# التقنيات المعاصرة لتطوير واجهات المستخدم في تطبيقات الويب
# دراسة تحليلية لأدوات *Front-End*

**الباحثة: د. إلهام القاسم**

**ملخص**

مع بداية التحول الرقمي الهائل، أصبحت واجهات المستخدم الأمامية (Front-End) حجر الأساس في تجربة المستخدم لتطبيقات الويب. أدى تعدد أطر العمل والمكتبات إلى صعوبة اختيار التقنية المناسبة، مما يستدعي إجراء دراسات مقارنة.

تهدف هذه الدراسة إلى تحليل ومناقشة أبرز تقنيات الواجهات الأمامية المعاصرة، مع إجراء مقارنة منهجية بين أطر العمل الرئيسية( React ، Angular، Vue.js) من حيث الأداء، وسهولة التعلم، وكفاءة الذاكرة، ومجالات التطبيق.

اعتمدت الدراسة على المنهج التحليلي المقارن، من خلال استعراض الدراسات السابقة وتحليل نتائجها لتقييم أداء الأطر بناءاً على معايير محددة كسرعة المعالجة ، وزمن التفاعل ، واستهلاك الذاكرة .

و تجدر الإشارة هنا إلى ندرة الدراسات الأكاديمية أو البحثية العربية المتعمقة في هذا المجال، حيث إن معظم المصادر المتاحة باللغة العربية تندرج في إطار المقالات التعليمية أو المدونات التوعية، مما يزيد من أهمية هذه الدراسة في سد جزء من هذه الفجوة

وقد كشفت النتائج عن تفوق Vue.js في سرعة المعالجة وكفاءة استخدام الذاكرة ، بينما أظهر React تفوقًا في زمن التفاعل، وكان أداء Angular مختلطًا ويميل ليكون أكثر ملاءمة للتطبيقات المؤسسية المعقدة على الرغم من استهلاكه الأعلى للموارد.

تخلص الدراسة إلى أنه لا يوجد إطار عمل "أفضل" بشكل مطلق، بل يعتمد الاختيار على متطلبات المشروع. كما تسلط الضوء على الاتجاهات المستقبلية التي ستشكل المجال، مثل دمج الذكاء

الاصطناعي، وWebAssembly، وواجهات المحادثة والواقع المعزز (AR/VR) ، والمكونات الخادمية(Server Components) ، مؤكدة على ضرورة مواكبة المطورين لهذه التطورات.

**الكلمات المفتاحية:** أطر عمل الواجهات الأمامية , React, Angular , Vue.js

## Abstract

With the onset of the massive digital transformation, front-end user interfaces have become the cornerstone of the user experience for web applications. The proliferation of frameworks and libraries has complicated the selection of appropriate technology, necessitating comparative studies.

This study aims to analyze and discuss the most prominent contemporary front-end technologies, conducting a systematic comparison between the main frameworks (React, Angular, Vue.js) in terms of performance, ease of learning, memory efficiency, and application domains.

The study adopted a comparative analytical methodology by reviewing previous studies and analyzing their results to evaluate framework performance based on specific criteria such as processing speed, response time, and memory consumption. It is noteworthy to mention the scarcity of in-depth Arabic academic or research studies in this field, as most available sources in Arabic fall within the scope of instructional articles or awareness blogs, which heightens the importance of this study in bridging part of this gap.

The results revealed Vue.js's superiority in processing speed and memory usage efficiency, while React demonstrated an advantage in response time. Angular's performance was mixed, tending to be more suitable for complex enterprise applications despite its higher resource consumption.

The study concludes that there is no universally "best" framework; rather, the choice depends on project requirements. It also highlights future trends that will shape the field, such as the integration of artificial intelligence, WebAssembly, conversational interfaces and augmented reality (AR/VR), and Server Components, emphasizing the necessity for developers to keep pace with these advancements.

**Keywords**: Front-End Frameworks, React, Angular, Vue.js.

## 1. المقدمة:

في عالم تطوير الويب، تُعتبر أطر عمل Front-End أدوات أساسية لبناء واجهات مستخدم تفاعلية وسريعة. مع تعدد الخيارات المتاحة، يصبح اختيار الإطار المناسب أمراً بالغ الأهمية. في هذه المقالة سنتعرف على أشهرها: React (مكتبة من تطوير Facebook)، Angular (إطار عمل من تطوير Google)، Vue.js (إطار عمل مفتوح المصدر).

استعرضا في هذا البحث تطور تقنيات واجهة المستخدم في الويب ومقارنتها، مع التركيز على الأدوات الحديثة لتحسين التفاعل والأداء وتجربة المستخدم.

شهد العالم خلال العقود الأخيرة تحولًا رقمياً هائلاً غيّر من طريقة تفاعل الإنسان مع البيانات والخدمات. وفي قلب هذا التحوّل تقف تطبيقات الويب التي باتت تشكل نافذة التواصل بين المستخدمين والمنصات الرقمية. تلعب واجهة المستخدم الأمامية (Front-End) دوراً جوهرياً في تحسين تجربة الاستخدام، إذ تمثل الوجه المرئي والتفاعلي للمستخدم عند تصفحه للمواقع والتطبيقات.

