# *بررسی مزایا، معایب و علل عدم محبوبیت وب کامپوننت‌ها در توسعه وب*

**نویسنده: علیرضا محبی[[1]](#footnote-1)**

ایمیل: [aasoftmohebbi@gmail.com](mailto:aasoftmohebbi@gmail.com)

**عنوان: بررسی مزایا، معایب و علل عدم محبوبیت وب کامپوننت‌ها در توسعه وب**

**چکیده:**

وب کامپوننت‌ها [[2]](#footnote-2)یکی از استانداردهای مهم وب هستند که قول ایجاد انقلابی در توسعه رابط‌های کاربری را می‌دادند. این فناوری که توسط گوگل معرفی شد و امکان ساخت اجزای قابل استفاده مجدد و مستقل را برای توسعه‌دهندگان فراهم می‌کند. با این حال، با گذشت بیش از یک دهه از معرفی وب کامپوننت‌ها، این فناوری هنوز نتوانسته جایگاه مورد انتظار را در جامعه توسعه وب پیدا کند.

این مقاله به بررسی مزایا و چالش‌های استفاده از وب کامپوننت‌ها می‌پردازد. با مطالعه این مقاله، خوانندگان با استانداردهای مرتبط با وب کامپوننت‌ آشنا شده و درک عمیق‌تری از دلایل عدم محبوبیت این اجزای سفارشی در مقایسه با فریم‌ورک‌های [[3]](#footnote-3)مدرن به دست می‌آورند. همچنین راهکارهای موجود برای غلبه بر محدودیت‌های این فناوری مورد بحث قرار خواهد گرفت. این مقاله با روش علمی-پژوهشی انجام شده است.

کلمات کلیدی: وب کامپوننت‌، توسعه وب، مزایا و معایب وب کامپوننت‌ها، طراحی رابط کاربری، مدل شیء سند سایه ای و کپسوله‌سازی، فریمورک های جاوا اسکریپت

## مقدمه

## در دنیای توسعه وب، نیاز به ابزارها و فناوری‌هایی که بتوانند طراحی و پیاده‌سازی رابط‌های کاربری را ساده‌تر و کارآمدتر کنند، همواره وجود داشته است. یکی از راه‌حل‌هایی که در دهه اخیر به عنوان یک نوآوری بومی و استاندارد معرفی شد، وب کامپوننت‌ها بودند. این فناوری با وعده‌هایی چون کپسوله‌سازی کامل، قابلیت استفاده مجدد، و استقلال از فریم‌ورک‌ها وارد صحنه شد.

## با این حال، با وجود پیشرفت‌های چشمگیر در استانداردهای وب و پشتیبانی گسترده مرورگرها، وب کامپوننت‌ها هنوز نتوانسته‌اند به محبوبیت و پذیرشی که انتظار می‌رفت دست پیدا کنند. چرا با وجود قابلیت‌های برجسته، این فناوری هنوز از سوی بسیاری از توسعه‌دهندگان نادیده گرفته می‌شود؟ آیا چالش‌های فنی و محدودیت‌های اکوسیستم دلیل این عدم موفقیت است، یا فریم‌ورک‌های مدرن جایگزین بهتری ارائه داده‌اند؟

## در این مقاله، با نگاهی دقیق به مزایا، معایب، و موانع پذیرش وب کامپوننت‌ها، تلاش می‌کنیم پاسخی برای این سؤالات بیابیم و تصویری روشن از جایگاه این فناوری در توسعه وب ارائه دهیم.

