AMERICAN JOURNALS
Open Access, Peer Reviewed Journals

American Journal of Technology and Applied Sciences
ISSN (E): 2832-1766 Volume 42, November - 2025

A COMPARATIVE ANALYSIS OF ANGULAR AND REACT FRAMEWORKS FOR WEB APPLICATION DEVELOPMENT

Ulugbek Urunov
Software Engineer, USA

ABSTRACT

This article examines modern approaches to web application development using Angular and React. It presents the historical evolution of client-side web development, examines the background to the emergence of modern JavaScript frameworks and libraries, and highlights the development of single-page application technologies. It analyzes the architectural principles of Angular and React, including the component approach, state management mechanisms, user interface update methods, the use of TypeScript, Virtual DOM, Change Detection, and modern performance optimization tools. A comparative analysis of the technologies under consideration is conducted based on key criteria, including architecture, performance, scalability, security, maintainability, development speed, integration with external services, and the scope of practical application. Particular attention is paid to modern trends in the development of Angular and React related to the implementation of reactive mechanisms, server-side rendering, cloud technologies, and artificial intelligence tools. The study found that the choice between Angular and React should be based on the project's scale, software architecture requirements, development team qualifications, and the future maintenance prospects of the information system. The findings can be used in the design and development of modern web applications for various purposes.

Scientific Novelty. The scientific novelty of this study lies in its comprehensive comparative analysis of Angular and React, taking into account current trends in front-end technologies and the latest architectural changes in both platforms. Unlike most existing studies, this paper systematizes the advantages and limitations of Angular and React not only from an architectural perspective but also from a performance, scalability, security, maintainability, integration with cloud services, and potential for use in developing web applications for various purposes. Particular attention is paid to the analysis of modern optimization mechanisms, including Signals in Angular, Concurrent Rendering, and Server Components in React. This allowed us to identify current development trends for these technologies and

KEYWORDS

Angular, React, web applications, JavaScript, TypeScript, front-end development, component architecture, Virtual DOM, Change Detection, user interface, SPA, software engineering.

formulate practical recommendations for choosing a software platform based on the requirements of a specific project. **Purpose of the Study.** The purpose of the study is to conduct a comparative analysis of the Angular framework and the React library, identify their architectural features, advantages, and limitations, and determine the most effective use cases for developing modern web applications, taking into account requirements for performance, scalability, security, and software maintainability.	

Introduction

The development of information technology in the 21st century is accompanied by constant improvements in software development tools. Particularly rapid changes are occurring in the field of web technologies, where demands for application speed, user interface convenience, scalability, and security are constantly increasing. While at the beginning of the World Wide Web, websites were primarily static HTML pages designed solely to display information, today they have transformed into complex software systems that perform a wide range of functions comparable to those of traditional desktop applications.

The history of web development dates back to the late 1980s, when British scientist Tim Berners-Lee proposed the concept of the World Wide Web. The first web pages were created exclusively in HTML and contained static text, images, and hyperlinks. User interaction with these resources was minimal, and any content change required a full page reload.

With the growth of the internet and the increasing number of users, the need to create more interactive web applications arose. In the mid-1990s, JavaScript, developed by Netscape, emerged. Initially, it was seen as a simple tool for running simple scripts directly in the browser. However, subsequent development of the language, the standardization of the ECMAScript specification, and significant improvements in browser engine performance have made JavaScript one of the most in-demand programming languages in the world.

The next important stage in the development of web technologies was the emergence of AJAX (Asynchronous JavaScript and XML), proposed in the early 2000s. The use of asynchronous data exchange between the client and server significantly increased the speed of web applications, as it made it possible to update individual page elements without completely reloading the page. The introduction of AJAX paved the way for the development of the single-page application (SPA) concept, which subsequently shaped the further development of front-end development.

The first major JavaScript libraries and frameworks began to actively develop in the late 2000s. One of the most popular solutions was jQuery, which significantly simplified working with the DOM tree, event handling, and AJAX requests. Despite its widespread popularity, jQuery did not provide a complete application architecture and gradually ceased to meet the requirements of developers of complex corporate information systems.