## 2. هدف البحث:

تهدف هذه الدراسة إلى تحليل التقنيات المعاصرة المستخدمة في بناء الواجهات الأمامية لتطبيقات الويب، واستكشاف أبرز الأدوات والممارسات التي ساهمت في تطورها، بالاضافة إلى إجراء مقارنات بين أحدث التقنيات، إلى جانب تسليط الضوء على الاتجاهات المستقبلية في هذا المجال.

## 3. الخلفية النظرية:

قبل التعرف على هذه التقنيات لابد من تسليط الضوء على بعض التعاريف والمصطلحات والمفاهيم الهامة المستخدمة في هذا المجال.

### 3.1.نشأة وتطور تقنيات الويب:

في بدايات الإنترنت، كانت صفحات الويب تُبنى باستخدام HTML فقط، مما جعلها محدودة من حيث الشكل والتفاعل. ثم ظهرت CSS لتحسين التصميم، ثم JavaScript لتوفير التفاعلية، وصولًا إلى أطر العمل الحديثة مثل React و Angular و Vue.

### 3.2.واجهة المستخدم الأمامية Front-End [11]:

وهي الجزء من تطبيق الويب الذي يتفاعل معه المستخدم مباشرة. وتشمل كل ما يظهر على الشاشة من نصوص، صور، أزرار، نماذج، والأنماط البصرية التي تنظم هذا المحتوى. تعتمد على تقنيات مثل: HTML, CSS, JavaScript

### 3.3.الواجهة الخلفية Back-End [11] :

هي الجزء المسؤول عن معالجة البيانات وتنفيذ العمليات المنطقية التي تحدث خلف الكواليس، مثل تخزين المعلومات في قواعد البيانات ومعالجة طلبات المستخدم. تعتمد على:

- لغات البرمجة: مثل.Python, PHP, ASP, Node.js
- قواعد البيانات: مثلMySQL, MongoDB, Oracle
- الخوادم (Servers): مثل Apache, Nginx.

يمكننا من خلال الجدول (1) عرض لأهم الصفات التي تميز كل من المفهومين السابقين:

| الصفة | Back-End | Front-End |
|---|---|---|
| الوظيفة | معالجة البيانات وتخزينها | العرض المرئي والتفاعل مع المستخدم |
| اللغات | PHP ،Python ،Node.js ، Java,…. | HTML ،CSS ،JavaScript |
| التفاعل | خلف الكواليس | مباشر مع المستخدم |

**الجدول (1): صفات الواجهات الأمامية والخلفية**

### 4. مكتبات وأطر عمل JavaScript[12]:

هي مجموعات من الأكواد الجاهزة والمبنية مسبقًا، تُستخدم لتبسيط تطوير التطبيقات والمواقع الإلكترونية الديناميكية. تعمل هذه الأدوات على توفير وظائف مُجرَّبة ومُحسَّنة، مما يقلل من الحاجة إلى كتابة الأكواد من الصفر وتسرع عملية التطوير. يمكن التمييز بين المفهومين:

### 4.1.المكتبات (Libraries) :

- هي مجموعات من الوظائف والأصناف البرمجية الجاهزة التي يمكن استدعاؤها حسب الحاجة لتنفيذ مهام محددة دون الحاجة لإعادة كتابة الكود .
- تُركِّز على تنفيذ مهام محددة مثل معالجة DOM (كمكتبة **jQuery** ( سيتم شرح هذا المفهوم ضمن ملحق A)) أو إدارة الحالة (مثل **Redux** (ملحق A))
- تتبع أسلوبًا تفاعليًّا حيث يتحكم المطور في متى وكيف يتم استخدامها.

### 4.2.أطر العمل (Frameworks) :

- هو بنية برمجية متكاملة تحتوي على مجموعة من الأدوات والمكتبات التي تفرض نمطًا معينًا لتطوير التطبيقات (مثل MVC)، وتقوم باستدعاء كود المطوّر ضمن هيكلها الخاص.
- تُقدِّم حلولًا شاملة للتوجيه (Routing) ، وإدارة الحالة (State Management) ، والتكامل مع الخدمات الخارجية.
- تبرز أهمية إطار العمل بشكل أساسي في كونه يتيح إمكانية إعادة استخدام التعليمات البرمجية بدلاً من إعادة كتابتها من جديد.

أخيراً يمكن القول أن الفرق الجوهري: المكتبات تُستخدم لأداء مهام محددة داخل مشروع موجود، بينما أطر العمل تُحدد كيفية بناء المشروع ككل

من أهم أطر العمل React, Vue.js, Angular .

**ملاحظة:** React (يعتبر مكتبة وأحيانًا إطار عمل)

نستعرض فيما يلي مقارنة للأطر السابقة، ومن ثم في الفقرة التي تليها عرض لأهم المفاهيم المستخدمة في بعضها.

### 4.2.1. مقارنة أطر العمل

تشير الدراسات المقارنة لتطبيقات الويب إلى أن Vue يتفوق على غيره في سرعة العرض. على سبيل المثال، أفاد Diniz–Junior (وآخرون 2022)) [13] أن زمن معالجة React أبطأ بنسبة 758% و Angular أبطأ بنسبة 595% مقارنة مع .Vue ، بينما لاحظ Lipski(وآخرون (2021)) [15] أن Vue يقدم باستمرار سرعة عرض أفضل مقارنة بـ . Angular أما من ناحية كفاءة الذاكرة، تشير العديد من الدراسات إلى أن Vue يستهلك ذاكرة أقل (أو يحقق حجم حزمة أصغر) منAngular ، الذي يظهر غالبًا استخدامًا أعلى للذاكرة؛ بينما أداء React في هذا الجانب معتدل لكن أقل توثيقًا.

