Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

Aplicaciones Web

Raúl Montes T.

Actividad

- Equipos de 4 personas

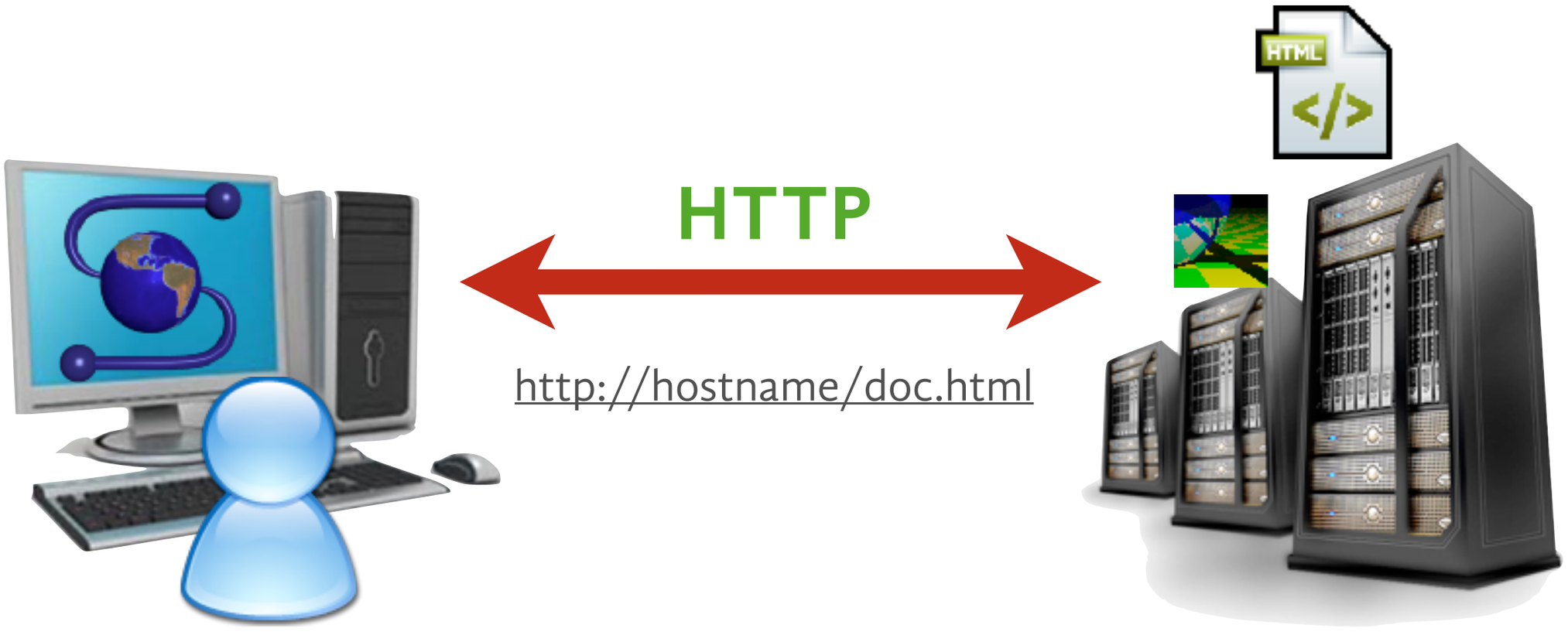
- Necesitan lápiz y papel, o un bloc de notas en computador

Para compartir documentos científicos...

