

Javascript入门

1.前端三板斧

- html => 简单易学，掌握常用的标签即可
- CSS => 属性规则较多，多做练习和项目
- javascript => 上手容易，但是精通很难

2.Javascript的重要性

根基

前端发展快速，开发模式、框架非常丰富 更新迭代迅速

但是不管学习的是Vue、React、Angular，包括jQuery，以及一些新出的框架。

他们本身都是基于JavaScript的，使用他们的过程中你都必须好好掌握JavaScript。

所以JavaScript是前端平地起高楼的根基

不可替代和广泛性

在工作中无论使用什么样的技术，比如Vue、React、Angular、uniapp、ReactNative、flutter

也无论做什么平台的应用程序，比如pc web、移动端web、小程序、公众号、移动端App。

都离不开JavaScript，并且深入掌握JavaScript不仅可以提高我们的开发效率，也可以帮助我们快速解决在开发中遇到

的各种问题 面试中考察面试者的JavaScript功底比重最大 属于每一个前端的本命语言和初恋

前端未来

学习前端开面对这些不断更新的技术，内心会有很多的焦虑，但是其实只要深入掌握了JavaScript就能以一敌百（例如之前提到的typescript）

- 因为JavaScript本身长期是没有对变量、函数参数等类型进行限制的 项目会有安全的隐患
- 它JavaScript提供类型检查，而不是取代JavaScript
- 如果某一天JavaScript有了类型检查和约束 ts也会自然的不存在

3.编程语言中的Js

编程语言的分类

- JavaScript是一门高级的编程语言

机器语言：1000100111011000，一些机器指令；

汇编语言：mov ax,bx，一些汇编指令；

高级语言：C、C++、Java、JavaScript、Python；

高级语言又可按照不同角度划分

一、按编程范式划分（核心思想不同）

1. 过程式语言 (Procedural Programming)

- **核心思想**：以“过程”为中心，通过函数或过程一步步执行指令。
- **特点**：强调控制流程（顺序、选择、循环）。
- **代表语言**：C、Pascal、BASIC
- **教学建议**：适合初学者学习编程基本结构和算法。

2. 面向对象语言 (Object-Oriented Programming, OOP)

- **核心思想**：一切皆对象，通过类与对象封装数据与操作。
- **三大特性**：封装、继承、多态。
- **代表语言**：Java、C++、Python、C#、Objective-C
- **教学建议**：适合引导学生从过程转向结构化设计思维。

3. 函数式语言 (Functional Programming)

- **核心思想**：函数是“第一类公民”，强调“不可变”和“无副作用”。
- **特点**：使用高阶函数、递归，避免状态改变。
- **代表语言**：Haskell、Lisp、Scala、Elm、JavaScript（部分）
- **教学建议**：进阶阶段引导学生理解纯函数、高阶函数思想。

4. 逻辑式语言 (Logic Programming)

- **核心思想**：描述“事实与规则”，程序是推理过程。
- **特点**：通过逻辑规则推导结论，不强调执行过程。
- **代表语言**：Prolog
- **教学建议**：适合人工智能推理、问题求解等高级教学内容。

二、按语言用途划分（实际应用方向）

1. 通用语言（General-purpose）

- **特点**：广泛应用于各类软件开发，功能全面。
- **代表**：Python、Java、C++、Go

2. 脚本语言（Scripting Language）

- **特点**：用于编写短小程序，快速开发与自动化。
- **代表**：Python、JavaScript、Shell、Ruby、PHP
- **用途**：自动化任务、网页交互、运维脚本等。

3. 系统编程语言（System Programming）

- **特点**：控制硬件，开发操作系统和驱动。
- **代表**：C、C++、Rust、Assembly
- **用途**：系统内核、嵌入式设备。

4. 数据科学与人工智能语言

- **代表**：Python（首选）、R、Julia
- **用途**：数据分析、机器学习、图像处理、模型训练。

5. 网页前端语言

- **HTML/CSS**：负责内容和样式（严格说不是编程语言，但必学）
- **JavaScript/TypeScript**：控制网页逻辑和交互
- **教学建议**：从静态网页过渡到动态交互，引导项目开发。