## تاریخچه و مفاهیم اصلی وب

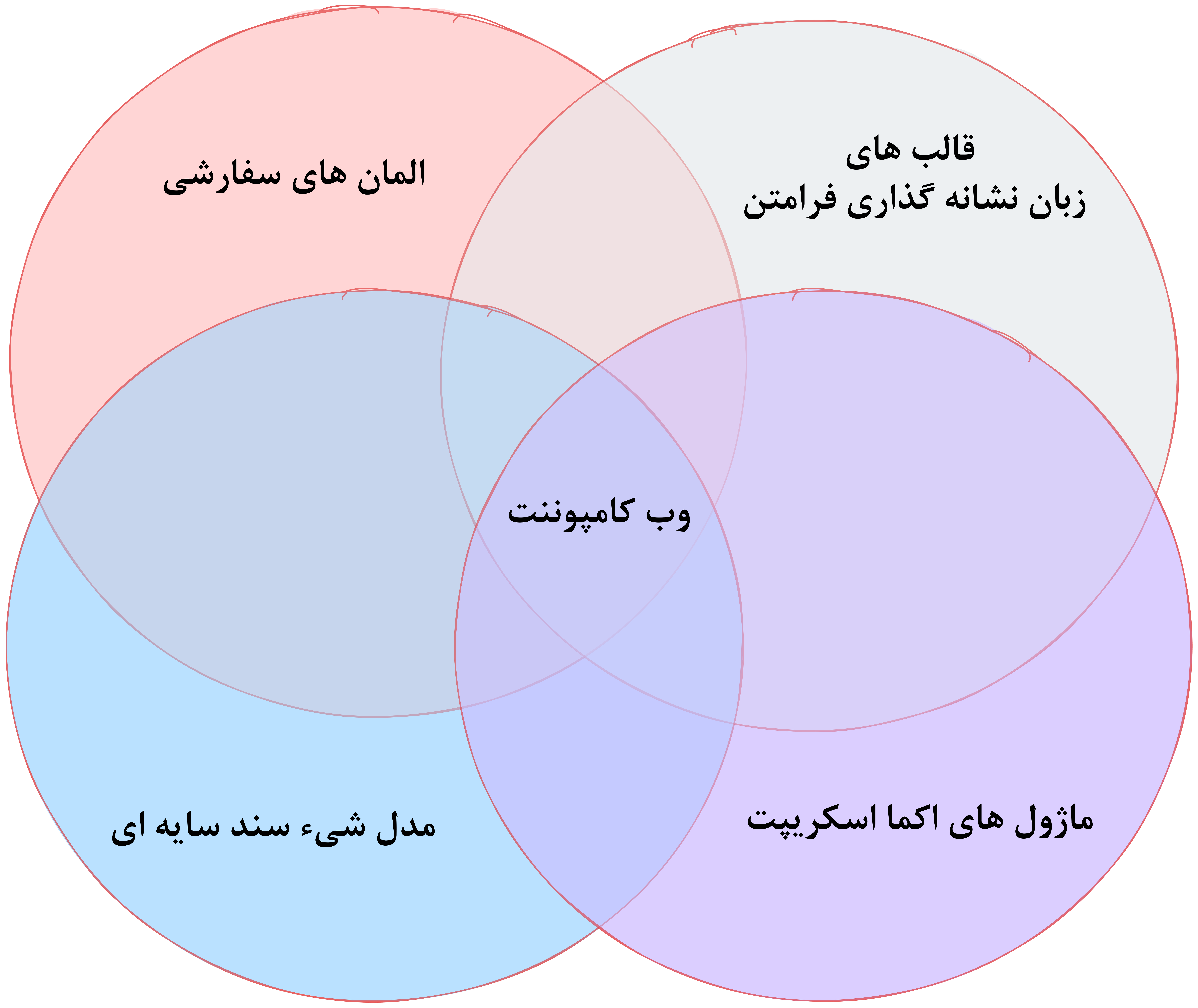
ریشه‌های تاریخی وب امروزی به دهه ۱۹۹۰ برمی‌گردد، زمانی که تیم برنرز لی زبان نشانه گذاری فرامتن [[4]](#footnote-4)را در سازمان اروپایی تحقیقات هسته ای توسعه داد . در آن زمان، وب عمدتاً از صفحات استاتیک تشکیل شده بود که با ابر پیوند[[5]](#footnote-5)‌ ها به هم متصل می‌شدند.

### پیدایش و تکامل وب کامپوننت‌

با گذشت زمان و پیشرفت فناوری‌های وب، نیاز به کامپوننت‌های قابل استفاده مجدد و مستقل افزایش یافت. وب کامپوننت‌ها به عنوان مجموعه‌ای از واسط برنامه‌نویسی نرم‌افزار کاربردی های [[6]](#footnote-6) استاندارد معرفی شدند که به توسعه‌دهندگان اجازه می‌دهند المان‌های سفارشی [[7]](#footnote-7)زبان نشانه گذاری فرامتن را با قابلیت استفاده مجدد ایجاد کنند.

### اجزای اصلی وب کامپوننت‌ها

وب کامپوننت‌ از سه تکنولوژی اصلی تشکیل شده‌است:



* المان سفارشی: واسط برنامه‌نویسی نرم‌افزار کاربردی های جاوااسکریپتی[[8]](#footnote-8) که امکان تعریف المان‌های زبان نشانه گذاری فرامتن سفارشی را فراهم می‌کنند.
* مدل شیء سند سایه ای[[9]](#footnote-9): فراهم‌کننده کپسوله‌سازی ساختار، استایل و رفتار کامپوننت
* الگوی[[10]](#footnote-10) زبان نشانه گذاری فرامتن: امکان تعریف قالب‌های قابل استفاده مجدد برای کامپوننت‌ها

مدل شیء سند سایه ای به عنوان یک نسخه کپسوله شده از مدل شیء سند عمل می‌کند که امکان مخفی کردن جزئیات پیاده‌سازی را فراهم می‌کند. این ویژگی باعث می‌شود استایل‌ها و انتخابگرها[[11]](#footnote-11) از مدل شیء سند اصلی به داخل و برعکس نشت نکنند.

### یک مثال ساده

کد بخش زبان نشانه گذاری فرامتن

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Simple Web Component</title>

</head>

<body>

<!-- استفاده از وب کامپوننت -->

<custom-button></custom-button>

<script src=”app.js”></script>

</body>

</html>

کد مربوط به بهش جاوا اسکریپت و تعریف وب کامپوننت

// تعریف کلاس برای المان سفارشی

class CustomButton extends HTMLElement {

constructor() {

super(); // فراخوانی سازنده والد

// ایجاد مدل شیء سند سایه ای

const shadow = this.attachShadow({ mode: 'open' });

// ایجاد دکمه

const button = document.createElement('button');

button.textContent = 'کلیک کنید!';

button.style.cssText = `

padding: 10px 20px;

background-color: #6200ea;

color: white;

border: none;

border-radius: 5px;

cursor: pointer;

