Arquitecturas Distribuidas

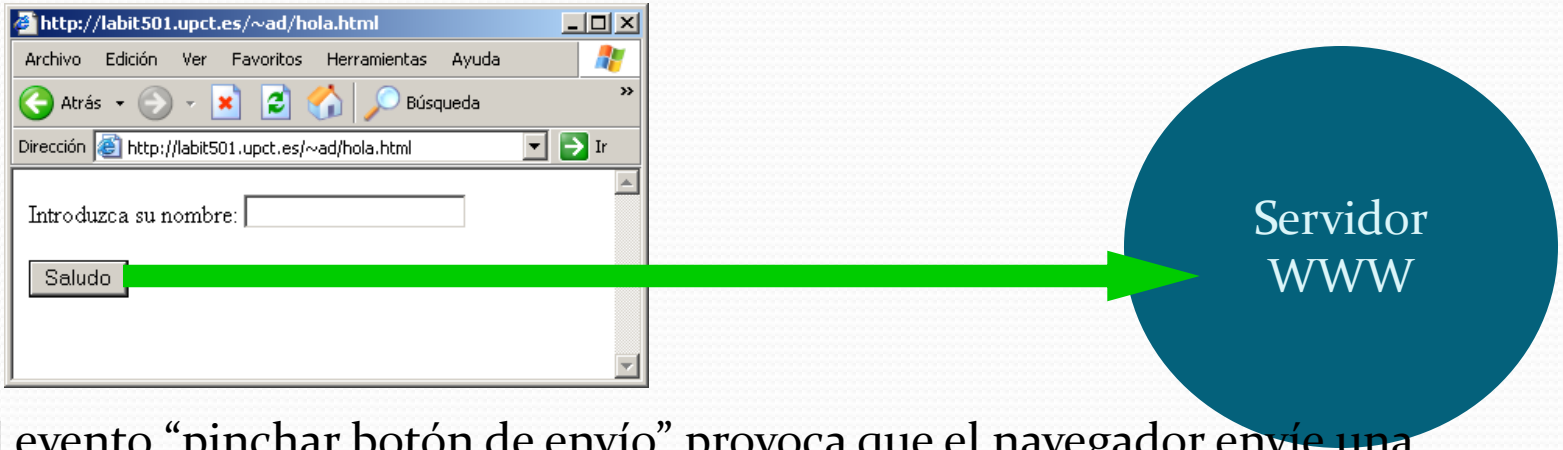
Tema 4. IV- Lenguajes de *script* en el cliente

IV. Lenguajes de *script* en cliente

1. Motivación
2. Funcionamiento
3. AJAX

Motivación (I)

