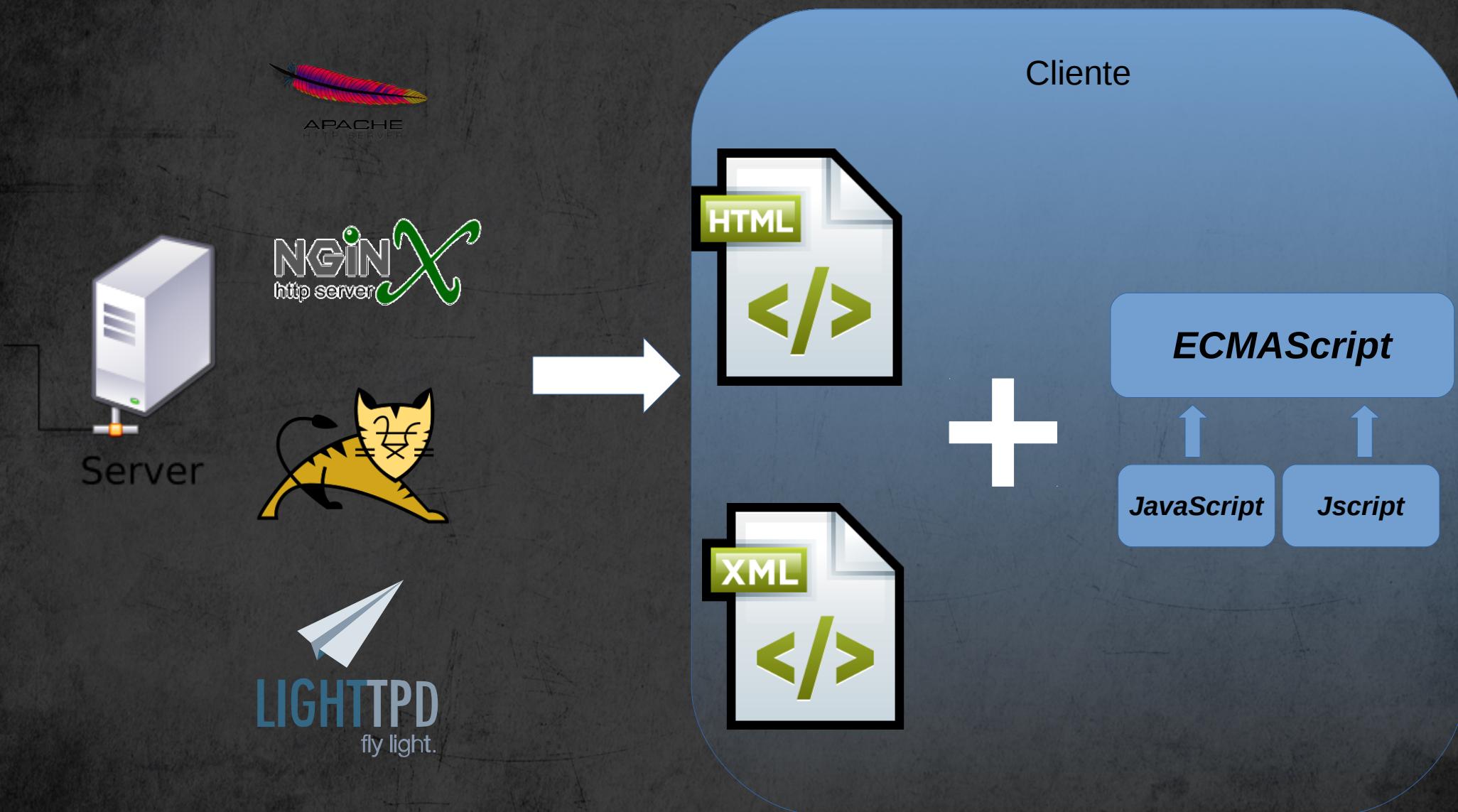
Preliminares (3)