6. 网页后端语言

- **代表**：PHP、Python、Java、Node.js、Go
- **教学建议**：结合数据库、网络知识教学完整 Web 开发流程。

三、按执行方式划分（语言运行方式）

1. 编译型语言（Compiled Language）

- **特点**：先编译成机器码再运行，速度快。
- **代表**：C、C++、Go、Rust
- **教学建议**：重视语法严谨性和效率分析。















2. 解释型语言（Interpreted Language）

- **特点**：逐行解释执行，开发快、调试方便。
- **代表**：Python、JavaScript、Ruby、PHP
- **教学建议**：适合教学入门和快速验证想法。

3. 混合型语言（虚拟机执行）

- **代表**：Java、C#（通过 JVM/CLR 运行）
- **特点**：跨平台、运行效率与开发便捷性兼顾。

Js的热度

Jun 2025	Jun 2024	Change	Programming Language		Ratings (评级)	Change
1	1			Python	25.87%	+10.48%
2	2			C++	10.68%	+0.65%
3	3			C	9.47%	+0.24%
4	4			Java	8.84%	+0.44%
5	5			C#	4.69%	-1.96%
6	6			JavaScript	3.21%	-0.11%
7	7			Go	2.28%	+0.35%
8	9	^		Visual Basic	2.20%	+0.54%
9	11	^		Delphi/Object Pascal	2.15%	+0.62%
10	10			Fortran	1.86%	+0.33%
11	25	^		Ada	1.70%	+0.91%
12	8	v		SQL	1.55%	-0.21%
13	27	^		Perl	1.47%	+0.77%
14	21	^		R	1.39%	+0.43%

4.Js来时的路

Javascript&浏览器&Nodejs

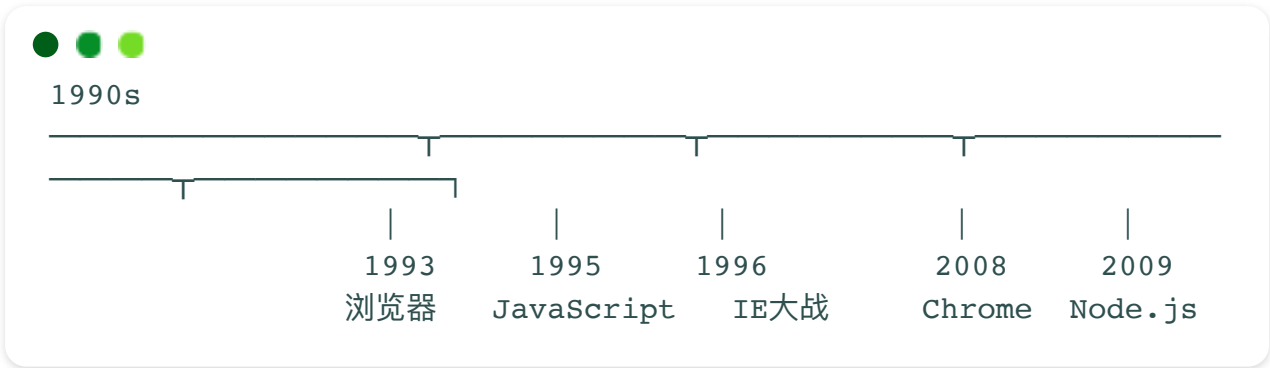
好的，Jett。下面是一个 **详细且系统的讲解材料**，可以用于你给学生上 JS、浏览器和 Node.js 的课时讲述这三者的发展历史和关联。内容围绕以下三个核心模块展开：

一、JavaScript 的诞生与发展

- 当然可以！以下是 **浏览器、JavaScript 和 Node.js 的发展史**，我将三者 **分开讲解**，再通过时间轴串联起来，确保你在授课时逻辑清晰、脉络分明。