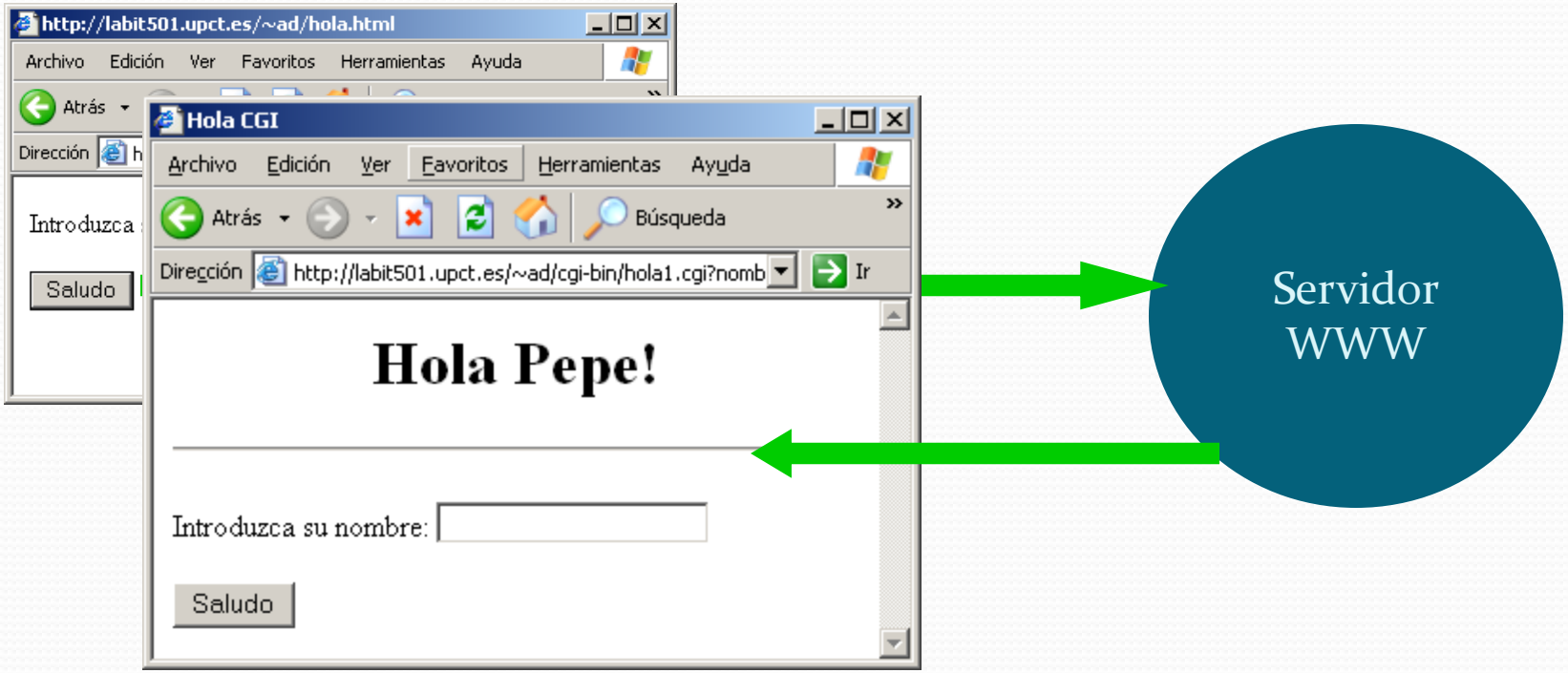
- El contenido HTML no se modifica dinámicamente como resultado de las acciones del usuario, a no ser que se invoque una nueva petición al servidor y se reciba nuevo contenido
- El funcionamiento de HTTP obliga a ejecutar **una petición y respuesta** por cada acción del usuario
- Petición dirigida a una URL concreta



El evento "pinchar botón de envío" provoca que el navegador envíe una petición al servidor, es decir, se construye una petición a partir de **una nueva URL**

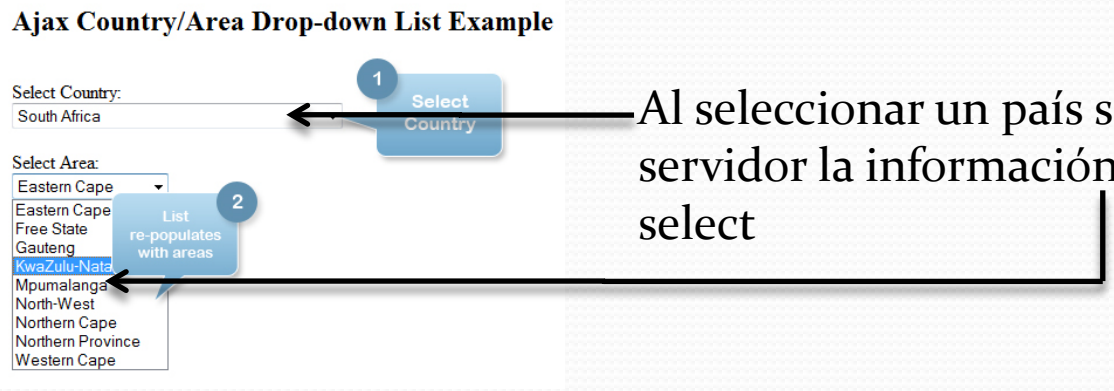
Motivación (II)

- Cada "evento" de usuario es procesado por el servidor que devuelve una respuesta adecuada



Motivación (III)

- Con este modelo, ¿Podemos hacer que un elemento de un menú cambie de color al pasar el ratón sobre él?
- Mejor aún, ¿podemos conseguir que cuando el usuario pase el ratón sobre un elemento se obtenga información del servidor y se muestre?
- Por ejemplo, al usar varias listas de selección, a medida que el usuario selecciona una categoría, se rellena automáticamente las opciones disponibles en las otras.



