1. 服务器端处理（PHP 执行）

当你在浏览器中输入一个 PHP 页面的 URL（如 `http://example.com/page.php`）时：

1. 请求发送到服务器：浏览器向服务器发送 HTTP 请求。
2. 服务器执行 PHP 代码
   ：服务器（如 Apache/Nginx）调用 PHP 解释器，执行

   ```
   page.php
   ```

   文件中的代码。
   - PHP 代码可能包含数据库查询、逻辑判断、动态内容生成等操作。
3. 生成静态内容
   ：PHP 代码运行后，最终会输出一个
   纯 HTML 字符串
   （可能包含 CSS/JS 链接）。

   ```php
   <?php
     // 服务器端执行的 PHP 代码
     $name = "World";
     echo "<html><body><h1>Hello, $name!</h1></body></html>";
   ?>
   ```
4. 返回响应：服务器将生成的 HTML 内容通过 HTTP 响应返回给浏览器。

---

2. 客户端渲染（浏览器处理）

浏览器收到服务器的响应后：

1. 解析 **HTML**：构建 DOM 树，识别页面结构（如 `<div>`、`<p>` 标签）。
2. 加载 **CSS/JS**：根据 HTML 中的 `<link>` 和 `<script>` 标签，下载并解析 CSS 和 JavaScript 文件。
3. 渲染页面：结合 HTML 和 CSS 生成渲染树，计算布局，最终绘制到屏幕上。
4. 执行 **JavaScript**：运行 JS 代码，处理交互逻辑（如点击事件、动态加载内容）。

关键点总结

阶段	执行位置	输入	输出
服务器端	PHP 解释器	PHP 代码 + 数据库	静态 HTML/CSS/JS
客户端	浏览器引擎	HTML/CSS/JS	可视化页面 + 交互

常见误解澄清

- **PHP** 对浏览器透明：浏览器收到的只有最终的 HTML，看不到原始的 PHP 代码。（尝试在浏览器中右键“查看网页源代码”，你会发现 PHP 代码已完全消失，只留下它生成的 HTML。）
- 动态内容的本质：PHP 的“动态性”体现在服务器端根据条件生成不同的 HTML，但浏览器始终只处理静态内容。
- 前后端分离架构：现代框架（如 React/Vue）进一步将前端与后端解耦，后端仅通过 API 提供数据，前端通过 JavaScript 动态渲染页面。

示例：用户登录流程

1. 用户在浏览器输入账号密码，点击“登录”。
2. 浏览器将表单数据发送到服务器的 PHP 脚本（如 `login.php`）。
3. PHP 脚本验证账号密码，若正确则生成 HTML 响应：“欢迎，用户A！”；若错误则生成：“密码错误”。
4. 浏览器收到 HTML 后，渲染出对应的提示页面。