Necesitamos

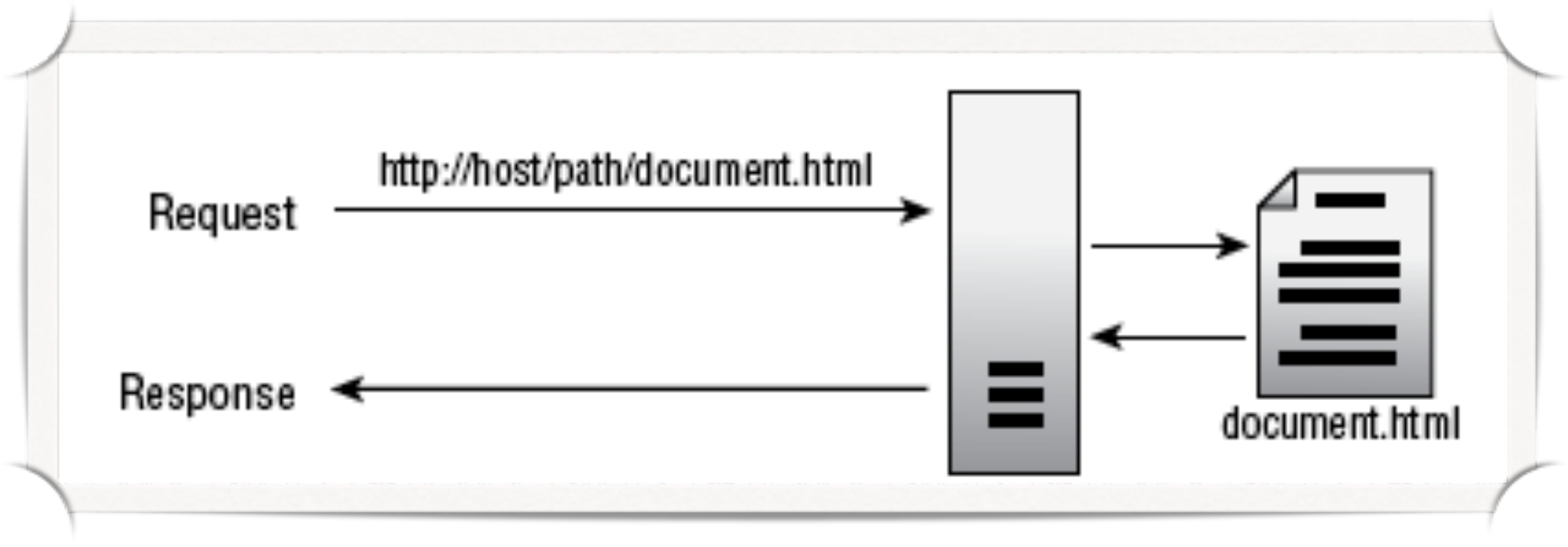
- representar el documento ➜ HTML
  - referencias a otros documentos ➜ Hyperlinks
- representar la referencia a un documento ➜ URL(I)
- servir el documento ➜ web server
- obtener el documento ➜ web browser
- que client y server puedan entenderse ➜ HTTP