The need to systematize web interface development processes has led to the emergence of JavaScript frameworks, among which AngularJS (Google, 2010) has become a key player. The introduction of concepts such as MVC, two-way data binding, and declarative programming has significantly optimized the process of creating high-load client applications.

However, as projects grew in scale, AngularJS's shortcomings began to emerge, including performance, the complexity of supporting large applications, and architectural limitations. This prompted the development of a completely new version of the Angular platform, which was released in 2016. Unlike its predecessor, Angular is a modern framework built on TypeScript, employing a component-based architecture, dependency injection, modular code organization, and modern performance optimization tools. According to Google developers, the new architecture significantly improved application performance, simplified the maintenance of large projects, and ensured the long-term development of the platform.

Parallel to the development of Angular, the React library, developed by Meta (formerly Facebook) and released in 2013, gained widespread adoption. React introduced a new paradigm for building user interfaces. Unlike monolithic frameworks, this library is limited to the functionality of the view layer, giving developers freedom in choosing the technology stack for state management, routing, and server-side integration.

A distinctive feature of React was its use of the Virtual DOM, which minimizes the number of operations that directly modify the browser document structure. This approach significantly improved interface performance, especially when working with large amounts of data and complex components. Furthermore, React offered a declarative programming model, component architecture, and code reuse, significantly simplifying the development of large web applications.

In recent years, Angular and React have firmly established themselves as key front-end development tools. These frameworks and libraries are actively used in the creation of high-load systems, from banking and government services to educational platforms and cloud solutions. Experts emphasize that the choice between these technologies remains a subject of heated debate in the professional community, as each has a unique set of functionality, specific limitations, and optimal use cases [1].

Modern trends in the digital transformation of the economy, the development of cloud computing, microservice architecture, artificial intelligence, and mobile technologies are placing new demands on user interface development tools. Application scalability, code maintainability, security, performance, and integration with other information systems are becoming increasingly important. Therefore, a comparative analysis of Angular and React is of significant scientific and practical interest.

As research by domestic and international authors shows, there is no universal solution equally effective for all project categories. In some cases, using Angular significantly reduces development time thanks to its built-in tools, while in other projects, using React provides greater architectural flexibility and better user interface performance [2]. Therefore, the choice of technology should be based on an analysis of the specific project's characteristics, the development team's qualifications, the client's requirements, and the software's future development prospects.

Furthermore, both technologies have continued to evolve rapidly in recent years. Angular regularly improves its Change Detection mechanisms, implements Signals, develops server rendering, and optimizes performance. React is also constantly updated, introducing new features like Concurrent Rendering, Server Components, Suspense, and improved application state management mechanisms. This demonstrates the rapid evolution of modern JavaScript and the need for regular reconsideration of approaches to choosing development tools.

Theoretical Foundations of Modern JavaScript Frameworks

Today, having a website or web application has become a must for any serious business. Development tools are evolving alongside this: frameworks and libraries have become industry standards, helping to cope with the growing complexity of modern web projects. Using community-tested solutions is a rational way to create high-quality and functional products. However, choosing a specific framework requires a balanced analysis of its strengths and weaknesses.

Modern web development is one of the most dynamically developing areas of software engineering. The constant growth of internet users, increasing demands on the speed of digital services, the spread of cloud computing, and the development of mobile technologies have led to the need for more advanced user interface development tools. While websites previously served primarily as informational platforms, today they are complex software systems that support numerous business processes in real time. Therefore, technologies that enable the creation of high-performance, scalable, and easy-to-maintain client applications are becoming increasingly important.

As researchers note, the development of front-end development is closely linked to the evolution of JavaScript. Initially, this language was used exclusively for implementing minor interactive elements on web pages, but with the increasing computing power of browsers, its functionality has expanded significantly [3]. Currently, JavaScript is used to develop full-fledged client-side applications, server-side solutions, mobile software, and even desktop applications. This versatility has led to the formation of a large ecosystem of libraries and frameworks, each focused on solving a specific class of problems.