`;

// افزودن رفتار به دکمه

button.addEventListener('click', () => {

alert('وب کامپوننت کار می‌کند!');

});

// افزودن دکمه به مدل شیء سند سایه ای

shadow.appendChild(button);

}

}

// ثبت کامپوننت سفارشی

customElements.define('custom-button', CustomButton);

**توضیح کد:**

1. **ایجاد یک کلاس جاوااسکریپت**:
   * کلاس CustomButton از HTMLElement ارث‌بری کرده و برای تعریف رفتار و ظاهر المان سفارشی استفاده می‌شود.
2. **استفاده از شیء مدل سند سایه ای**:
   * با استفاده از attachShadow، کپسوله‌سازی انجام می‌شود تا استایل‌ها و رفتار این کامپوننت از سایر بخش‌های صفحه جدا باشد.
3. **افزودن رفتار به دکمه**:
   * رویداد کلیک به دکمه اضافه شده تا تعامل‌پذیری آن نشان داده شود.
4. **تعریف و ثبت وب کامپوننت**:
   * با استفاده از customElements.define، کامپوننت با نام custom-button به مرورگر معرفی می‌شود.

این کد یک نمای ساده و عملکرد پایه وب کامپوننت‌ها را به مخاطب نشان می‌دهد.

**استاندارد های موجود**

استانداردهای وب کامپوننت‌ها به طور مداوم در حال تکامل هستند. برخی ویژگی‌ها به خوبی در استاندارد مدرن زبان نشانه گذاری فرامتن یکپارچه شده‌اند، در حالی که برخی دیگر هنوز در مرحله پیش‌نویس قرار دارند . پشتیبانی مرورگرها از این استانداردها در سال‌های اخیر به طور قابل توجهی بهبود یافته است.

مهم‌ترین ویژگی وب کامپوننت‌ها، قابلیت استفاده از آن‌ها با تقریباً هر فریم‌ورک جاوااسکریپت موجود است . این امر به دلیل استفاده از کلاس‌های اکما اسکریپت 6 [[12]](#footnote-12) است که زبان نشانه گذاری فرامتن المان بومی را گسترش می‌دهند و باعث می‌شود کامپوننت‌های سازنده در کل پروژه یا اکوسیستم شرکت قابل اشتراک‌گذاری باشند.

## مزایای استفاده از وب کامپوننت‌ها

امروزه توسعه‌دهندگان وب به دنبال راهکارهایی برای ساخت رابط‌های کاربری قدرتمند و قابل نگهداری هستند. وب کامپوننت‌ها با ارائه مجموعه‌ای از ویژگی‌های کلیدی، مزایای قابل توجهی را برای توسعه‌دهندگان فراهم می‌کنند.

### استقلال و کپسوله‌سازی کامل با مدل شیء سند سایه ای

مدل شیء سند سایه ای یکی از قدرتمندترین ویژگی‌های وب کامپوننت‌ها است که کپسوله‌سازی کامل استایل‌ها و رفتارها را امکان‌پذیر می‌سازد. این ویژگی به هر کامپوننت اجازه می‌دهد مدل شیء سند مخصوص به خود را داشته باشد که کاملاً از سند اصلی جدا شده است . استایل‌های سراسری نمی‌توانند بر آن تأثیر بگذارند و استایل‌های داخلی نیز محدود به همان کامپوننت می‌مانند.

انتقاد:

پیچیدگی در استایل‌دهی مشترک:

اگرچه مدل شیء سند سایه ای مانع نشت استایل‌ها می‌شود، اما همین ویژگی در پروژه‌هایی که نیاز به استایل‌دهی مشترک بین چند کامپوننت دارند، به چالش تبدیل می‌شود. برای مثال، تعریف یک تم سراسری یا تغییرات یکپارچه در استایل‌ها به دلیل کپسوله‌سازی مدل شیء سند سایه ای دشوارتر می‌شود. توسعه‌دهندگان مجبورند برای این کار از راه‌حل‌هایی مانند مقادیر شخصی یازی شده در سی اس اس [[13]](#footnote-13)یا ایجاد استایل در جاوا اسکریپت [[14]](#footnote-14)استفاده کنند، که پیچیدگی پروژه را افزایش می‌دهد.

عدم سازگاری با ابزارهای قدیمی:

بسیاری از ابزارهای قدیمی مانند کتابخانه‌های تست و حتی برخی از فریم‌ورک‌ها، هنوز پشتیبانی کامل و بومی از مدل شیء سند سایه ای ندارند. این موضوع ممکن است توسعه‌دهندگان را در تست یا ادغام کامپوننت‌های خود با این ابزارها محدود کند.

مشکلات در سئو و رندرینگ سمت سرور[[15]](#footnote-15):

اگرچه مدل شیء سند سایه ای برای اپلیکیشن‌های مدرن مناسب است، اما در پروژه‌هایی که به رندر سمت سرور یا بهینه‌سازی موتور جستجو نیاز دارند، چالش‌هایی ایجاد می‌کند. اطلاعات موجود در مدل شیء سند سایه ای ممکن است توسط برخی از ربات‌های خزنده قابل‌دسترسی نباشد، که این می‌تواند به کاهش رتبه سئو منجر شود.