Los inicios de la Web



~1990-93

El Rol de HTTP y HTML



Principales Requests en Hypertext Transfer Protocol

GET - Busca y entrega el recurso asociado al URI

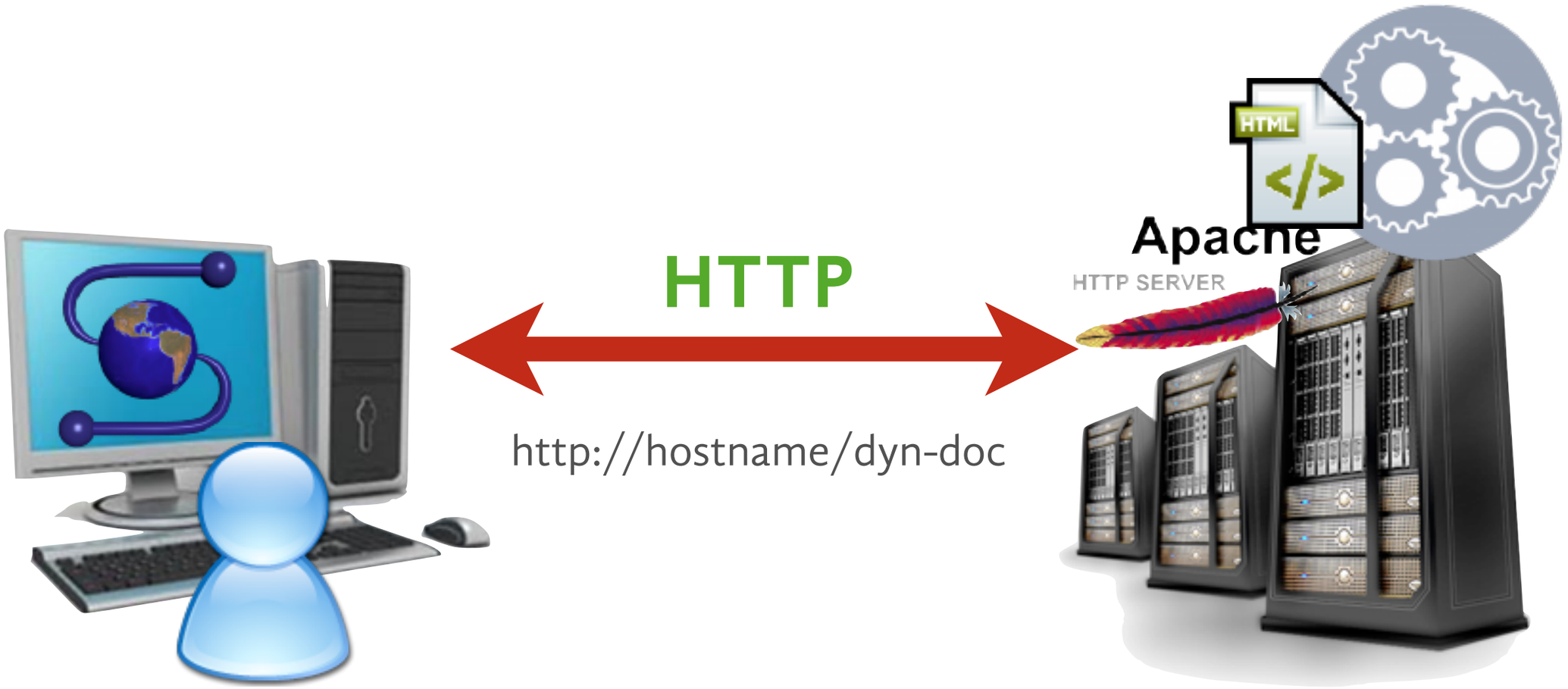
POST - Acepta la entidad adjunta en el URI indicado

PUT - Acepta la entidad adjunta y cambia el recurso

DELETE - Elimina el recurso asociado al URI

Ooooohhhmmm... be the browseeeeeeer...