Al seleccionar un país se carga desde el servidor la información de área en el otro select

Motivación (IV)

- Con el funcionamiento original de un navegador y HTTP:
  - Realmente, pocos “eventos” disponibles. Prácticamente sólo pinchar un enlace (o un botón de envío) = introducir una nueva URL y pedir el documento.
  - Interacción con el usuario muy limitada.
    - Limita el desarrollo de interfaces de usuario ricas y degrada la “experiencia” del usuario. La aplicación resulta “menos interactivas”. Las aplicaciones de escritorio cuentan con ventajas en ese sentido: tienen un conjunto muy rico de eventos a los que responder.
    - No permite el desarrollo de aplicaciones que devuelvan un contenido en función de la actividad del usuario sobre la ventana
  - El servidor tiene que encargarse de realizar todo el procesado, por sencillo que sea (comprobar que un campo se ha rellenado correctamente, por ejemplo), con lo que se puede sobrecargar al servidor de peticiones.

Motivación (V)

- Solución:
  - Los documentos pueden contener instrucciones que se ejecuten en el lado del cliente.
    - Java → Lenguaje "pesado", pero compatible entre equipos. Se usa un plug-in: se invoca el ejecutable de la máquina virtual de Java, el llamado Java Runtime Environment instalado en el equipo.
    - JavaScript y similares → Lenguaje "ligero", pero con incompatibilidades entre equipos. Lenguaje interpretado. Intérprete implementado por el navegador.
      - Estándar ECMA-262. El motor del navegador (y otras aplicaciones) implementa el intérprete (según este estándar) que recibe su propio nombre (Javascript es el más genérico, pero la implementación de IExplorer se denomina JScript, por ejemplo).
      - Al ser estándar "debería" funcionar igual en cada navegador. Cada vez hay menos incompatibilidades entre implementaciones.

Funcionamiento (I)

- Orientado a **eventos**: La ocurrencia del evento desencadena una acción, es decir, se invoca una función declarada en el código del *script* que acompaña el documento HTML
  - Algunos tipos de eventos:
    - Cargar pagina completamente
    - Ratón pasa sobre un elemento
    - Ratón hace click sobre un elemento
    - Temporizadores
    - Etc

Funcionamiento (II)

- Al capturar un evento, se invoca una función del código (*script*) descargado.
- Lo habitual es que cambie el valor de algún atributo en la página, que compruebe si los valores de un formulario son correctos antes de enviarlos, etc.
- El tipo de acciones que se pueden realizar normalmente está limitado por motivos de seguridad

Funcionamiento (III)

```
<html>

<script language="JavaScript">
<!--
      function dihola() {
            alert("Hola a todos!");
      }
-->
</script>

<body onLoad="dihola();">

</body>

</html>
```

Funcionamiento (IV)