فيما يخص زمن التفاعل(Time–to–Interactive) ، قاست إحدى الدراسات زمن التفاعل في React بـ 300 مللي ثانية، وأظهرت دراسة Diniz–Junior (وآخرون (2022) )

أن Vue وAngular كانا أبطأ من React بنسبة 33% و50% على التوالي. كما تشير بعض الدراسات إلى أن Angular يعاني من أوقات تحميل أو بدء تشغيل أطول، على الرغم من وجود تقرير واحد يشير إلى تحسن بنسبة 45% على كل من React وVue في العمليات المعقدة. تدعم هذه النتائج بشكل مباشر أن Vue يقدم عادة سرعة عرض وكفاءة ذاكرة متفوقة، بينما قد يتيح React تفاعلية أسرع في بعض السيناريوهات، ويظهر Angular أداءً مختلطًا يميل إلى تفضيل العمليات المعقدة على حساب زيادة استخدام الذاكرة وأوقات التحميل.

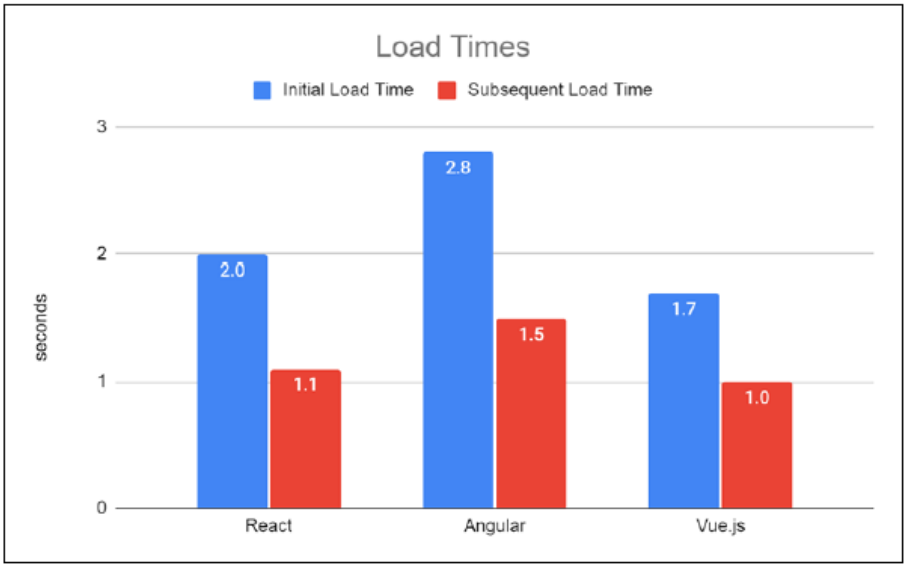
يظهر الجدول (2) مقارنة بين أهم الأطر:

| نوع المقارنة | React | Vue | Angular |
|---|---|---|---|
| نوع الأداة [9][2][1][12] | مكتبة | إطار عمل | إطار عمل |
| المطوّر [2][1][12] | Meta(Facebook) | Evan Yoy | من تطوير Google |
| طريقة كتابة الأكواد [2][1][12] | JSX | HTML ، JavaScript | TypeScript (ملحق A) |
| سرعة العرض (Rendering Speed) [14] | أداء جيد بفضل الـ Virtual DOM (ملحق A) | أوقات عرض أسرع | أداء جيد لكن مع أوقات تحميل أطول |
| زمن المعالجة [13] | أبطأ بـ 758% من في زمن Vue المعالجة | الأسرع في زمن المعالجة | أبطأ بـ 595% من في زمن Vue المعالجة |
| استهلاك الذاكرة (Memory Usage) [13][15] | حجم الحزمة أقل بـ 45% من Angular | حجم الحزمة أقل بـ 54% منAngular<br>استهلاك أقل للمساحة على القرص | أعلى استهلاك للذاكرة<br>استهلاك أعلى للمساحة على القرص |
| زمن التفاعل (Time-to-Interactive) [13] | أفضل زمن تفاعل (300 مللي ثانية) | أبطأ من React بـ 33% | أبطأ من React بـ 50% |
| إدارة الحالة [2][1][12] | متقدمة (Redux أو Context API) | بسيطة باستخدام Vuex و Data | - |
| الأداء [2][1][12] | عالي جداً باستخدام Virtual DOM | سريع | جيد ويستخدم (Change Detection) |