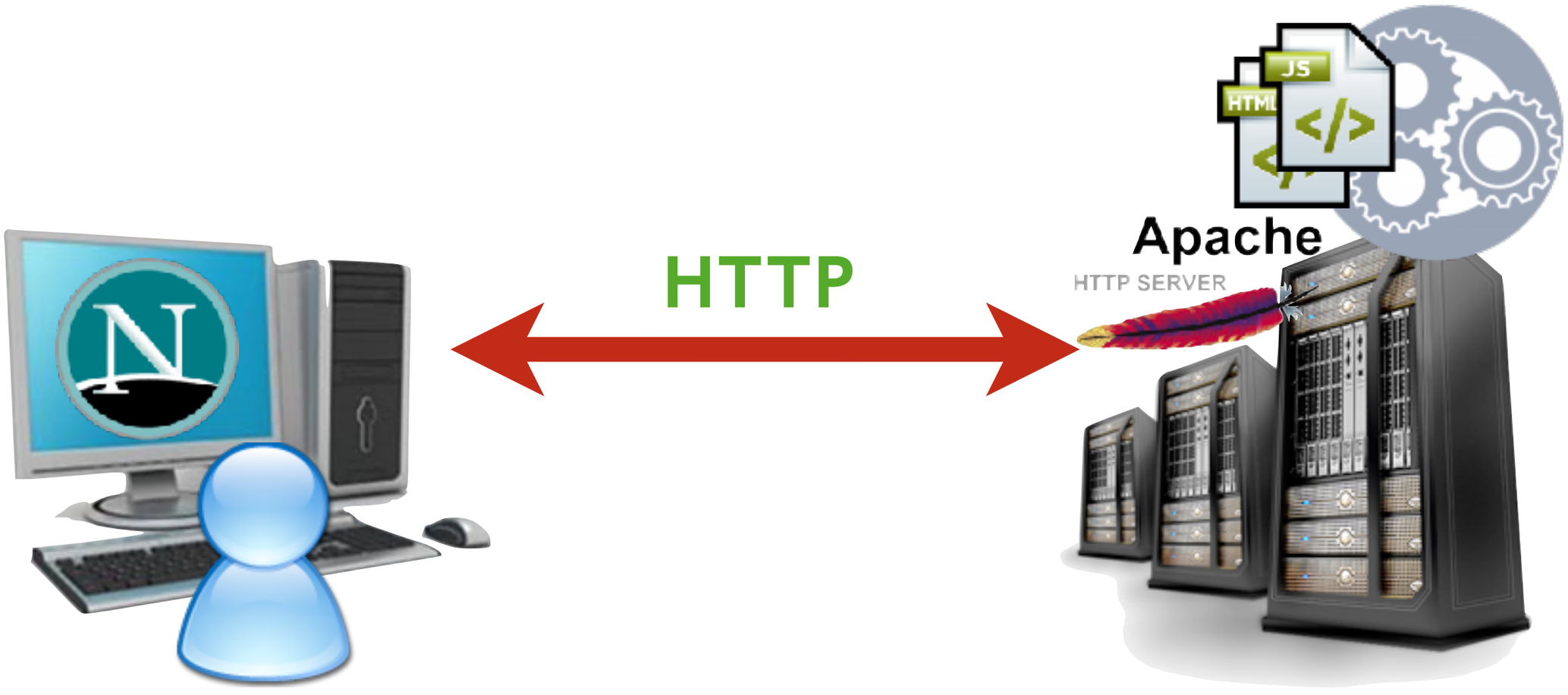
Los inicios de la Web



Necesitamos contenido dinámico

~1993

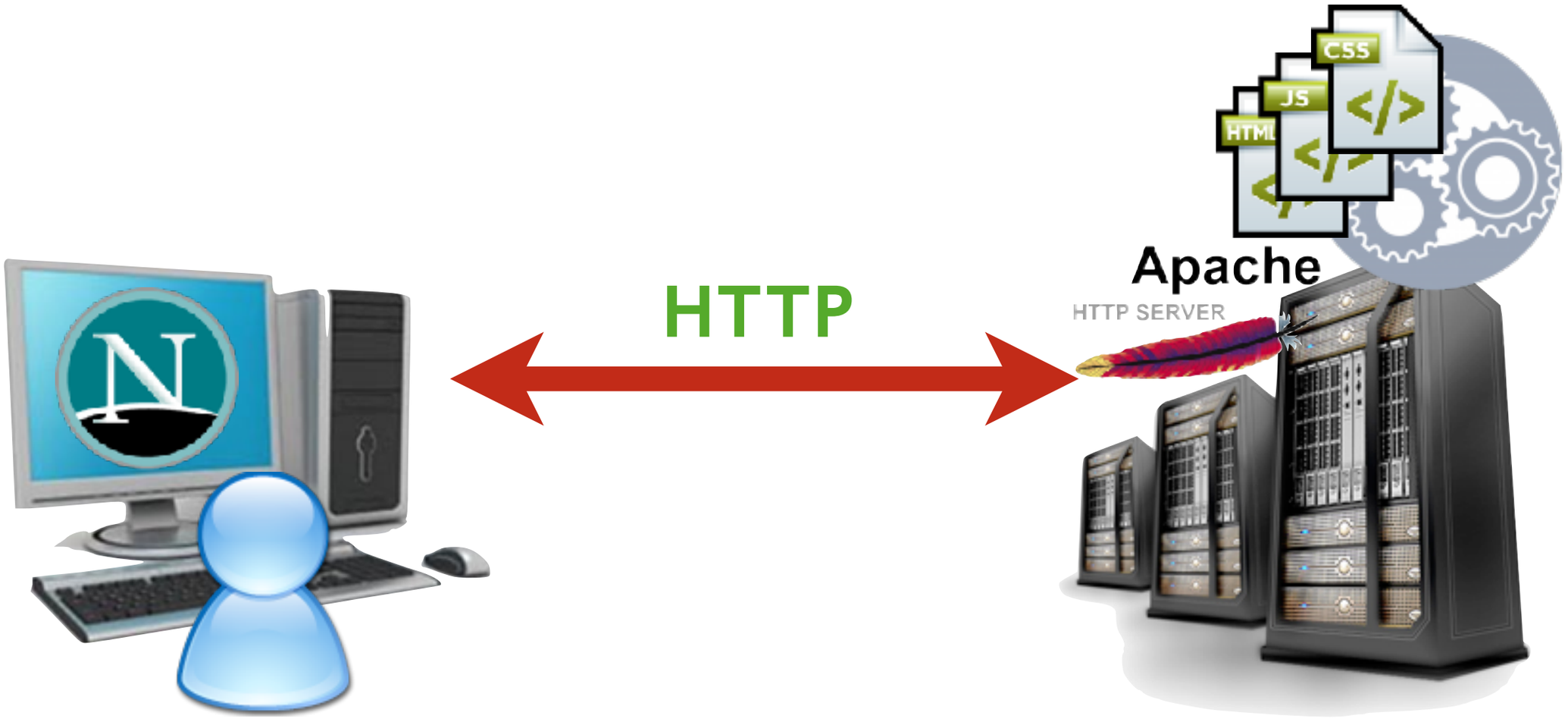
Los inicios de la Web



También en el lado del cliente

~1995

Los inicios de la Web



Necesitamos representar estilo más complejo

~1994-96

Aplicaciones Web

trabajo coordinado entre cliente y servidor

no pueden correr por si solas