JavaScript

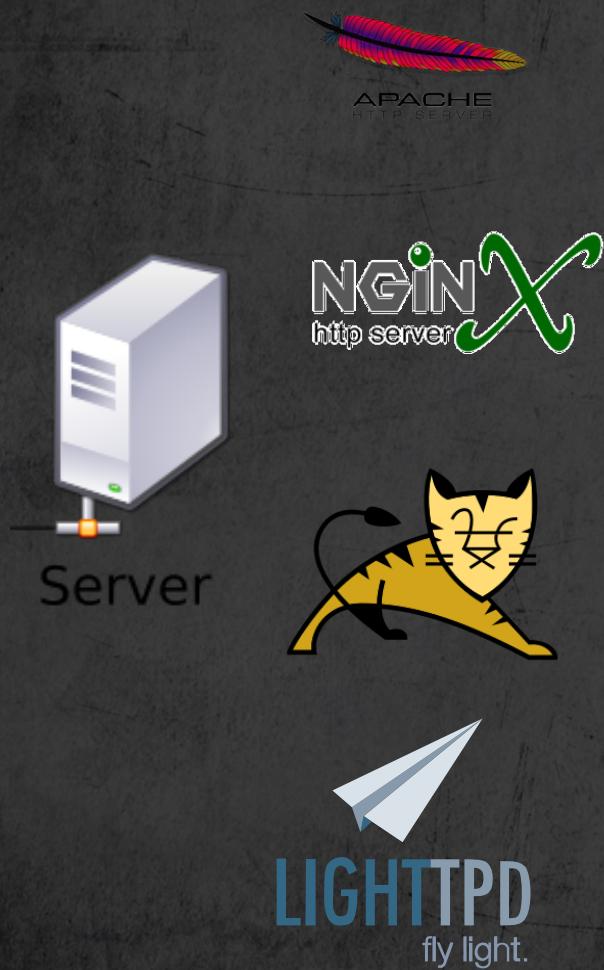


Jscript

Preliminares (3)



Preliminares (3)



Preliminares (3)

Navegadores



Preliminares (3)

Motor web Gecko



Picasa



Instantbird



Índice

- Preliminares
- • JavaScript
- Librerías JavaScript
- Comparativa
- Conclusiones

JavaScript (JS)

JS

- Lenguaje de **Programación Funcional**
- Brinda Objetos Nativos tales como String, Integer, Array.
- Modelo de **Herencia** ‘especial’ (Prototypal Inheritance)

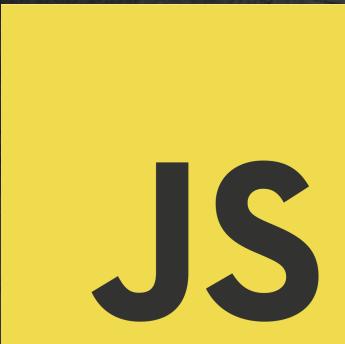
JS

JavaScript (JS)

- Tiene la capacidad de **Modificar** HTML, CSS, Reacción a eventos, Creación de eventos nuevos
- Capacidad de **transmitir** el estado del navegador mediante documentos HTML (**Asincronismo**)

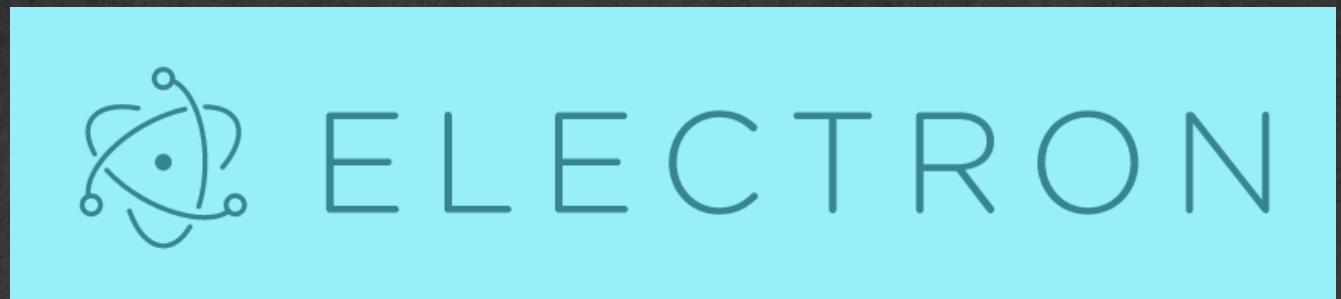
JavaScript (JS)

- No es **PURA Y EXCLUSIVAMENTE** del Navegador



JavaScript (JS)

JS



JavaScript (JS)



Características

- Herencia
- Auto-referencia

Ejemplo 2

Lenguaje: PHP, HTML, CSS

Objetivo: Demostrar un escenario en el cual el uso de JS podría ahorrar datos y brindar una mejor experiencia a la hora del uso de un sistema web.

JavaScript (JS)

Incompatibilidad

JavaScript (JS)

Incompatibilidad



=

Librerías JS

Índice

- Preliminares
- JavaScript
- • Librerías JavaScript
- Comparativa
- Conclusiones

Librerías JS



write less, do more.