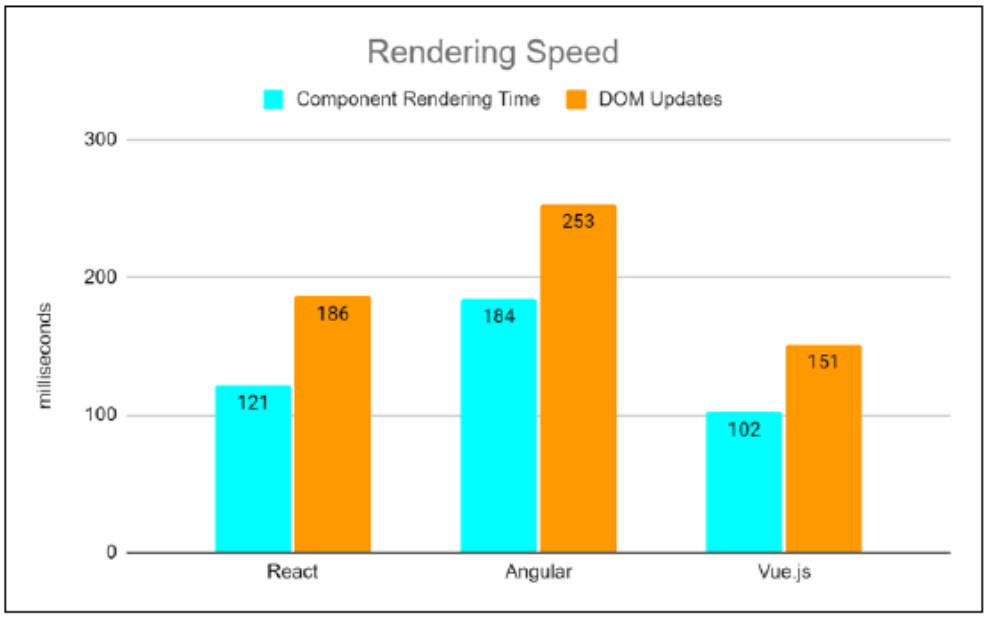
| نوع المقارنة | React | Vue | Angular |
|---|---|---|---|
| مكان الاستخدام [2][1][12] | للمشاريع الكبيرة | مناسب للمشاريع الصغيرة والكبيرة | تطبيقات Enterprise الكبيرة والمعقدة |
| المكتبات المتوافقة | آلاف المكتبات والحزم على npm (تقدّر بعشرات الآلاف)، **مثل**: React Router، Redux، Axios، Styled-components، React Hook Form | مئات المكتبات النشطة، مع نمو سريع Vue 3خاصة بعد **مثل**: Vue Router ، Pinia ، Vuex، Quasar، Vuetify | منظومة رسمية قوية + مكتبات خارجية ( مئات الحزم)، مثل: Angular Material، NgRx، PrimeNG، Nebular، Kendo UI |
| مستوى التعقيد [2][1][12] | أعلى قليلاً | أكثر بساطة | - |
| سهولة التعلم [2][1][12] | متوسطة | سهلة جداً | صعب التعلم |
| إمكانية التعديل [17] | يعتمد على مبدأ المكونات مما يجعل الكود قابلًا لإعادة الاستخدام وسهل التحديث | يتميز ببساطة الكود وسهولة التعلم، مما يجعل التحديث أسهل خاصة للمشاريع الصغيرة والمتوسطة. لكن في المشاريع الضخمة قد يحتاج إلى تنظيم إضافي | يوفر هيكلية صارمة (MVC) وأدوات مدمجة مثل Dependency Injection، مما يسهل التعديل في المشاريع الكبيرة، لكنه يتطلب خبرة أكبر من الفريق |

**الجدول (2): مقارنة لأطر العمل**

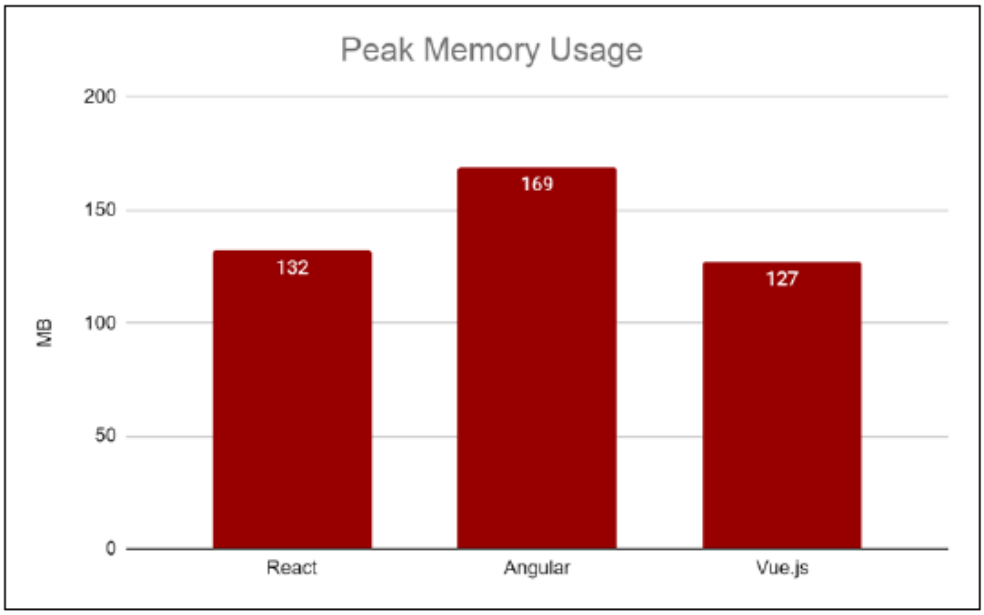
كما تظهر الأشكال(1,2,3,4) مخططات بيانية للمقارنة بين النماذج السابقة في أهم المقارنات [14]