One of the most significant stages in the development of web technologies was the introduction of a component-based approach to user interface design. In traditional web applications, the interface was designed as a single unit, significantly complicating code maintenance. Even minor changes to individual elements could affect a large number of interconnected files, increasing the likelihood of errors and hampering the further development of the software.

Component architecture offered a fundamentally different approach to software organization. This concept breaks the user interface into independent functional elements, each responsible for performing a clearly defined task. This organization ensures component reuse, reduces code coupling, and significantly facilitates testing of individual application components. According to software engineering experts, the component approach is one of the key factors in improving the quality of modern web applications [4].

Along with component architecture, the concept of declarative programming has become widespread. Unlike the imperative approach, in which the developer describes in detail the sequence of actions required to change the interface, the declarative model involves describing the final state of the user interface. All page element updates are performed automatically by the library or framework used. This significantly reduces the amount of code and reduces the likelihood of logical errors.

Another important feature of modern JavaScript frameworks is the use of design patterns, which standardize application architecture. The most common are MVC (Model View Controller), MVVM (Model View ViewModel), and component-oriented architecture. Using such architectural solutions allows for the separation of business logic, user interface, and data management, significantly facilitating software maintenance and making collaborative development more efficient.

As an analysis of modern research shows, one of the most serious problems in large web applications is state management. In simple projects, information can be passed directly between components via

parameters, but as the number of interconnected elements increases, this approach becomes ineffective. This leads to difficulties with data synchronization, change control, and component reuse.

To solve this problem, specialized mechanisms for centralized application state storage have been developed. In the Angular ecosystem, the most widely used tool is NgRx, which implements the Redux architecture using reactive programming. In React, a similar function is performed by Redux Toolkit, Zustand, MobX, Recoil, and several other libraries. As researchers note, properly organized state management can significantly improve software reliability and simplify the maintenance of large corporate information systems.

Performance occupies a special place in modern front-end development. Users expect near-instantaneous interface response, regardless of the complexity of the operations performed. According to modern UX research, even a page load delay of several hundred milliseconds can negatively impact user satisfaction and lead to lower conversion rates for digital services [5, 6].

To improve performance, modern frameworks use various algorithms for updating the user interface. The most well-known solution is Virtual DOM technology, first widely adopted by React. Instead of directly modifying the browser's actual DOM, a virtual copy is created, after which the previous and new states of the element tree are compared. Changes are applied only to those nodes that have actually been modified. This approach significantly reduces the number of costly browser interactions and ensures high UI performance.

Angular uses a different performance optimization mechanism. Instead of the Virtual DOM, it uses its own change detection system, which analyzes data changes within components and updates only the necessary UI elements. In recent versions of Angular, this mechanism has been significantly improved by the introduction of Signals technology, which reduces the number of unnecessary checks and improves overall application performance.

The advent of TypeScript marked a turning point in web development. The introduction of strong typing, object-oriented principles, and other tools common to classical languages allowed developers to minimize errors at compile time. This significantly simplified the maintenance of large-scale systems and improved code readability compared to standard JavaScript.

Angular is natively focused on TypeScript and treats it as the primary development language. React, on the other hand, can be used with both JavaScript and TypeScript, giving developers greater freedom of choice. According to some authors, the mandatory use of TypeScript has been one of the factors behind Angular's popularity among large corporate organizations, which place particular emphasis on the quality of software architecture and long-term project maintenance.

It's worth noting that modern web applications rarely function in isolation. They are typically part of distributed information systems and interact with numerous external services via REST APIs, GraphQL, WebSockets, and other data exchange technologies. Therefore, an important characteristic of any modern framework is ease of integration with external information resources.

Angular includes a built-in HTTP client, dependency injection, request processing mechanisms, and interceptors, which significantly simplify network interactions. React takes a more flexible approach and allows the use of any third-party libraries, such as Axios, TanStack Query, or the built-in Fetch API. This approach gives the developer greater freedom of choice, but requires the developer to design the server-side interaction architecture themselves.