تفاوت در مدیریت رویدادها:

رویدادهایی که در داخل مدل شیء سند سایه ای تعریف می‌شوند، به‌صورت پیش‌فرض به دامنه بیرونی منتقل نمی‌شوند. این ویژگی اگرچه باعث افزایش امنیت می‌شود، اما در برخی موارد، ارتباط بین کامپوننت‌ها را دشوارتر می‌کند و نیازمند پیاده‌سازی‌های اضافی است.

### قابلیت استفاده مجدد و ماژولار[[16]](#footnote-16)

وب کامپوننت‌ها امکان ایجاد اجزای مستقل و قابل استفاده مجدد را فراهم می‌کنند. مزایای کلیدی این رویکرد عبارتند از:

* کاهش چشمگیر زمان توسعه
* امکان توسعه مستقل بخش‌های مختلف نرم‌افزار
* قابلیت استفاده مجدد کامپوننت‌ها در پروژه‌های مختلف
* بهبود قابلیت نگهداری و توسعه نرم‌افزار

### انتقاد:

یک اکوسیستم مرکزی و استاندارد برای مدیریت و اشتراک‌گذاری وب کامپوننت‌ها، این قابلیت را در مقایسه با فریم‌ورک‌های مدرن محدود کرده است. علاوه بر این، ادغام وب کامپوننت‌ها در پروژه‌های مبتنی بر فریم‌ورک‌هایی مانند ری اکت و ویو ممکن است با چالش‌هایی در مدیریت وابستگی‌ها و هماهنگی حافظه مواجه شود. در حالی که استفاده مجدد از وب کامپوننت‌ها یک مزیت کلیدی است، تقویت ابزارها و زیرساخت‌های مرتبط می‌تواند این مزیت را برای توسعه‌دهندگان کاربردی‌تر کند.

### عملکرد بهینه و سریع

از آنجا که وب کامپوننت‌ها مستقیماً از واسط برنامه‌نویسی نرم‌افزار کاربردیهای بومی مرورگر استفاده می‌کنند، عملکرد بهینه‌تری نسبت به راهکارهای مبتنی بر فریم‌ورک دارند. این کامپوننت‌ها در واقع کلاس‌های اکما اسکریپت 6 هستند که زبان نشانه گذاری فرامتن المان بومی را گسترش می‌دهند .

**انتقاد:**

اگرچه استفاده از رابط کاربری اپلیکیشن های بومی مرورگر، سرعت اجرا را افزایش می‌دهد، اما این رویکرد ممکن است در پروژه‌های پیچیده که نیازمند مدیریت وضعیت یا تعامل‌های پیچیده هستند، کارایی محدودی داشته باشد. علاوه بر این، بهینه‌سازی عملکرد در فریم‌ورک‌هایی مانند ری اکت یا اسولت[[17]](#footnote-17) به لطف کامپایلرها و مدل شیء سند مجازی، در بسیاری از موارد به نتیجه مشابه یا حتی بهتر منجر می‌شود.

### سازگاری با تمام فریم‌ورک‌ها

یکی از برجسته‌ترین مزایای وب کامپوننت‌ها، سازگاری آن‌ها با تمامی فریم‌ورک‌های مدرن است. این قابلیت به توسعه‌دهندگان اجازه می‌دهد:

* استفاده از کامپوننت‌ها در پروژه‌های مختلف با فریم‌ورک‌های متفاوت
* ترکیب وب کامپوننت‌ها با کتابخانه‌های موجود
* حفظ انعطاف‌پذیری در انتخاب ابزارهای توسعه

**انتقاد:**

هرچند این سازگاری در تئوری مزیتی مهم است، اما در عمل، پیاده‌سازی و ادغام وب کامپوننت‌ها با فریم‌ورک‌هایی مانند ری اکت و انگولار می‌تواند پیچیده باشد. برای مثال، مدیریت حافظه یا استفاده از ویژگی‌های خاص فریم‌ورک‌ها، به دلیل رفتار مستقل وب کامپوننت‌ها، ممکن است به کدنویسی اضافی و استفاده از الگوهای خاص نیاز داشته باشد.

سازگاری وب کامپوننت‌ها با فریم‌ورک‌ها یک مزیت مهم است، اما توسعه‌دهندگان باید برای یکپارچه‌سازی روان در پروژه‌های فریم‌ورکی، چالش‌های بالقوه را در نظر بگیرند.

### سهولت در اشتراک‌گذاری و استفاده از کامپوننت‌ها بین پروژه‌ها

وب کامپوننت‌ها به دلیل استاندارد بودن، به راحتی قابل اشتراک‌گذاری هستند. توسعه‌دهندگان می‌توانند کامپوننت‌های خود را در پلتفرم‌هایی مانند وب سایت رسمی این تکنولوژی [[18]](#footnote-18) به اشتراک بگذارند .این قابلیت باعث می‌شود:

* صرفه‌جویی در زمان توسعه با استفاده از کامپوننت‌های آماده
* ایجاد اکوسیستمی غنی از کامپوننت‌های قابل استفاده مجدد
* تسهیل همکاری بین تیم‌های مختلف توسعه

**انتقاد:**

هرچند اشتراک‌گذاری کامپوننت‌ها به دلیل استاندارد بودن آن‌ها آسان به نظر می‌رسد، اما همانطور که قبلا اشاره شد نبود یک اکوسیستم جامع و متمرکز، این مزیت را محدود کرده است. اگرچه وب سایت‌هایی مانند وب سایت رسمی شان برای اشتراک‌گذاری وجود دارند، اما همچنان در مقایسه با مارکت‌های گسترده‌ای مانند وب سایت بسته های جاوا اسکریپت[[19]](#footnote-19)، تنوع و گستردگی کمتری دارند. پیش‌تر نیز به این موضوع اشاره شد که نبود چنین اکوسیستمی، مدیریت و یافتن کامپوننت‌های مناسب را چالش‌برانگیز می‌کند.