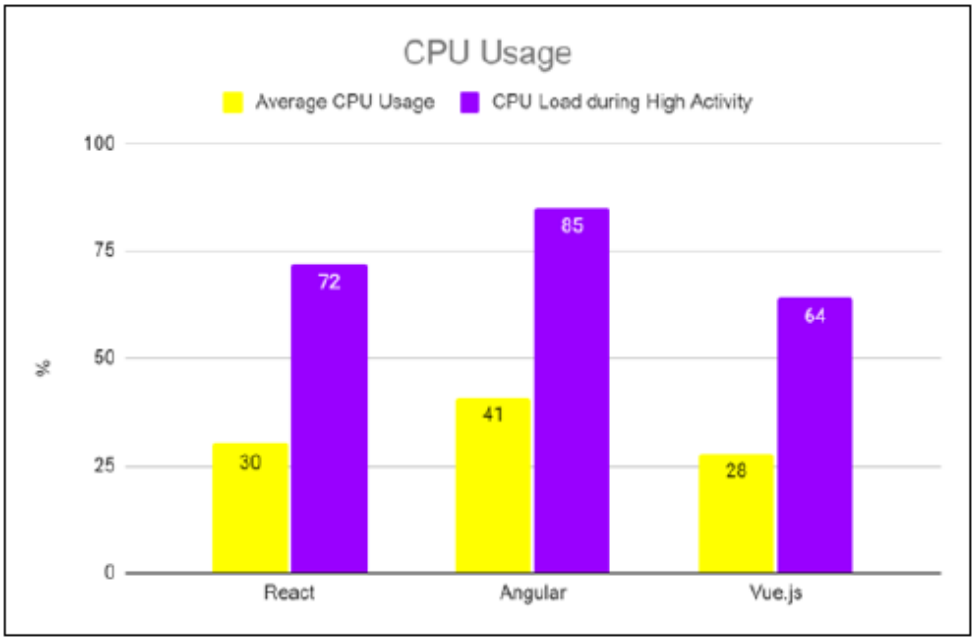

الشكل (1) Load Times




**الشكل (2) Rendering Speed**




**الشكل(3) Peak Memory Usage**




**الشكل(4) CPU Usage**

**كذلك يمكن عرض مثال متشابه لبعض الأطر موضع الدراسة:**

**مثال عملي باستخدام React.js**

ننشئ مكونًا يعرض عدّادًا يمكن زيادته بالنقر على زر ويُعاد عرض المكون تلقائيًا.

```
// JavaScript
import React, { useState } from 'react';
function Counter() {
      const [count, setCount] = useState(0);
    return (
        <div style={{ textAlign: 'center' }}>
         <h2>العدد الحالي: {count}</h2>
          <button onClick={() => setCount(count + 1)}>زيادة</button>
          </div>
        );
```

```
}
```

**الشرح:**

- useState لتخزين القيمة.
- كل مرة يُنقر الزر، تزداد القيمة ويُعاد عرض المكون تلقائيًا.
- يتطلب هذا وجود مشروع React باستخدام أدوات مثل Create React App أو Vite

**باستخدام Vue:**

```
//html
<div id="app">
  <h2>العدد الحالي: {{ count }}</h2>
  <button @click="count++">زيادة</button>
</div>

<script src="https://cdn.jsdelivr.net/npm/vue@2"></script>
<script>
  new Vue({
    el: '#app',
    data: {
      count: 0
    } });
</script>
```

**الشرح:**

- البيانات موجودة داخل data
- التفاعل يتم عبر @click ويتم ربطه مباشرة بالبيانات باستخدام الربط الثنائي.
- يمكن تشغيل هذا الكود مباشرة في ملف HTML دون إعداد مشروع معقد، مما يجعل Vueمثاليًا للتطبيقات الخفيفة.

```
<div id="app">
  <h2>{{ count }}</h2>
  <button @click="count++">زيادة</button>
```

```
</div>
<script>
  new Vue({ el: '#app', data: { count: 0 } });
</script>
```

**باستخدام Angular**

```
//TypeScript
// counter.component.ts
import { Component } from '@angular/core';

@Component({
  selector: 'app-counter',
  template: `
    <div>
      <h1>Counter: {{ count }}</h1>
      <button (click)="increment()">Increment</button>
    </div>
  `,
})
export class CounterComponent {
  count = 0;

  increment() {
    this.count++;
  }
}
```

**الخصائص:**

- يستخدم **TypeScript** بشكل افتراضي.
- يعتمد على **Decorators** مثل `@Component`
- **Template**منفصل عن المنطق (يمكن أن يكون في ملف `html` منفصل).

**4.2.2. المفاهيم الرئيسية لأطر العمل:**

نذكر فيما يلي بعض المفاهيم التي وردت في جدول (2) :

**أولا: المفاهيم الرئيسية في React:**

- **المكونات :(Components)** كل جزء في الواجهة يُبنى كمكون مستقل مثل زر، مربع نص، قائمة، دالة (function)، صف(class)، ما يسهّل إعادة الاستخدام.
- **الحالة :(State)** تخزين البيانات الخاصة بكل مكون، وتحديثها يؤدي لإعادة عرض المكون تلقائيًا.
- **JSX:** صيغة خاصة تسمح بكتابة كود HTML داخل JavaScript

**ثانياً: المفاهيم الرئيسية في:Vue**

- **التوجيه الثنائي للبيانات :(Two-Way Binding)** يربط البيانات بين النموذج والواجهة بسهولة.
- **المكونات :**كما فيReact ، يتم بناء الواجهة عبر مكونات قابلة لإعادة الاستخدام.
- **قالب HTML واضح :** يفصل بين منطق التطبيق والتصميم.