An equally important aspect of modern web development is security. The increasing number of cyberattacks necessitates the implementation of user data protection mechanisms as early as the software design stage. The most common threats remain cross-site scripting (XSS), malicious HTML injection, cross-site request forgery (CSRF) attacks, and various methods of unauthorized data access. A review of scientific literature shows that Angular contains built-in protections against the most common attacks, including automatic HTML content sanitization and the DomSanitizer mechanism, which prevents the execution of potentially dangerous code [7]. In React, security is more dependent on the developer's actions, as the library provides more flexible mechanisms for working with user-supplied content. Despite the presence of built-in data escaping, the use of certain features, such as dangerouslySetInnerHTML, requires strict adherence to information security regulations.

Angular's architectural features. Angular is a full-featured framework for developing client-side web applications, supported by Google and the global developer community. Unlike many JavaScript libraries, Angular provides a ready-made ecosystem of tools that allows you to implement virtually all the main tasks of modern front-end development without the need for numerous third-party solutions. This approach ensures uniformity in project architecture, promotes adherence to unified programming standards, and simplifies the maintenance of complex information systems.

The history of the modern version of Angular began in 2016, when Google developers decided to completely redesign the AngularJS architecture. According to company specialists, this decision was driven by limitations of the previous version that became apparent when developing large corporate projects. AngularJS relied on a two-way data binding mechanism, which significantly reduced application performance as the number of components increased.

The new version of Angular was developed virtually from scratch. The architecture is based on a component model, which ensures high code modularity and the reusability of individual interface elements. According to this concept, each application is a collection of independent components, each responsible for performing its own functional task. This approach significantly simplifies software maintenance, since changes to one component have virtually no impact on the rest of the application.

As researchers note, component architecture helps reduce code coupling and improves its readability. Each Angular component includes several interconnected elements: a user interface template (Template), program logic (Component Class), a stylesheet (CSS or SCSS), and metadata describing the component's operating parameters. This separation of responsibilities allows for development to be organized according to object-oriented programming principles and significantly facilitates team collaboration [8].

One of Angular's key advantages is its mandatory use of TypeScript. Unlike traditional JavaScript, TypeScript supports strong typing, interfaces, enumerations, access modifiers, and other features typical of classical programming languages. This allows a significant number of errors to be detected at the compilation stage, increasing software reliability and reducing the likelihood of critical errors during operation.

According to many authors, TypeScript is particularly effective when developing large corporate projects where dozens of developers work on a single application simultaneously. Strong typing makes data structures easier to understand, accelerates code refactoring, and improves collaboration between developers.

Another fundamental feature of Angular is its use of dependency injection. This mechanism automatically creates necessary objects and passes them to components without requiring direct control over their creation. This significantly reduces the coupling between individual application modules and increases code reuse.

For example, a service responsible for interaction with the server is created automatically by the system and can be used simultaneously by several components. Furthermore, changing the service's internal implementation requires virtually no modification to the components that utilize its functionality. As experience in developing large information systems shows, this approach significantly simplifies software maintenance and improves its scalability.

A key feature of Angular's architecture is its modular organization. In early versions of the platform, the core structural element was the module (NgModule), which united interconnected components, services, directives, and data conversion channels (Pipes). This approach ensured the logical separation of application functionality and facilitated a more streamlined organization of the code.

However, further development of Angular led to the introduction of so-called standalone components, which allow applications to be built without traditional modules. According to Google developers, this solution significantly reduces the amount of boilerplate code, simplifies the project structure, and makes the development process more understandable. While support for classic modules remains, most new projects are gradually moving toward a standalone architecture.