HTTP

poco acceso a recursos del server

no son simplemente programas sino
verdaderos ecosistemas

~1999-2005

Aplicaciones Web

JavaScript

CSS

HTML

Front-end

HTTP

Node.js

BD

Back-end

Componentes

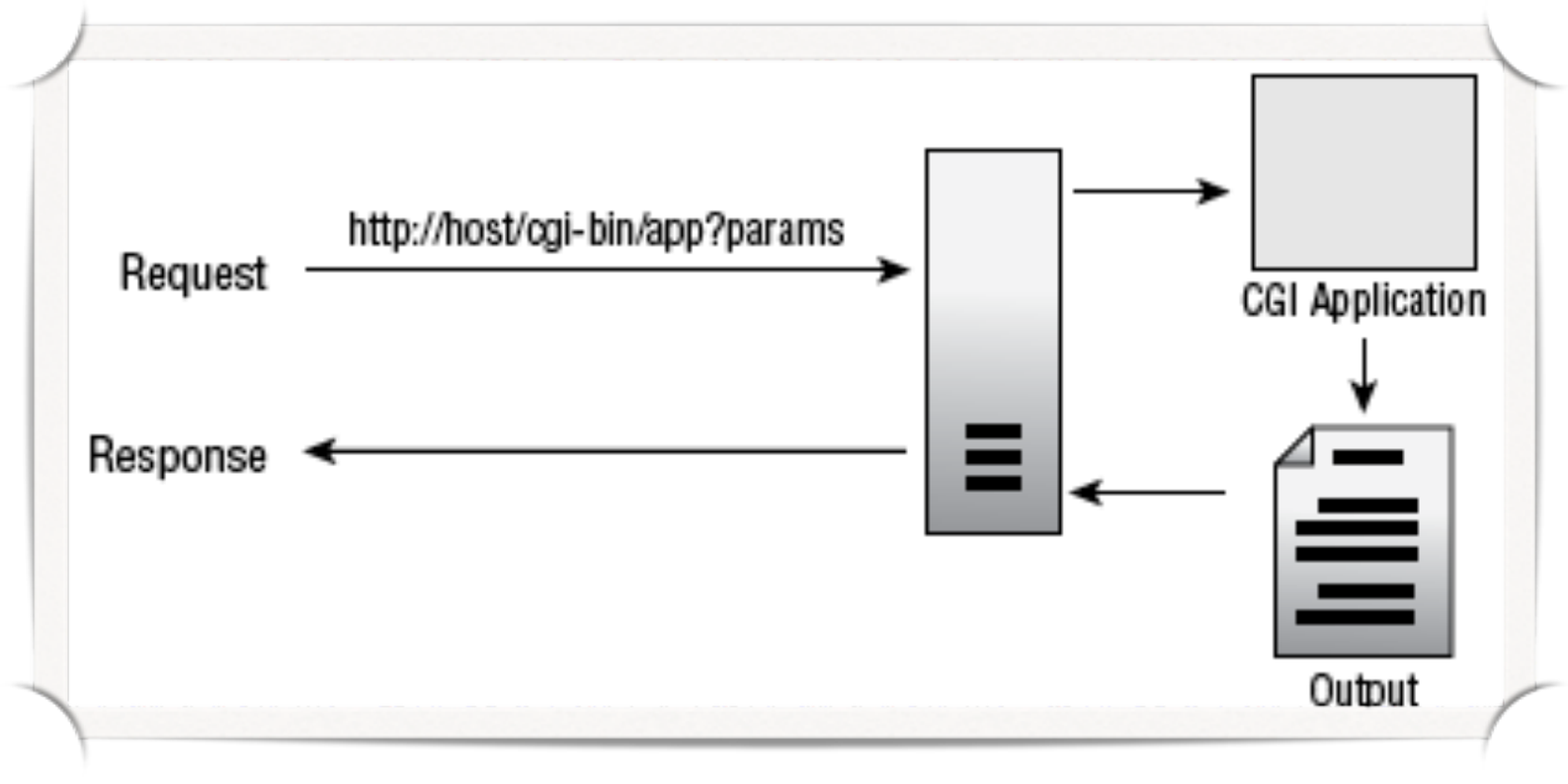
- **HTML** para estructura de la UI
- **CSS** para controlar el aspecto de la UI
- **JavaScript** para comportamiento de la UI
- **{Node.js, Ruby, ...}** para lógica de la aplicación
- motor de **BD** para persistencia de datos
- **REST API** para interactuar con otras aplicaciones

De páginas a Aplicaciones Web

- Arquitectura original de la WWW consideraba:
  - documentos codificados en HTML
  - interacción cliente servidor mediante HTTP
  - cliente capaz de desplegar contenido HTML entregado por el servidor