**ثالثاً: الفرق بين React و Vue في بنية المكونات والتعامل معها:**

من خلال الجدول (3) يمكن توضيح الفرق بينهما كمايلي:

| العنصر | React | Vue |
|---|---|---|
| هيكل المكون | Class أو Function | كائن يحتوي على قوالب وبيانات |
| لغة القالب | JSX<br>(دمج HTML داخل JavaScript) | Vue عادي + توجيهات HTML |
| التعامل مع البيانات | useState, props | data, props, computed |
| التفاعل مع الأحداث | onClick={...} | @click="..." |
| إدراج مكونات | <Component /> | <component></component> |

**الجدول (3): الفرق بين React و Vue في بنية المكونات والتعامل معها**

## 5. أدوات التطوير وإدارة المشاريع فيFront-End

### 5.1.أدوات إدارة الحزم NPM وYarn :

تسمح هذه الأدوات بإدارة المكونات الطرفية والتحديثات بسلاسة.

- **NPM (Node Package Manager)** [3]: يأتي مع Node.js ويُستخدم لتحميل وإدارة المكتبات والأطر (مثل React وVue )
- **Yarn**بديل لـ NPM ويتميز بسرعة أكبر وإدارة أفضل للاعتمادات[6] .

### 5.2.أدوات التهيئة والبناء Webpack وBabel :

- **Webpack** [7]: يجمع كل ملفاتJavaScript ، CSS، والصور في ملف واحد لتحسين الأداء.
  - يدعم التقسيم الديناميكي للصفحات.
  - يسمح بتحميل Lazy للعناصر حسب الحاجة (ملحق A)
- **Babel** [8]: محوّل JavaScript يُستخدم لتحويل الكود الحديث (ES6+) إلى صيغة متوافقة مع المتصفحات القديمة.

يبين المثال الثالي كيفية تحويل التعليمات البرمجية باستخدام Bable

```
// ES6
const greet = name => Hello ${name}';
// بعد تحويل Babel
var greet = function(name) {
  return "Hello " + name;
};
```

### 5.3.التحكم في الإصدارات Git وGitHub :

- **Git:** أداة لتتبع التعديلات على الملفات وتسهيل العمل الجماعي.
- **GitHub:** منصة لاستضافة الأكواد ومشاركة المشاريع مع الآخرين .

## 6. التصميم المتجاوب:[16]

في ظل التنوع الهائل لأجهزة الوصول إلى الإنترنت، والتي تتراوح بين الهواتف الذكية والأجهزة اللوحية والحواسيب المكتبية وأجهزة التلفاز الذكية، لم يعد تصميم موقع ويب بحجم ثابت يناسب جميع الشاشات مقبولاً.

استجابة لهذه التحديات، ظهر مفهوم "التصميم المتجاوب" (Responsive Web Design) الذي يُعرّف على أنه "نهج في تصميم وتطوير الويب يهدف إلى جعل صفحات الويب تُعرض وتُشغّل بشكل مثالي على مجموعة متنوعة من الأجهزة وحجم الشاشات" .

### 6.1.مفهوم التصميم المتجاوب:

يشير هذا المفهوم إلى تصميم صفحات ويب **تتكيف تلقائياً** مع حجم الجهاز المستخدم، مهما كان نوعه (Mobile, Desktop...,. )

الهدف هو **توفير تقنية مثالية** للمستخدم دون الحاجة إلى التمرير أو التكبير اليدوي.

### 6.2.التقنيات المستخدمة:

- **Media Queries:** تُستخدم لتطبيق أنماط CSS وفقًا لأبعاد الشاشة:

```
/* شاشة صغيرة مثل الهاتف*/
@media (max-width: 600px) {
  body { background-color: lightblue; }
}
```

يسمح هذا بتغيير التصميم مثل الخطوط أو توزيع العناصر بحسب حجم الشاشة.

### 6.3.الأطر المساعدة في التصميم المتجاوب:

- **Bootstrap:** إطار شهير يعتمد على **نظام الشبكةGrid System** ، ويحتوي على مكونات جاهزة مثل الأزرار والقوائم التفاعلية.

```
//HTML
<div class="container">
```

```
<div class="row">
  <div class="col-md-6">نصف الشاشة</div>
  <div class="col-md-6">نصف الشاشة الآخر</div>
</div>
</div>
```

- **Tailwind CSS:** نظام تصميم حديث يعتمد على **الصفوف البرمجية الجاهزة (Utility-first)**، ويتيح تخصيصًا واسعًا مع كود أقل.

```
<button class="bg-blue-500 text-white px-4 py-2 rounded">
  اضغط هنا
</button>
```

## 7. بعض من الاتجاهات المستقبلية في تقنيات Front-End :

### 7.1. الدمج بين الذكاء الاصطناعي وواجهات المستخدم

وذلك من خلال استخدام خوارزميات الذكاء الاصطناعي لتخصيص واجهة المستخدم بناءً على سلوك المستخدم وتفضيلاته.

**أمثلة تطبيقية:**

- **أنظمة التوصية الذكية** :مثل عرض منتجات بناءً على النشاط في الموقع.
- **مساعدات صوتية مدمجة** :تعالج اللغة الطبيعية لتقديم دعم فوري.
- **أتمتة تصميم الواجهة** :باستخدام أدوات تعتمد على AI لتوليد تخطيطات ديناميكية.