One of the most important elements of Angular is the change detection system. The primary purpose of this mechanism is to automatically update the user interface after application data changes. Every time an event occurs, such as receiving a server response, changing a variable value, or user interaction with the interface, Angular analyzes components and determines whether the displayed information needs to be updated. In earlier versions of the platform, this mechanism was criticized for its relatively high computational load. As the number of components increased, so did the number of checks performed, which could negatively impact the performance of large applications. To address this issue, various optimization strategies were implemented, including the use of OnPush mode, which allows checking only those components whose data has actually changed.

A key component of Angular is its built-in routing system (Angular Router). Almost any modern web application consists of multiple pages or logical sections, between which the user constantly switches. Angular provides a ready-made routing mechanism that supports nested routes, lazy loading of modules, route guards, dynamic navigation, and data prefetching.

Lazy loading is especially important when developing large enterprise systems. This mechanism allows loading only those parts of the application that are actually needed by the user at the current moment. This significantly reduces the amount of code initially downloaded, shortens application startup time, and reduces the load on the client device.

Another advantage of Angular is its built-in form tools. The framework supports two approaches to organizing forms: template-driven forms and reactive forms. The former is designed for developing relatively simple user interfaces, while reactive forms are used to implement complex business logic, including dynamic validation, runtime modification of the form structure, and interaction with the server-side application.

Angular is a comprehensive platform for creating modern web applications, primarily focused on developing large-scale enterprise information systems. Its architecture is based on the principles of

modularity, strong typing, dependency injection, and reactive programming, ensuring high scalability, reliability, and maintainability. However, the large number of built-in mechanisms makes Angular more complex to learn than some alternative solutions. However, this complexity is one of the key factors in its popularity for large-scale projects.

React Architectural Features. React is one of the most popular technologies in modern front-end development, used to create dynamic user interfaces of varying complexity. Unlike Angular, React is not a full-fledged framework. Its developers position React as a library for building interfaces, the primary purpose of which is to efficiently display data and update the user interface when the application state changes. This approach gives developers significantly greater freedom in designing the software architecture and choosing additional tools.

React development was started by engineer Jordan Walke at Facebook, now Meta. The first version of the library was released in 2013 and was initially used exclusively in internal company projects. The primary motivation for creating this new technology was the rapidly increasing complexity of social media interfaces. Existing development tools at the time did not provide the necessary performance when constantly changing a large number of page elements. As a result, a fundamentally new model for updating user interfaces was proposed, based on the use of a virtual DOM.

The emergence of React had a significant impact on the subsequent development of the entire web development industry. Many architectural solutions first implemented in this library were subsequently adopted by other modern frameworks. According to experts, React became one of the technologies that defined the development of component-based programming in client-side development.

The component-based approach is the foundation of React's architecture. The application interface is viewed as a collection of independent components, each responsible for displaying a specific part of the user interface. Components can contain their own logic, state, event handlers, and visual representation. This allows for the reuse of the same element across different parts of the application without the need for code duplication.

As industrial development practice shows, component reuse significantly reduces project delivery time and reduces the incidence of software errors. When a component's internal logic changes, all its references are automatically updated, significantly simplifying software maintenance.

A key feature of React is its use of a declarative programming model. The developer describes the desired state of the user interface, while the library automatically determines the exact changes to be made to the browser document structure. This approach significantly simplifies the development of complex interfaces and makes the code more understandable.

Unlike traditional imperative programming, the developer doesn't need to manually search for page elements, change their properties, or track the order of numerous operations. React automatically analyzes component state and synchronizes the UI with the latest application data.

One of React's most well-known mechanisms is the Virtual DOM. This technology is a software model of the page's element tree, existing exclusively in RAM. When data changes, a new version of the virtual tree is created and compared with the previous one. The minimum number of changes required to synchronize the virtual tree with the browser's real DOM is then determined.

As researchers note, direct DOM modification operations are among the most resource-intensive in a browser. Using a virtual DOM significantly reduces their number, thereby improving application

performance. This advantage is especially noticeable when working with large tables, lists, interactive control panels, and other interfaces containing a large number of simultaneously changing elements.