Librerías JS



write less, do more.



Librerías JS



“jQuery es una Biblioteca de JavaScript rápida y concisa que simplifica el recorrido, manejo de eventos, animación, y interacciones Ajax en el documento HTML para un rápido desarrollo web. jQuery está diseñado para cambiar la manera que tu escribes JavaScript.” – Página oficial JQuery

Librerías JS



“MooTools es un compacto, modular, framework Orientado a Objeto JavaScript diseñado para el programador JavaScript intermedio a avanzado. Este permite escribir un código potente, flexible, y cross-browser⁴ con su elegante, bien documentada y coherente API.” – Página oficial MooTools

Índice

- Preliminares
 - JavaScript
 - Librerías JavaScript
-
- Comparativa
 - Conclusiones

Comparativa

- 1) JavaScript y el DOM
- 2) Mismas Tareas
- 3) Estilos y Sintaxis
- 4) Reutilización Código
- 5) Extensión e Implementación de Clases

Comparativa

2) Mismas Tareas

```
// JQuery
$(document).ready(function() {
    $("#JQ").click(function(event) {
        alert("Esto es JQuery!");
    });
}); // ventana que dice: Esto es JQuery al hacer click en el elemento con id=JQ
```

```
// MooTools
window.addEvent('domready', function() {
    $$('#MT').addEvent('click', function(event) {
        alert('Esto es MooTools!');
    });
}); // ventana que dice: Esto es MooTools al hacer click en el elemento con id=MT
```

Comparativa

2) Mismas Tareas

```
// JQuery
$(document).ready(function() {
    $("#divId").hover(function() {
        $(this).addClass("green");
    },
    function() {
        $(this).removeClass("green");
    });
}); // Al pasar el cursor del mouse por el elemento con id=divId se le agrega una clase (green) a este elemento

// MooTools
window.addEvent('domready',function() {
    $$('divId').addEvents({
       mouseenter: function() {
            this.addClass('green');
        },
       mouseleave: function() {
            this.removeClass('green');
        }
    });
}); // Al pasar el cursor del mouse por el elemento con id=divId se le agrega una clase (green) a este elemento
```

Comparativa

3) Estilos y Sintaxis

Emulando la sintaxis de jQuery en MooTools

Ejemplo 5:

```
Element.implement({
    hover : function(enter, leave){
        return this.addEvents({ mouseenter : enter, mouseleave : leave });
    }
});

// en otra parte del código se procede a llamar a la función con nueva sintaxis:
$$('#divId').hover(function(){
    this.addClass('green');
},
function(){
    this.removeClass('green');
});
```

Comparativa

4) Reutilización de Código

jQuery

```
// Definimos la función que vamos a reutilizar
function diccionario(contenedor, term, def) {
    $(contenedor).find(term).hide().end().find(def).click(function() {
        $(this).next().slideToggle();
    });
}

// utilizamos la función definida anteriormente
$(document).ready(function() {
    diccionario('#dic', 'dt', 'dd');
});
```

Comparativa

4) Reutilización de Código

MooTools

```
String.implement({
    trim: function() {
        return this.replace(/^\s+|\s$/g, '');
    }
});
// Lo unico que queda, es al tener un objeto de tipo String hacer lo siguiente:
objTipoString.trim();
```

Comparativa JQuery – Plugins

```
JQuery.fn.diccionario = fuction(opc) {
    var settings = jQuery.extend({
        term: 'dt',
        def: 'dd'
    }, opc);
    // "this" es el contexto actual; en este caso, los elementos que queremos transformar a un layout
    de diccionario
    $(this).find(settings.terms).hide().end().find(settingsdefinitions).click(function() {
        $(this).next().slideToggle();
    });
    return this;
};

// Procedemos a la utilización del plugin mediante:
$('#dic').diccionario();
```

Comparativa

5) Extension/Implementación de Clases

jQuery

```
jQuery.fn.ajaxDiccionario = function(opc) {
    var settings = jQuery.extend({
        //algunas opciones específicas de ajax para obtener la información
        url: '/getfaq.php'
        def: 'dd'
    }, opc);
    // "this" en el contexto actual; en este caso, el elemento que queremos
    transformarlo en el layout diccionario
    $(this).find(settings.def).click(function() {
        $(this).load('...'); //la lógica que obtiene el contenido
    });
    this.diccionario(); //la llamada a nuestro plug-in original
});
```