### 7.2. WebAssembly الأداء الحقيقي داخل المتصفح

**WebAssembly** : هو معيار جديد يتيح تشغيل لغات مثل C++ أو Rust داخل المتصفح بسرعة تقارب التطبيقات الأصلية[10].

**الميزات:**

- أداء عالي للألعاب والتطبيقات الرسومية.
- دمج التقنيات غير المدعومة أصلاً بالمتصفح.

**التطبيقات المحتملة:**

- تحرير الصور والفيديو داخل المتصفح.
- محاكاة أنظمة تعليمية أو علمية تفاعلية.

### 7.3. واجهات غير مرئية المحادثات وAR/VR:

**واجهات محادثة:**

- **التحكم بالصوت** عبر Web Speech API
- **Chatbots** داخل صفحات الويب تقدم خدمة مباشرة دون تدخل بشري.

**واجهات الواقع المعزز والافتراضي:**

- استخدام **WebXR** لبناء مشاهد ثلاثية الأبعاد في متصفحات حديثة.
- تفاعل المستخدم مع عناصر الواجهة في بيئة افتراضية.

### 7.4. التصميم الموجه بالبياناتData-Driven UI

في المستقبل، لن يُبنى التصميم يدويًا بالكامل، بل يتكوّن **ديناميكيًا من بيانات حية:**

**من هذه التطبيقات نذكر:**

- لوحات تحكم تتغير بناءً على نشاط المستخدم.
- تكيّف المحتوى والخيارات تلقائيًا حسب السياق.

## 8. الخاتمة:

يتضح من خلال هذه الدراسة أن تقنيات **Front-End** ليست مجرد أدوات برمجية، بل منظومة متكاملة تؤثر بشكل مباشر على تجربة المستخدم، الأداء، وسهولة تطوير المشاريع.

أظهرت المقارنات أن **Vue.js** يتميز بالبساطة وسرعة المعالجة، بينما يوفر **React** مرونة كبيرة ودعمًا واسعًا من المجتمع، في حين أن **Angular** يظل الخيار الأنسب للتطبيقات المؤسسية المعقدة رغم استهلاكه العالي للموارد.

لا يوجد إطار عمل "أفضل" بشكل مطلق، بل يعتمد الاختيار على **طبيعة المشروع، حجم الفريق، والموارد المتاحة** .ومع ذلك، فإن الاتجاهات المستقبلية مثل **WebAssembly، الذكاء**

**الاصطناعي، وواجهات الواقع المعزز/الافتراضي** ستعيد تشكيل طريقة بناء واجهات المستخدم، مما يستدعي من المطورين مواكبة هذه التطورات باستمرار.

**التوصيات**

- **للباحثين** :توسيع الدراسات حول تأثير الذكاء الاصطناعي في تخصيص واجهات المستخدم وتفاعليتها.
- **للمطورين** :الاستثمار في تعلم أكثر من إطار عمل، مع التركيز على بناء تطبيقات متجاوبة وآمنة وسريعة الأداء.
- **للمؤسسات التعليمية** :إدراج مقررات حديثة في مناهج تطوير الويب تشمل WebAssembly، Server Components، وتقنيات حماية واجهة المستخدم.
- **لأصحاب المشاريع** :اختيار إطار العمل الأنسب بناءً على طبيعة المشروع، وليس وفقاً للشهرة فقط.
- **للمستقبل** :الاستعداد للتكامل مع تقنيات الصوت، الواقع المعزز، وواجهات غير تقليدية.

## ملحق A

**تعاريف ومصطلحات:**

**نستعرض ضمن هذا الملحق شرح لبعض المصطلحات (مكتبات – لغات.. التي ذكرت ضمن المقالة:**

**1. مكتبة jQuery:**

- هي مكتبة JavaScript شهيرة ظهرت قبلReact ، وتُستخدم لتبسيط التعامل مع DOM، الأحداث، والطلبات الشبكية.

**مثال :**

```
// html
<button id="btn">اضغط هنا</button>
```

```
<script src="https://code.jquery.com/jquery-
3.6.0.min.js"></script>
<script>
  $('#btn').click(function() {
    alert('تم الضغط!');
  });
</script>
```

**2. Redux:**

- هي مكتبة جافا سكريبت لإدارة حالة التطبيقات (application state) بطريقة مركزية وقابلة للتنبؤ، خاصةً في التطبيقات الكبيرة والمعقدة .تعتمد على مفهوم "مصدر واحد للحقيقة" (single source of truth) حيث يتم تخزين حالة التطبيق بالكامل في كائن واحد يسمى "المخزن" (store) بدلاً من تخزينه في مكونات منفصلة، وذلك بهدف تسهيل تتبع التغييرات والوصول إلى الحالة من أي مكان في التطبيق.
- يتم تحديث الحالة فقط من خلال "الإجراءات" (actions) التي تصف التغييرات، والتي يتم معالجتها بواسطة "دوال الاختزال" (reducers)، وهي عبارة عن دوال نقية (pure functions) والتي تقوم بمعالجة الإجراءات وتحديث الحالة.
- Redux يهدف إلى جعل إدارة الحالة أكثر وضوحًا، قابلة للتصحيح، وقابلة للتوسع، وهو شائع الاستخدام مع إطارات العمل مثل React