It's worth noting that the widespread belief that the Virtual DOM is unconditionally superior to other interface update mechanisms is somewhat oversimplified. Modern versions of Angular also use efficient change detection algorithms, and the development of Signals technology has significantly reduced the performance gap between the two platforms. Nevertheless, the Virtual DOM remains one of React's most recognizable features.

The concept of component state is crucial to the React architecture. State is the collection of data that determines the current rendering of a component. When the state changes, React automatically triggers a re-render of the corresponding component and all dependent interface elements.

Built-in library mechanisms are used to store local state, while global application state is typically managed using specialized solutions. For many years, Redux was the most common tool. However, the continued evolution of React has led to the emergence of a large number of alternative libraries, including Zustand, MobX, Recoil, Jotai, and others. Modern projects increasingly utilize the Context API in conjunction with custom hooks, eliminating the need for complex external libraries when implementing relatively small applications.

One of the most significant milestones in React's development was the introduction of functional components and the Hooks mechanism. Before this technology, JavaScript classes were the primary way to create components. Despite its extensive functionality, the class-based approach entailed a significant amount of boilerplate code and made it difficult to reuse logic across different components. The advent of Hooks has eliminated the need for classes in almost all new projects. Using built-in functions, developers can manage component state, perform side operations, access the application lifecycle, and reuse program logic without significantly complicating the architecture.

The most commonly used hooks are useState, useEffect, useMemo, useCallback, useRef, and useContext. Using them together can significantly reduce the amount of code and improve the readability of an application. Furthermore, developers can create their own custom hooks that combine repetitive algorithms into independent software modules.

Another unique feature of React is its use of JSX syntax. JSX isn't technically a standalone programming language. It's a special extension of JavaScript syntax that allows you to describe the structure of a user interface directly within your code. Once compiled, JSX is automatically converted into standard React function calls.

Initially, combining application logic and display templates generated debate among specialists. However, over time, this approach has become widespread due to its high code readability and the ease of component maintenance. Experience has shown that colocating logic and user interface significantly facilitates understanding of the application structure.

It's worth emphasizing that React offers virtually no restrictions to the developer's choice of project architecture. Unlike Angular, the library doesn't include built-in routing, state management, server interaction, forms, or international localization. All these tasks are accomplished through the use of additional libraries.

As many researchers have noted, this flexibility is both React's main advantage and disadvantage. On the one hand, a development team can independently select the optimal set of tools for a specific project. On the other hand, the lack of a unified standard means that applications created by different

teams can differ significantly in their internal organization, which sometimes complicates software maintenance [8].

Modern versions of React are actively developing to improve performance. One of the most important innovations is Concurrent Rendering, which distributes the computational load and improves UI responsiveness even when performing resource-intensive operations. Suspense and Server Components mechanisms have been further developed, allowing for more efficient data loading and server-side page rendering.

In recent years, special attention has been paid to optimizing the initial load of web applications. As a result, specialized frameworks built on React have become widespread. The most well-known of these is Next.js, which provides server-side rendering, static page generation, image optimization, routing, and a number of other features. Using such solutions can significantly improve the performance of web applications and improve their indexing by search engines.

Along with web development, React is widely used in mobile app development. Thanks to React Native technology, developers can use a single programming language to create software for both Android and iOS operating systems. Despite certain architectural differences between React and React Native, using a unified ecosystem significantly reduces the costs of developing and maintaining mobile solutions.

A comparative analysis of Angular and React in web application development. Angular and React have remained the most popular technologies for developing modern web applications in recent years. Despite their common scope, they are based on different software design concepts and offer developers different approaches to architecture organization, component interaction, state management, and performance optimization. Therefore, choosing between them is not a one-size-fits-all approach and should be tailored to the specific needs of a specific project.

One of the key differences between the technologies under consideration is their conceptual model. Angular is a full-fledged framework containing virtually all the necessary tools for creating modern web applications. It includes built-in tools for routing, dependency injection, HTTP request handling, form management, testing, internationalization, and project build. This provides developers with a ready-made software platform within which most architectural decisions are already defined.