一、整体时间轴概览（关键年份标注）



二、浏览器的发展史（Browser）



1. 初代浏览器（1990–1995）

时间	浏览器	说明
1990	WorldWideWeb（Tim Berners-Lee）	第一个图形网页浏览器，仅限 NeXT 计算机使用。
1993	Mosaic （NCSA）	第一个大众化图形浏览器，引爆 Web 革命。
1994	Netscape Navigator	创立网景公司，开启浏览器商业化时代。



2. 浏览器大战（1995–2001）

时间	浏览器	说明
1995	Internet Explorer 1.0	微软发布，搭载在 Windows 上迅速扩张。
1996-2001	Netscape vs IE	微软利用 Windows 绑定 IE，打败 Netscape。
2001	Netscape 退场，Mozilla Firefox 项目诞生。	

3. 新时代与多核浏览器（2003–至今）

时间	浏览器	说明
2003	Safari（苹果）	macOS/iOS 默认浏览器，基于 WebKit。
2004	Firefox	Mozilla 的新浏览器，更快更安全。
2008	Chrome	Google 推出，使用 V8 引擎，掀起性能革命。
2015	Microsoft Edge	Windows 新浏览器，后转为基于 Chromium。



三、JavaScript 的发展史（JS）



1. 诞生与早期（1995–1999）

时间	事件	说明
1995	Netscape 内部开发 JavaScript	Brendan Eich 在 10 天内完成原型，最初叫 Mocha。后改名为 LiveScript，最终叫 JavaScript。
1996	IE 开始支持 JScript	微软仿制 JavaScript，命名为 JScript，兼容性混乱。
1997	ECMAScript 标准发布 (ES1)	ECMA-262 发布，规范 JS。

2. 规范演进与停滞（2000–2008）

时间	事件	说明
2000–2005	ES3 占主导地位	多数浏览器支持 ES3，期间发展缓慢。
2007	ES4 计划失败	因复杂性被废弃，推动“轻量级”改进版本。

3. 现代 JavaScript 爆发（2009–至今）

时间	标准/事件	说明
2009	ES5 发布	严格模式、JSON 支持、Array 新方法。
2015	ES6 (ES2015) 大更新	<code>let/const</code> 、箭头函数、类、模块、Promise。
之后	每年发布新标准 (ES7、ES8...)	Async/Await、Optional Chaining 等现代特性。

四、Node.js 的发展史

1. 起源 (2009)

时间	事件	说明
2009	Ryan Dahl 发布 Node.js	解决传统服务器无法处理高并发的問題。使用 Chrome V8 引擎。

2. 成熟期 (2010–2014)

时间	事件	说明
2011	NPM 成为默认包管理器	Node.js 引入模块系统，生态快速发展。
2014	io.js 分支事件	社区对发展缓慢不满，分支出 io.js。

3. 合并与稳定（2015–至今）

时间	事件	说明
2015	Node.js 与 io.js 合并	成立 Node.js Foundation，走向标准化。
之后	Node.js LTS（长期支持）版本制度	每年发布新版本，支持 async/await、ESM 模块等。

五、三者关系总结（适合课堂讲解）

- 浏览器是 JavaScript 的 **起源地**。
- JavaScript 最早是为浏览器交互设计的语言。
- Node.js 是让 JS 能在 **服务器端运行** 的突破点，扩展了 JS 的用途。
- 三者是一个生态的三个阶段：
 - 浏览器带来了网页；
 - JavaScript 让网页动起来；
 - Node.js 让 JS 走出浏览器，成为全栈开发语言。