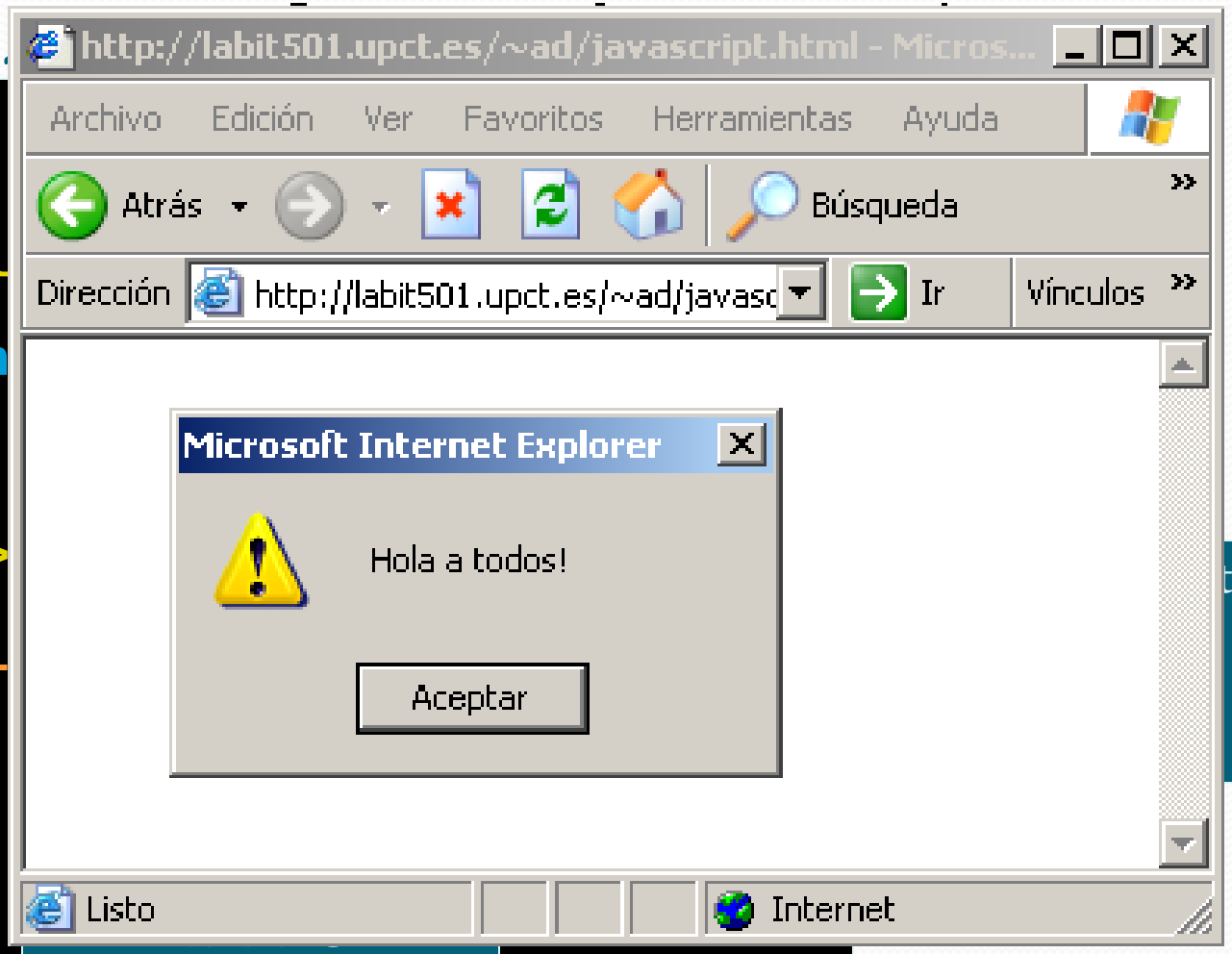
```
<html>

<script language="JavaScript">
<!--
      function dihola() {
            alert("Hola a todos!");
      }
-->
</script>

<body onLoad="dihola();">

</body>

</html>
```

Código de la función que se ejecuta

EVENTO: documento completamente cargado

Funcion

```
<html>

<script l
<!--
      fun

      }
-->
</script>

<body onL

</body>

</html>
```



Funcionamiento (VI)

- Programación similar a Java o C.
  - `while`
  - `do..while`
  - `for`
  - `etc`
- Orientados a objetos
  - No se hace distinción entre “clase” y “objeto”.
  - No hay que declarar el tipo de los datos.
- Intérprete implementado por el navegador: el rendimiento depende de la calidad de la implementación