Comparativa

5) Extension/Implementación de Clases

MooTools

```
var Persona = new Class({  
    initialize: function(nom, edad) {  
        this.nombre = nom;  
        this.edad = edad;  
    },  
    energia: 1,  
    comer: function() {  
        this.energia = this.energia + 1;  
    }  
});
```

Comparativa

5) Extension/Implementación de Clases

MooTools

```
var Alumno = new Class({  
    extends: Persona,  
    initialize: function(nom, edad) {  
        this.grado = 0;  
        this.materiasAprobadas = 0;  
        this.parent(nom, edad);  
    },  
    aprueba: function() {  
        this.materiasAprobadas = this.materiasAprobadas + 1;  
    },  
    pasaGrado: function () {  
        this.grado = this.grado + 1;  
    }  
});
```

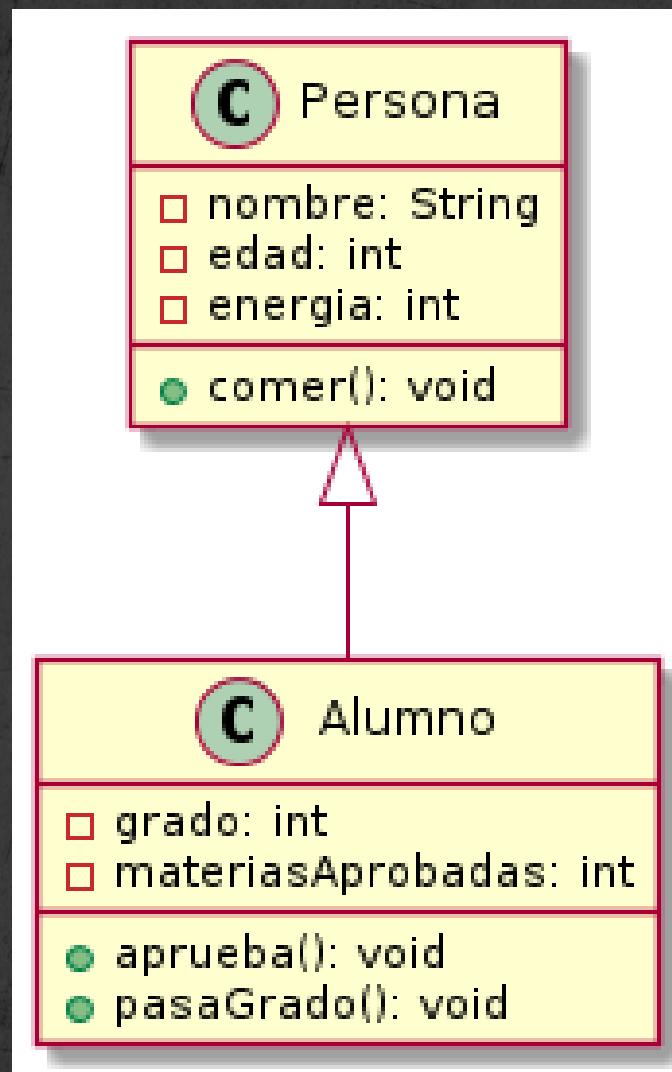
Comparativa

5) Extension/Implementación de Clases

MooTools

```
// Utilización
var unAlumno = new Alumno('Juan', 20);
// unAlumno.nombre = Juan
// unAlumno.materiasAprobadas = 0
unAlumno.aprueba;
// unAlumno.materiasAprobadas = 1
```

MooTools



Ejemplo 3 y 4

Lenguaje: PHP, HTML, CSS, JavaScript

Objetivo: Demostrar como favorece la implementación de JavaScript a el escenario propuesto en el ejemplo 2, mediante el uso de jQuery (EJ3) y MooTools(EJ4)

Índice

- Preliminares
- JavaScript
- Librerías JavaScript
- Comparativa
- • Conclusiones

Conclusiones



- Fácil de aprender
- Gran conjunto de herramientas para el manejo del DOM
- Extensión de plugins puede resultar problemática



- Ofrece flexibilidad
- Ofrece extensibilidad
- Brinda herramientas nativas para la personalización de los prototipos primitivos de JS
- Fácil implementación de un modelo de clases

Conclusiones



Manejo de los elementos del DOM, operaciones básicas con los prototipos nativos de JavaScript



Proyectos cuyo alcance escapa al manejo del DOM y se extiende hacia áreas en las cuales se necesitará más orientación a objetos