Aplicaciones web, los inicios

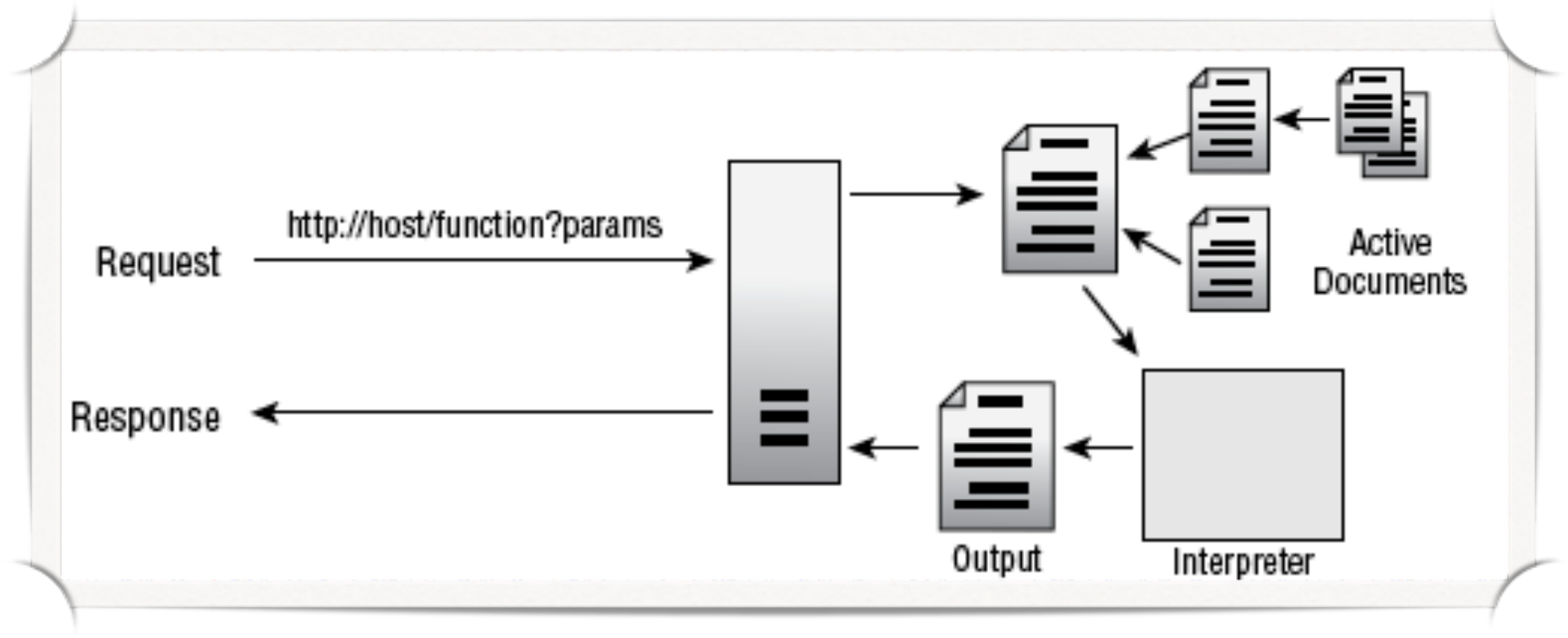
Common Gateway Interface (CGI)



Segunda Etapa: el documento activo

Ejemplos: SSI, ASP y... PHP

documento contiene código que debe ser interpretado
antes de devolverlo al cliente