## چالش‌های فنی وب کامپوننت‌ها

علی‌رغم مزایای متعدد وب کامپوننت‌ها، چالش‌های فنی قابل توجهی در مسیر پیاده‌سازی و استفاده از آن‌ها وجود دارد که توسعه‌دهندگان باید با آن‌ها دست و پنجه نرم کنند.

### مشکلات رندر سمت سرور

رندر سمت سرور یکی از چالش‌برانگیزترین جنبه‌های استفاده از وب کامپوننت‌ها است. این کار معمولاً نیازمند شبیه‌سازی مدل شیء سند در نود جی اس[[20]](#footnote-20) و سپس سریالی‌سازی مدل شیء سند است . این فرآیند می‌تواند:

* مدت زمان پاسخ را افزایش دهد
* حجم فایل‌های پاسخ را بیشتر کند
* پیچیدگی برنامه را به طور قابل توجهی افزایش دهد

**انتقاد**

مشکلات رندر سمت سرور در وب کامپوننت‌ها بیشتر در پروژه‌هایی که نیاز شدید به سئو دارند، اهمیت پیدا می‌کند، اما این چالش‌ها در همه موارد کاربرد ندارد. برای مثال، در یک پنل مدیریتی داخلی، که سئو اهمیتی ندارد، رندر سمت سرور نیازی نیست و پیچیدگی اضافه‌ای ایجاد نمی‌شود.علاوه بر این، ابزارهایی مانند لیت[[21]](#footnote-21) رندر سمت سرور و تکنیک‌های پیش رندر [[22]](#footnote-22)می‌توانند مشکلات رندر سمت سرور را کاهش دهند. از طرفی، حتی فریم‌ورک‌های مدرنی مثل ری اکت در هنگام مدیریت تعاملات پس از رندر سمت سرور، با چالش‌های مشابهی مواجه هستند. بنابراین، این مسئله به خودی خود یک ضعف منحصربه‌فرد برای وب کامپوننت‌ها محسوب نمی‌شود.

### پیچیدگی‌های مدل شیء سند سایه ای

مدل شیء سند سایه ای، علی‌رغم مزایای کپسوله‌سازی، چالش‌های خاص خود را دارد. هر المان در صفحه فقط می‌تواند یک سایه داشته باشد. پیچیدگی‌های اصلی عبارتند از:

* دسترسی به مدل شیء سند سایه ای به طور کامل از سند اصلی محدود شده است
* رویدادها در داخل و خارج از محدوده سایه به شکل متفاوتی رفتار می‌کنند
* برخی از ابزارهای شخص ثالث با کامپوننت‌های دارای مدل شیء سند سایه ای سازگار نیستند

**انتقاد**

مدل شیء سند سایه ای به دلیل محدودیت دسترسی از سند اصلی و رفتار متفاوت رویدادها، ممکن است چالش‌هایی ایجاد کند. با این حال، این محدودیت‌ها بیشتر برای حفظ کپسوله‌سازی طراحی شده‌اند. مثلاً با استفاده از ویژگی composed: true در رویدادهای سفارشی یا رابط کاربری اپلیکیشن shadowRoot، می‌توان بسیاری از این چالش‌ها را مدیریت کرد. مشکل سازگاری ابزارهای شخص ثالث نیز به مرور زمان با پذیرش گسترده‌تر استانداردها کاهش یافته است.

const button = document.querySelector('custom-button').shadowRoot.querySelector('button');

button.textContent = "Updated!";

### در این مثال، shadowRoot برای دسترسی به محتوای داخل که مدل شیء سند سایه ای استفاده می‌شود، که به ما اجازه می‌دهد متن دکمه را تغییر دهیم. این نشان می‌دهد که اگرچه که مدل شیء سند سایه ای دسترسی مستقیم را محدود می‌کند، اما با استفاده از رابط کاربری اپلیکیشن مناسب، می‌توان نیازهای خاص را مدیریت کرد.

### در حالی که مدل شیء سند سایه ای پیچیدگی‌هایی دارد، این موارد عمدتاً قابل حل هستند و هدف اصلی آن‌ها حفظ کپسوله‌سازی و جلوگیری از تداخل است. با تکنیک‌های مناسب، می‌توان از مزایای آن بهره‌برد و چالش‌ها را مدیریت کرد.