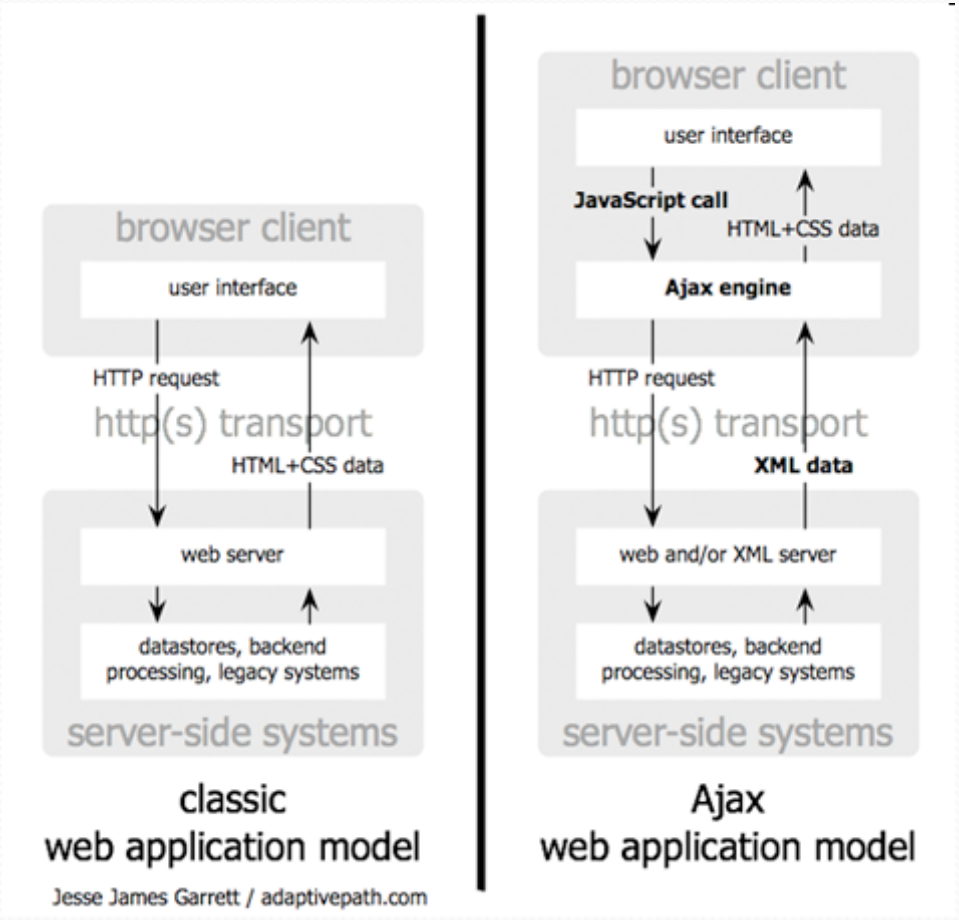
AJAX (I)

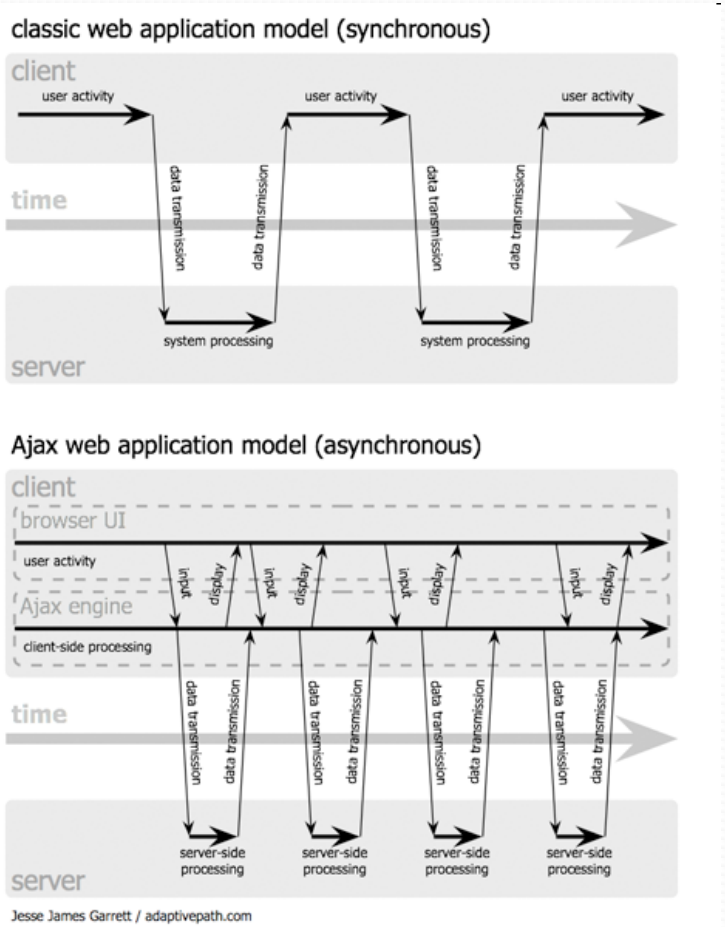
- Asynchronous Javascript and XML (AJAX)
  - Grupo de técnicas para crear aplicaciones web más interactivas.
  - Utiliza tecnologías estándar: HTML y CSS, interfaz DOM, XML como lenguaje de intercambio y javascript.
- Se solicitan datos del servidor de manera asíncrona y en segundo plano
  - Asíncrona: se realiza una petición al servidor y se continúa la ejecución del código. Cuando la respuesta está disponible se invoca automáticamente la función que la tratará.
  - Por el contrario, el modelo tradicional HTTP consiste en peticiones síncronas: se realiza una petición y no se hace nada más hasta recibir la respuesta
  - No es necesario solicitar otra URL para que cambie el contenido del documento. Como resultado del procesado de la respuesta asíncrona del servidor, normalmente se modifica el contenido.