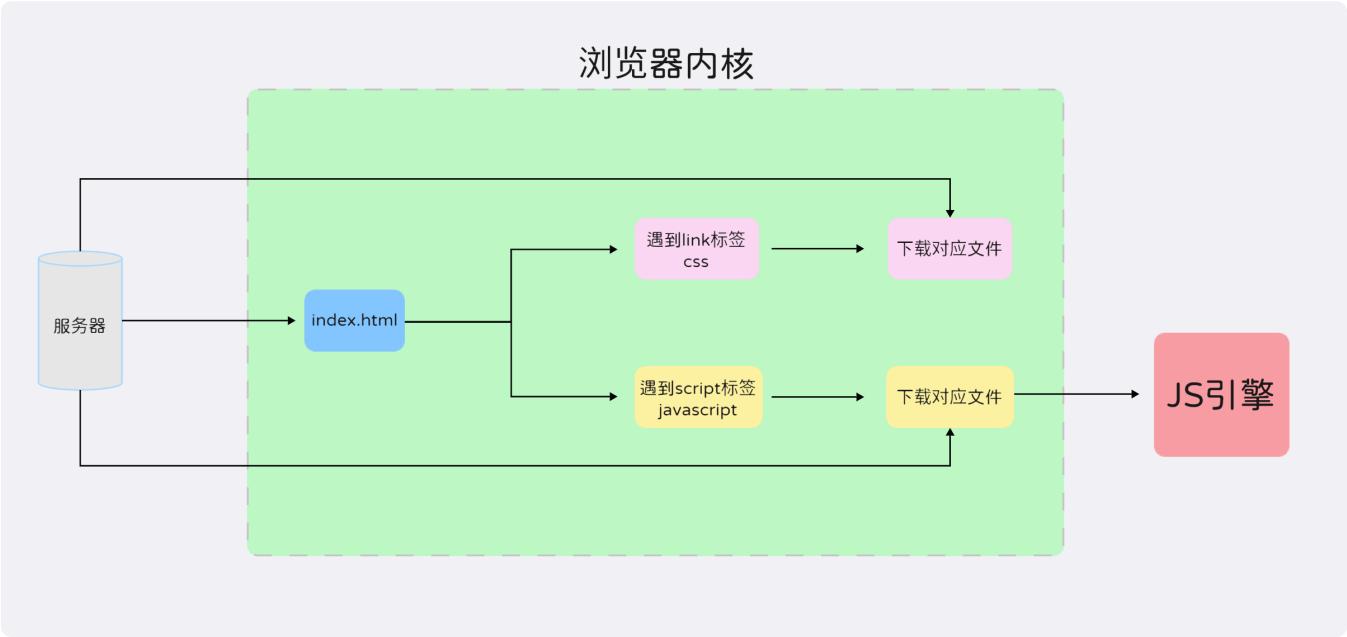
项目	JavaScript	浏览器	Node.js
角色	编程语言（脚本语言）	执行环境（JS 的原生舞台）	JS 的服务端执行平台
运行位置	浏览器 / Node.js 环境	本地客户端	服务器
能力范围	基础语法、DOM 操作等	提供 DOM、BOM、事件机制等	提供文件系统、网络、数据库等 API
引擎	由浏览器或 Node 提供	内置 JS 引擎（如 V8）	基于 V8，引入更多 Node API

关键时间轴总结

年份	事件
1995	JavaScript（Mocha）诞生于 Netscape
1996	微软推出 JScript，浏览器大战加剧
1997	ECMAScript 1.0 标准发布
2005	Ajax 技术兴起，JS 角色提升
2008	Chrome 发布 + V8 引擎登场
2009	Node.js 正式发布
2015	ECMAScript 6（ES6）发布，JS 转向现代语言
至今	JS 成为前后端通用语言，生态持续扩展

5.浏览器工作原理

浏览器的解析过程



浏览器内核（排版引擎）

Dom树

```
<!DOCTYPE html>
<html lang="en">
```

```

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>DOM Tree</title>
</head>
<body>
  <header>
    <h1>Welcome to DOM Tree</h1>
  </header>
  <section>
    <p>DOM tree</p>
    <ul>
      <li>Item 1</li>
      <li>Item 2</li>
      <li>Item 3</li>
    </ul>
  </section>
  <footer>
    <p>DOM End</p>
  </footer>
</body>
</html>

```



Document