### محدودیت‌های استایل‌دهی

استایل‌دهی در وب کامپوننت‌ها با محدودیت‌های قابل توجهی همراه است. سند استایل ها و مدل شیء سند سایه، مجزا از یکدیگر هستند. این موضوع چند چالش اساسی ایجاد می‌کند:

* قوانین استایل خارجی در مدل شیء سند سایه ای عمل نمی‌کنند .
* برای استایل‌دهی با CSS سراسری، باید از مدل شیء سند سایه ای صرف نظر کرد .
* ارسال المان‌های سفارشی با CSS محدود به حوزه[[23]](#footnote-23) مدل شیء سند سایه ای نیازمند قرار دادن CSS درون جاوا اسکریپت است.

علاوه بر این، عناصر شبه CSS مانند (:checked) با عناصر سفارشی کار نمی‌کنند. همچنین، عناصر استاندارد زبان نشانه گذاری فرامتن در صورت مرتبط بودن رفتار آن‌ها با یکدیگر، عناصر سفارشی را نادیده می‌گیرند . به عنوان مثال، عنصر فرم یک عنصر ورودی سفارشی را نادیده می‌گیرد.

**انتقاد**

یکی از انتقادات به محدودیت‌های استایل‌دهی در وب کامپوننت‌ها، استفاده الزامی از سی اس اس درون Shadow DOM است. این محدودیت باعث می‌شود تغییرات پویا در استایل یا به‌روزرسانی‌های گسترده به سختی انجام شود، به‌خصوص زمانی که پروژه نیاز به هماهنگی بصری در سطح کل سیستم دارد. با این حال، استفاده از راهکارهایی مانند ترکیب Shadow DOM با اسکوپ‌های گلوبال، می‌تواند این مشکل را به حداقل برساند.

مثال عملی:

در اینجا از یک راهکار ترکیبی استفاده شده که به کمک CSS گلوبال و Shadow DOM، انعطاف‌پذیری بیشتری ایجاد می‌کند:

<style>

.global-theme {

--primary-color: blue;

}

</style>

<custom-button class="global-theme"></custom-button>

<script>

class CustomButton extends HTMLElement {

constructor() {

super();

const shadow = this.attachShadow({ mode: 'open' });

const button = document.createElement('button');

const style = document.createElement('style');

style.textContent = `

button {

background-color: var(--primary-color, gray);

color: white;

padding: 10px 20px;

border: none;

border-radius: 5px;

cursor: pointer;

}`;

button.textContent = 'Click Me';

shadow.append(style, button);

}}

customElements.define('custom-button', CustomButton);

</script>

در این روش، از یک کلاس سراسری برای ارسال متغیرهای سی اس اس به Shadow DOM استفاده شده است. این ترکیب به ما امکان می‌دهد استایل‌های سراسری را حفظ کنیم و در عین حال، کپسوله‌سازی Shadow DOM را نیز اعمال کنیم.

## مقایسه با فریم‌ورک‌های مدرن

در دنیای توسعه وب، انتخاب ابزار مناسب برای ساخت رابط‌های کاربری اهمیت ویژه‌ای دارد. مقایسه وب کامپوننت‌ها با فریم‌ورک‌های مدرن مانند React، Vue و Angular می‌تواند دید جامعی از نقاط قوت و ضعف هر رویکرد ارائه دهد.

### تفاوت‌های معماری

معماری وب کامپوننت‌ها با فریم‌ورک‌های مدرن تفاوت‌های اساسی دارد. فریم‌ورک‌هایی مانند React از مدل شیء سند مجازی برای بهینه‌سازی عملکرد استفاده می‌کنند [[12]](https://arioit.com/best-front-end-framework/). در مقابل، وب کامپوننت‌ها مستقیماً با مدل شیء سند واقعی کار می‌کنند که این موضوع می‌تواند در برخی موارد به کارایی بهتر منجر شود.

تفاوت‌های کلیدی معماری عبارتند از:

* مدیریت وضعیت (State Management)
* چرخه حیات کامپوننت‌ها
* نحوه رندرینگ و بروزرسانی مدل شیء سند
* مدیریت رویدادها

### مقایسه تجربه توسعه‌دهنده

تجربه توسعه‌دهنده در استفاده از فریم‌ورک‌های مدرن و وب کامپوننت‌ها متفاوت است. فریم‌ورک‌هایی مانند Vue.js از سادگی و انعطاف‌پذیری بالایی برخوردار هستند [[13]](https://barnamenevisan.org/Articles/Article7674.html). این در حالی است که Angular با معماری MVC خود، ساختاری یکپارچه برای برنامه‌نویسی فرانت‌اند ارائه می‌دهد [[12]](https://arioit.com/best-front-end-framework/).

جدول مقایسه تجربه توسعه:

ویژگیوب کامپوننت‌هافریم‌ورک‌های مدرن

منحنی یادگیری

متوسط

نسبتاً پیچیده

ابزارهای توسعه

محدود

گسترده

مستندات

کمتر

جامع و به‌روز

اکوسیستم

در حال رشد

بالغ و غنی

### عملکرد و کارایی

در زمینه عملکرد، هر دو رویکرد مزایا و معایب خود را دارند. فریم‌ورک‌هایی مانند Svelte با رویکرد متفاوت خود، کامپوننت‌ها را در زمان کامپایل به کد ساده جاوا اسکریپت تبدیل می‌کنند که باعث کاهش نیاز به بارگذاری و بهبود کارایی می‌شود [[12]](https://arioit.com/best-front-end-framework/).