AJAX (II)

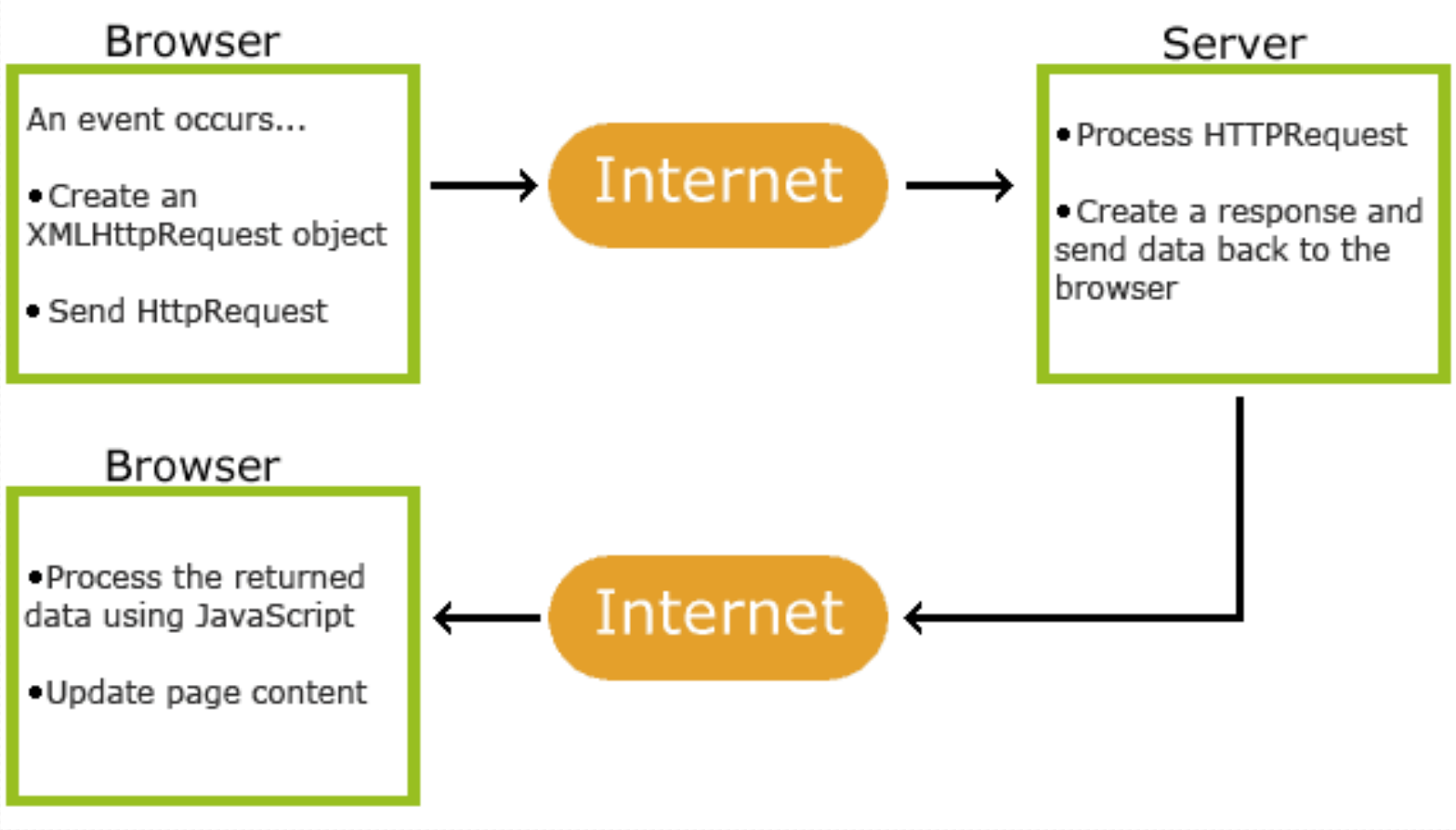
- Para la comunicación se utiliza el objeto XMLHttpRequest de Javascript.
  - Objeto javascript que permite realizar peticiones HTTP a un servidor. Interfaz definida en la especificación DOM.
  - Una función que procesa un evento crea una instancia de este objeto para realizar un petición asíncrona al servidor.
- Los datos intercambiados están en formato XML. Aunque se puede recibir texto plano u otros formatos más eficientes (como JSON).