React, by contrast, is a library whose primary purpose is to build the user interface. All other functionality is implemented through third-party libraries. This approach significantly increases development flexibility, as it allows the application architecture to be tailored to the specific project's requirements. At the same time, developers are free to make decisions regarding routing, state management, data exchange with the server, and other design aspects.

An equally important criterion is the organization of the application architecture. Angular is inherently focused on strictly structured code. Each component has a clearly defined place in the overall project architecture, interacts with services through dependency injection, and adheres to uniform application organizational principles. This standardization significantly simplifies the maintenance of large information systems, as any developer familiar with Angular can quickly grasp the project structure, regardless of the team that created it.

React has no such restrictions. The application architecture is entirely determined by the developers. This provides significant freedom, but also leads to a wide variety of different approaches to organizing

the same software product. As a result, projects implemented by different teams can differ significantly from each other, despite using the same technology.

Particular attention should be paid to the performance of web applications. For many years, React was considered to provide faster performance due to its use of the Virtual DOM. Indeed, this mechanism minimizes the number of UI updates and efficiently re-renders components. React's advantage is particularly noticeable in applications that intensively update a large number of interface elements, such as analytics systems, financial dashboards, interactive maps, and data editors.

However, modern versions of Angular have significantly reduced performance differences. The implementation of Signals technology, improved Change Detection, the use of the OnPush strategy, and optimization of the compilation process have significantly reduced the amount of unnecessary calculations. As a result, in many practical scenarios, the performance differences are minimal and practically unnoticeable to the end user.

Therefore, the claim of React's unconditional superiority in performance cannot be considered universally valid. The actual performance of an application is determined not only by the technology used, but also by the quality of the architecture, the skills of the developers, and the proper implementation of the business logic.

Another important comparison criterion is software scalability. Angular was created primarily as a tool for developing large corporate information systems. Thanks to its strict architecture, mandatory use of TypeScript, built-in dependency injection, and unified code organization rules, maintaining projects with a large number of developers becomes significantly easier.

As experts note, long-term software maintenance is particularly important when working on corporate systems. In such environments, the existence of unified architectural standards reduces the likelihood of technical debt and facilitates the integration of new specialists into the development team.

React is also successfully used in large projects, but achieving a similar level of structure requires additional architectural decisions. Developers must independently select state management libraries, determine project directory organization principles, component naming standards, and rules for interaction between individual application modules.

Of particular interest is the comparison of approaches to application state management. In Angular, a significant portion of component interaction is implemented through services and reactive programming. Using RxJS allows for the organization of complex asynchronous processes, efficient data flow management, and minimization of errors during concurrent operations.

React takes a different approach. Local state is handled extensively using built-in hooks, while global state is managed through the Context API or specialized libraries. This approach provides a highly flexible architecture but requires more careful design of information flows within an application.

In terms of ease of learning, researchers' opinions vary significantly. Initially, React is generally perceived as a simpler tool due to its relatively few required concepts. To create their first applications, a developer only needs to master components, JSX, props, state, and a few basic hooks.

Angular requires significantly more initial knowledge. Developers must understand the principles of TypeScript, dependency injection, routing, reactive programming, services, directives, pipes, modular architecture, and other platform elements. For this reason, the initial learning curve for Angular is typically longer.

However, as a project's complexity increases, the situation gradually changes. Thanks to its strict architecture, maintaining large Angular applications is often easier than maintaining React projects, whose architecture was developed without uniform standards. Thus, Angular's complexity is primarily apparent during the learning curve, while its advantages become noticeable when developing large-scale information systems.

An equally important criterion is the quality of development tools. Angular provides developers with a unified set of officially supported tools, including the Angular CLI, a built-in HTTP client, a router, a testing system, and application build optimization tools. This approach significantly simplifies the start of development and reduces the likelihood of incompatibilities between different libraries.