برای پروژه‌های کوچک و متوسط، Vue و React انتخاب‌های مناسبی هستند [[12]](https://arioit.com/best-front-end-framework/). در مقابل، برای پروژه‌های بزرگ و سازمانی، Angular و Ember می‌توانند گزینه‌های بهتری باشند [[12]](https://arioit.com/best-front-end-framework/). کتابخانه‌های کامپوننت مانند Material UI و Ant Design نیز امکان شخصی‌سازی گسترده‌ای را از طریق متغیرهای تم فراهم می‌کنند [[13]](https://barnamenevisan.org/Articles/Article7674.html).

نکته قابل توجه این است که فریم‌ورک‌های مدرن معمولاً دارای ویژگی‌های اضافی مانند مدیریت وضعیت، روتینگ و ابزارهای توسعه یکپارچه هستند [[14]](https://bugeto.net/blog/blazor-vs-react-comparison). این در حالی است که وب کامپوننت‌ها به دلیل سادگی و استفاده از واسط برنامه‌نویسی نرم‌افزار کاربردیهای استاندارد مرورگر، می‌توانند در برخی موارد عملکرد بهتری داشته باشند.

## دلایل عدم محبوبیت در جامعه توسعه‌دهندگان

علیرغم پتانسیل بالای وب کامپوننت‌ها در توسعه وب مدرن، این فناوری با چالش‌های متعددی در مسیر پذیرش گسترده مواجه شده است. بررسی دقیق این چالش‌ها می‌تواند درک عمیق‌تری از وضعیت کنونی این فناوری ارائه دهد.

### کمبود منابع یادگیری و اکوسیستم

یکی از مهم‌ترین موانع در مسیر گسترش وب کامپوننت‌ها، محدودیت منابع آموزشی و ضعف اکوسیستم است. برخلاف فریم‌ورک‌های محبوب که از منابع آموزشی غنی برخوردار هستند، وب کامپوننت‌ها با کمبود قابل توجه منابع یادگیری مواجه هستند [[15]](https://7learn.com/blog/what-is-react). این مشکل به‌ویژه برای توسعه‌دهندگان تازه‌کار چالش‌برانگیز است.

مقایسه وضعیت منابع آموزشی:

نوع منبعوب کامپوننت‌هافریم‌ورک‌های رایج

کتاب‌های تخصصی

محدود

گسترده

دوره‌های آنلاین

بسیار کم

متنوع و فراوان

مستندات فنی

ناکافی

جامع و به‌روز

نمونه پروژه‌ها

اندک

فراوان

### پیچیدگی یادگیری و استفاده

پیچیدگی‌های فنی وب کامپوننت‌ها باعث شده است که یادگیری و استفاده از آن‌ها برای توسعه‌دهندگان چالش‌برانگیز باشد [[11]](https://roocket.ir/articles/what-happened-to-web-components). این پیچیدگی‌ها شامل موارد زیر است:

* حجم ورودی بیش از حد زیاد در پیاده‌سازی
* عدم وجود راهنمای مشخص برای بهترین شیوه‌های پیاده‌سازی
* دشواری در یافتن و انتخاب کامپوننت‌های مناسب
* فقدان مستندات کافی برای کامپوننت‌های موجود

### پشتیبانی دیرهنگام مرورگرها و تکمیل استانداردها در سال ۲۰۲۰

تأخیر در پشتیبانی کامل مرورگرها از وب کامپوننت‌ها یکی از عوامل اصلی عدم محبوبیت این فناوری بوده است. تا پیش از سال ۲۰۲۰، بسیاری از مرورگرها پشتیبانی کاملی از استانداردهای وب کامپوننت‌ها نداشتند [[11]](https://roocket.ir/articles/what-happened-to-web-components). این تأخیر باعث شد توسعه‌دهندگان به سمت راه‌حل‌های جایگزین و فریم‌ورک‌های موجود روی آورند.

### ابهامات پیرامون زبان نشانه گذاری فرامتن Imports و مسیر توسعه آن

یکی از چالش‌های جدی در مسیر توسعه وب کامپوننت‌ها، ابهام در مورد زبان نشانه گذاری فرامتن Imports بوده است. حذف خصوصیات نحوه ایمپورت در زبان نشانه گذاری فرامتن باعث پیچیده‌تر شدن فرآیند توسعه شد [[11]](https://roocket.ir/articles/what-happened-to-web-components). در حال حاضر، توسعه‌دهندگان مجبور به استفاده از ماژول‌های اکما اسکریپت 6 به جای ایمپورت‌های زبان نشانه گذاری فرامتن هستند که این موضوع پیچیدگی بیشتری را به فرآیند توسعه اضافه می‌کند [[11]](https://roocket.ir/articles/what-happened-to-web-components).

نکته مهم: عدم وجود یک دایرکتوری مرکزی برای کامپوننت‌ها باعث شده است که یافتن و ارزیابی کامپوننت‌های مناسب برای پروژه‌ها دشوار باشد [[11]](https://roocket.ir/articles/what-happened-to-web-components). این مشکل زمانی پیچیده‌تر می‌شود که توسعه‌دهنده با چندین گزینه مواجه است و باید بدون داشتن مستندات کافی، مناسب‌ترین گزینه را انتخاب کند.