AJAX (II)


browser client
user interface
HTTP request
http(s) transport
HTML+CSS data
web server
datastores, backend processing, legacy systems
server-side systems
classic web application model
Jesse James Garrett / adaptivepath.com
browser client
user interface
JavaScript call
HTML+CSS data
Ajax engine
HTTP request
http(s) transport
XML data
web and/or XML server
datastores, backend processing, legacy systems
server-side systems
Ajax web application model


classic web application model (synchronous)
client
user activity
user activity
user activity
data transmission
data transmission
data transmission
data transmission
time
system processing
system processing
server
Ajax web application model (asynchronous)
client
browser UI
user activity
input
display
Ajax engine
client-side processing
data transmission
server-side processing
time
server
Jesse James Garrett / adaptivepath.com

AJAX (III)


Browser
An event occurs...
• Create an XMLHttpRequest object
• Send HttpRequest
Internet
Server
• Process HTTPRequest
• Create a response and send data back to the browser
Internet
Browser
• Process the returned data using JavaScript
• Update page content

AJAX (IV)

- Ventajas
  - Mejoran la experiencia del usuario: las aplicaciones son más interactivas y responden antes (aparentemente).
  - Se reduce el consumo de ancho de banda. No es necesario volver a cargar toda la página, sólo las porciones relevantes.
  - No interfiere con el aspecto y el comportamiento de la página actual
- Desventajas
  - No funciona si el Javascript está deshabilitado o no se implementa. Necesario proporcionar contenido alternativo.
  - Las páginas creadas dinámicamente no se registran en el navegador. Botón "atrás" no funciona correctamente.
  - Los "robots" de internet no ejecutan normalmente Javascript. No se indexa correctamente el contenido dinámico.

Referencias y bibliografía

- Especificaciones:
  - ECMAscript es la norma, javascript y jscript (microsoft) son implementaciones
  - ECMAScript:
    - http://www.ecma-international.org/publications/files/ecma-st/ECMA-262.pdf
- Tutoriales:
  - Javascript: http://www.w3schools.com/js/default.asp
  - Jscript: http://msdn.microsoft.com

Referencias y bibliografía

- Tutoriales en español:
  - http://www.unav.es/cti/manuales/TutorialJavaScript/indices/ → Universidad de Navarra
- Utilidades de programación
  - Firebug, complemento para Firefox