**3. Virtual DOM : [14]**

- هيكل بيانات خفيف في الذاكرة يمثل نسخة تجريدية (افتراضية) من شجرة الـ DOM الحقيقية الموجودة في المتصفح، تُستخدم في أطر العمل البرمجية (مثل React و Vue) لتحديث واجهة المستخدم بكفاءة.
- عند حدوث تغيّر في البيانات أو الحالة (state)، يُنشئ الإطار نسخة محدثة من Virtual DOM، ثم يقارن الاختلاف بين النسخة القديمة والنسخة الجديدة، ويحوِّل فقط الفروقات الضرورية إلى تغييرات على الـ DOM الحقيقي (عملية تُعرف بـ reconciliation). بهذا الأسلوب، يُقلل من عدد عمليات التحديث المكلفة على الـ DOM الفعلي، مما يحسِّن الأداء.

**4. TypeScript:[5]**

- لغة برمجة عالية **المستوى** تُعتبر مجموعة شاملة فوق JavaScript، فأي كود JavaScript هو أيضاً صالح أن يكون كود TypeScript
- الهدف الأساسي منها هو **اكتشاف الأخطاء في وقت التطوير (compile-time)** بدلاً من وقت التشغيل(runtime) ، مما يزيد من متانة الكود ويقلل من الأخطاء المنطقية.

**5. التحميل المؤجل Lazy في تقنيات الوب:[8]**

- **بدون Lazy:** عندما زيارة موقعاً إلكترونياً، يقوم المتصفح بتحميل جميع الأجزاء والموارد الخاصة بالصفحة مرة واحدة، حتى تلك التي لا تحتاج لرؤيتها فوراً أو قد لا تحتاجها أبدًا (مثل صور في أسفل الصفحة، أو أقسام مخفية).
- **Lazy Loading:** هي تقنية ذكية حيث لا يتم تحميل المورد (مثل صورة، مقطع فيديو، أو حتى جزء من الكود) إلا في اللحظة التي يصبح فيها هذا المورد مطلوباً أو على وشك أن يظهر للمستخدم.

**يطبق هذا المفهوم بشكل أساسي من خلال:**

1. التقسيم الديناميكي للشفرة :(Dynamic Code Splitting) حيث يقسم Webpack الكود إلى عدة "حزم (Bundles) "أو قطع صغيرة بدلاً من ملف واحد ضخم.
2. التحميل عند الحاجة :الملفات أو المكونات الخاصة بقسم معين في الموقع (مثل صفحة "الإعدادات" أو "الملف الشخصي") لا يتم تحميلها مع الصفحة الرئيسية وإنما يتم تحميلها فقط عندما يقوم المستخدم بالضغط للانتقال إلى ذلك القسم.

**مثال على ذلك:**

**في موقع تجارة إلكترونية:** كود وصور صفحة "عربة التسوق" لا يتم تحميلها إلا عندما يضغط المستخدم على أيقونة العربة.

**في تطبيق وسائل التواصل الاجتماعي:** كود وصور قسم "الإشعارات" لا يتم تحميله إلا عندما يضغط المستخدم على جرس الإشعارات

- **الفائدة من ذلك:**

- **تحسين أداء تحميل الصفحة الأولي** :حيث يصبح حجم الملف الأولي الذي يجب تحميله أصغر بكثير.
- **توفير في استهلاك البيانات** :للمستخدمين الذين لا يتصفحون كل أقسام الموقع.
- **استخدام أكثر كفاءة لموارد المتصفح والخادم**

## المراجع

1.Vue.js Official Docs – https://vuejs.org

2. React Official Docs – https://reactjs.org

3. https://docs.npmjs.com/

4. Node.js in Action (Cantelon, 2020)

5. https://www.typescriptlang.org/

6. classic.yarnpkg.com/docs

7. https://webpack.js.org/

8. https://babeljs.io/docs

9. https://angular.dev/overview

10. https://webassembly.org/

11. Full Stack Web Development: Master Front-End and Back-End Development Techniques Paperback – April 14, 2025
by Oliver Sutherland (Author)

12. Haverbeke, M. (2018). Eloquent JavaScript: A Modern Introduction to Programming (3rd ed.). No Starch Press

13. Diniz-Junior Raimundo N. V., Caio César L. Figueiredoy, Gilson De S.Russo, Marcos Roberto G. Bahiense-Junior, Arbex Mateus V. L., Lanier M. Dos Santos, R. F. Da Rocha, Renan R. Bezerra, and F. Giuntini. “Evaluating the Performance of Web Rendering Technologies Based on JavaScript: Angular, React, and Vue.” Latin American Computing Conference / Conferencia Latinoamericana En Informatica, 2022.

14. Mikita Piastou. “Comprehensive Performance and Scalability Assessment of Front-End Frameworks: React, Angular, and Vue.js.” World Journal of Advanced Engineering Technology and Sciences, 2023.

15. Piotr Lipski, Jarosław Kyć, and B. Pańczyk. “Comparative Analysis of the Angular 10 and Vue 3.0 Frameworks.” Journal of Computer Sciences Institute, 2021.

16. **Marcotte, E. (2010, May 25).** *Responsive Web Design***. A List Apart.**

Available at: https://alistapart.com/article/responsive-web-design/

17. https://namastedev.com/blog/react-vs-vue-vs-angular-a-2025-comparison-9/