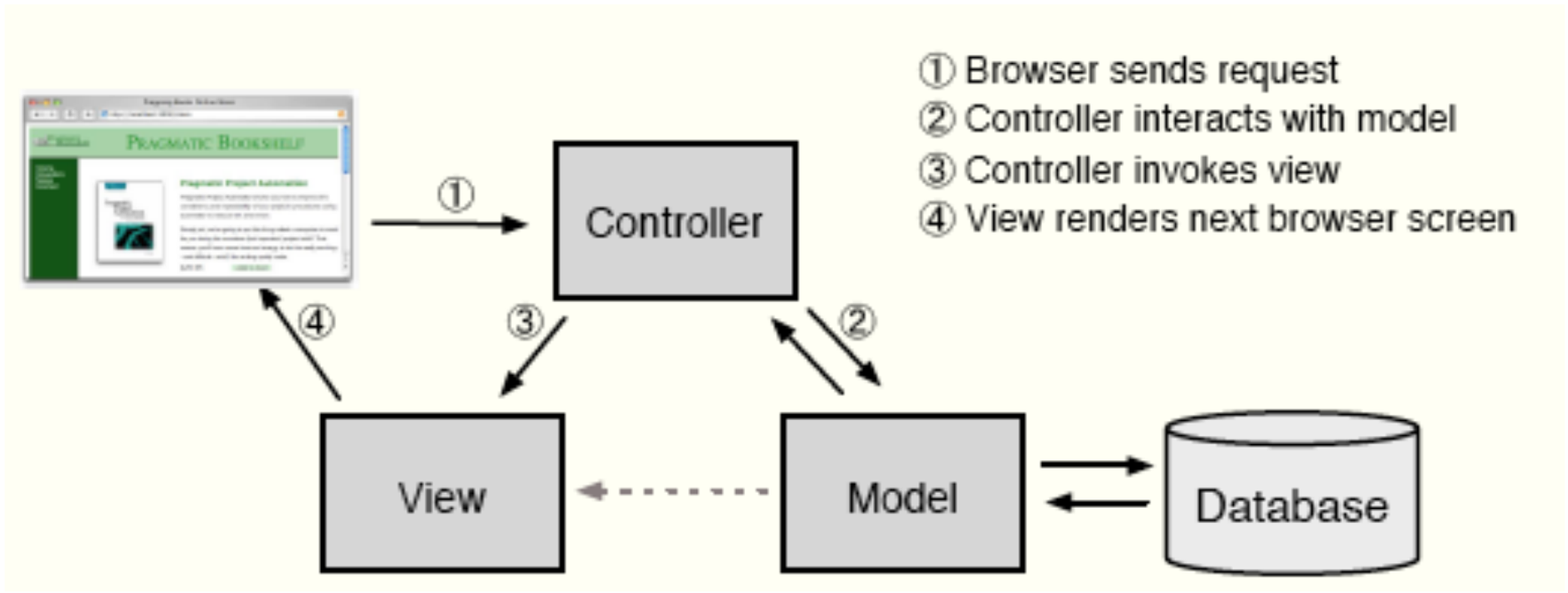
Surgimiento de Arquitectura MVC

- **Vista** - código que aplicación usa para interactuar con el usuario (HTML, CSS, JavaScript)

- **Modelo** - objetos y métodos con la lógica de la aplicación (JavaScript, Ruby, ...)

- **Controlador** - código que sirve de nexo entre vista y modelo (JavaScript, Ruby, ...)

MVC


① Browser sends request
② Controller interacts with model
③ Controller invokes view
④ View renders next browser screen
①
②
③
④
Controller
View
Model
Database

koa no es “opinionated”

- la decisión de cómo estructurar la aplicación es nuestra (no fuerza patrón MVC)

- sin embargo, usaremos el patrón MVC por decisión propia

- partiremos nuestras aplicaciones con un template especialmente hecho para el curso

Soporte del Modelo: koa-orm/sequelize

- modelo por lo general tiene una contrapartida en una BD relacional
- filas de la tabla son objetos del modelo
- atributos de objetos son columnas de la tabla
- métodos de clase para facilitar las búsquedas, ordenamientos, etc

Ejemplo

```
const User = sequelize.define('user', {
  firstName: {
    type: Sequelize.STRING
  },
  lastName: {
    type: Sequelize.STRING
  }
});

User.create({
  firstName: 'John',
  lastName: 'Hancock'
});
```

Soporte de la Vista: EJS

- Parte de la vista es HTML plano
- Hay partes dinámicas generadas a partir del controlador
- Contenido dinámico se maneja con plantillas (templates) de Embedded JavaScript

Ejemplo

```
<% if (user) { %>
  <p>Hello, <%= user.name %></p>
<% } else { %>
  <p>We don't know you</p>
<% } %>
```

y el Controlador

- el gran orquestador
- responsable del enrutamiento a la acción correspondiente
- maneja el cache, la sesión y módulos auxiliares

Hello World en koa

- `$ yarn init`
  - inicializa definición del módulo que será nuestra app
- `$ yarn add koa`
  - agrega koa como dependencia
- `$ echo "const Koa = require('koa');\n\nconst app = new Koa();\napp.use((ctx) => ctx.body = 'Hello World!');\napp.listen(3000);" > index.js`
  - programa que crea app Koa y la deja escuchando en puerto 3000
- `$ node index.js`
  - parte la aplicación