## نتیجه‌گیری

وب کامپوننت‌ها با وجود مزایای قابل توجه مانند کپسوله‌سازی، قابلیت استفاده مجدد و سازگاری با فریم‌ورک‌های مختلف، هنوز نتوانسته‌اند جایگاه مورد انتظار را در جامعه توسعه وب پیدا کنند. چالش‌های فنی مانند محدودیت‌های رندر سمت سرور، پیچیدگی‌های شادو دام و مشکلات استایل‌دهی، موانع اصلی پذیرش گسترده این فناوری هستند.

کمبود منابع آموزشی، تأخیر در پشتیبانی مرورگرها و نبود اکوسیستم قوی نیز عواملی هستند که محبوبیت وب کامپوننت‌ها را محدود کرده‌اند. فریم‌ورک‌های مدرن با ارائه راهکارهای جامع‌تر و ابزارهای توسعه پیشرفته‌تر، همچنان گزینه‌های مطلوب‌تری برای توسعه‌دهندگان محسوب می‌شوند.

آینده وب کامپوننت‌ها به بهبود استانداردها، افزایش منابع آموزشی و تکامل ابزارهای توسعه بستگی دارد. توسعه‌دهندگان باید با درک دقیق مزایا و محدودیت‌های این فناوری، تصمیم آگاهانه‌ای برای استفاده از آن در پروژه‌های خود بگیرند.

## سوالات متداول

متاسفانه هیچ سوالی از بخش "People Also Ask" در دسترس نیست. اگر سوالی دارید که مرتبط با موضوع مقاله باشد، لطفاً آن را مطرح کنید تا به آن پاسخ دهیم.

## مراجع

[1] -   
[2] -   
[3] -   
[4] -   
[5] -   
[6] -   
[7] -   
[8] -   
[9] -   
[10] -   
[11] -   
[12] -   
[13] -   
[14] -   
[15] -<https://raman01.ir/a-brief-history-of-the-world-wide-web/>[https://www.dntips.ir/post/3187/زبان نشانه گذاری فرامتن 5-web-component-قسمت-اول-معرفی-و-مفاهیم-اولیه](https://www.dntips.ir/post/3187/html5-web-component-%D9%82%D8%B3%D9%85%D8%AA-%D8%A7%D9%88%D9%84-%D9%85%D8%B9%D8%B1%D9%81%DB%8C-%D9%88-%D9%85%D9%81%D8%A7%D9%87%DB%8C%D9%85-%D8%A7%D9%88%D9%84%DB%8C%D9%87)[https://fa.جاوا اسکریپت.info/webcomponents-intro](https://fa.javascript.info/webcomponents-intro)<https://roocket.ir/articles/web-components-in-2021-the-good-the-bad-and-the-ugly>[https://roocket.ir/articles/scoping-css-using-سایه-مدل شیء سند](https://roocket.ir/articles/scoping-css-using-shadow-dom)[https://fa.جاوا اسکریپت.info/سایه-مدل شیء سند](https://fa.javascript.info/shadow-dom)<https://danup.ir/what-is-component-in-programming/><https://www.robotarh.com/Blog/Tutorial/Using-Blog-Posts-Components>[https://fa.vuejs.org/guide/extras/web-components.زبان نشانه گذاری فرامتن](https://fa.vuejs.org/guide/extras/web-components.html) <https://roocket.ir/articles/server-side-rendering-your-react-app-in-three-simple-steps><https://roocket.ir/articles/what-happened-to-web-components><https://arioit.com/best-front-end-framework/>[https://barnamenevisan.org/Articles/Article7674.زبان نشانه گذاری فرامتن](https://barnamenevisan.org/Articles/Article7674.html) <https://bugeto.net/blog/blazor-vs-react-comparison><https://7learn.com/blog/what-is-react>

1. .گروه مهندسی کامپیوتر، واحد قم، پیام نور، قم، ایران [↑](#footnote-ref-1)
2. . Web Components [↑](#footnote-ref-2)
3. . Framework [↑](#footnote-ref-3)
4. . HTML [↑](#footnote-ref-4)
5. . Hyper Link [↑](#footnote-ref-5)
6. . API [↑](#footnote-ref-6)
7. . Custom Elements [↑](#footnote-ref-7)
8. . Javascript [↑](#footnote-ref-8)
9. . Shadow DOM [↑](#footnote-ref-9)
10. . Template [↑](#footnote-ref-10)
11. . Selector [↑](#footnote-ref-11)
12. . ES6 [↑](#footnote-ref-12)
13. . CSS [↑](#footnote-ref-13)
14. . CSS In JS [↑](#footnote-ref-14)
15. . SSR [↑](#footnote-ref-15)
16. . Modular [↑](#footnote-ref-16)
17. . Svelte [↑](#footnote-ref-17)
18. . https://www.webcomponents.org [↑](#footnote-ref-18)
19. . NPM [↑](#footnote-ref-19)
20. . NodeJS [↑](#footnote-ref-20)
21. . Lit [↑](#footnote-ref-21)
22. . pre-rendering [↑](#footnote-ref-22)
23. . Scope [↑](#footnote-ref-23)