The situation is different in the React ecosystem. The vast number of available libraries gives developers virtually unlimited choice. However, this diversity necessitates constant analysis of their compatibility, relevance, and long-term support. Without uniform standards, there's a risk of using libraries whose development may subsequently be abandoned.

Significant differences are also observed in terms of information security. Angular contains built-in protection mechanisms against common web threats. Automatic processing of potentially dangerous HTML content, strict template handling, and a sophisticated validation system reduce the likelihood of vulnerabilities associated with malicious code injection.

In React, security issues are more dependent on the developer's skills. Despite built-in data escaping, the library provides more opportunities for direct interaction with HTML content, which requires stricter adherence to secure coding requirements. Therefore, when developing mission-critical information systems, adhering to internal coding standards and conducting regular security audits is especially important.

Choosing a development technology is one of the most critical stages of software design. Mistakes made early in the development of an information system's architecture can significantly increase project maintenance costs, hinder the implementation of new functionality, and necessitate a complete redesign of individual software modules. Therefore, the choice between Angular and React should be based not only on the technology's popularity but also on an analysis of the specifics of the subject area, customer requirements, the composition of the development team, and the prospects for future development of the software product.

As practice in the modern software industry shows, Angular is most effectively used in the development of large-scale corporate information systems. This category includes banking applications, government information platforms, healthcare systems, corporate CRM and ERP systems, educational portals, industrial process management systems, and other projects characterized by complex business logic and long life cycles.

One of the reasons for Angular's popularity in corporate environments is its high degree of development standardization. Nearly all key architectural elements are already defined by the platform's developers, allowing different teams to use consistent approaches to code organization. This significantly simplifies project maintenance over many years and reduces the onboarding time for new developers.

It's worth noting that many international companies view the mandatory use of TypeScript as a key factor in increasing the reliability of their software products. For this reason, Angular is often the

preferred choice for organizations with high demands on the quality of the architecture and security of the information systems they develop.

React, in turn, has become widely adopted by companies focused on rapidly creating digital products and continually introducing new functionality. Startups, online stores, social platforms, media services, marketing websites, and various cloud applications often use React due to its highly flexible architecture and relatively low entry barrier for developers.

Experience shows that using React can significantly reduce the time it takes to create a minimum viable product. By enabling developers to include only the necessary libraries, they can create an architecture that best suits the specific project's requirements without adding unnecessary software modules.

React is particularly effective when developing interfaces with numerous interactive elements. Online builders, analytics systems, dashboards, financial services, document editors, and collaboration platforms are characterized by constantly changing user interface states. In such environments, using the Virtual DOM ensures high-speed updates of individual components and contributes to increased application responsiveness.

Another factor influencing technology choice is the composition of the development team. If an organization has specialists experienced in industrial-scale development using Angular, switching to another platform may require additional time spent training staff and adapting existing development processes. A similar situation applies to React.

According to IT project management experts, the cost of owning a software product is determined not only by the technical characteristics of the technology used, but also by the costs of maintenance, upgrades, and employee training. Therefore, when choosing between Angular and React, it is important to consider not only the platform's functionality but also the development team's accumulated experience [9].

An important aspect is the ability to maintain software over the long term. Modern corporate information systems are used for many years, regularly upgraded, and adapted to changing business requirements. In such circumstances, the stability of the application architecture is of paramount importance.

Angular is designed specifically for this type of use case. The use of consistent architectural principles ensures a highly predictable project structure, even years into development. As a result, software maintenance costs gradually decrease.

React is also successfully used in long-term projects, but the quality of maintenance is largely determined by the discipline of the development team. Without consistent internal standards, the code can gradually become more complex, leading to the accumulation of technical debt and reducing the ease of further development.

An equally important criterion is integration with modern cloud technologies. Today, most web applications operate as part of distributed information systems interacting via REST APIs, GraphQL, WebSockets, and microservice architectures. Both Angular and React offer extensive integration capabilities with such technologies, but they implement them in different ways.

Angular includes built-in HTTP request handling tools, a request interceptor mechanism, a dependency injection system, and advanced error handling. This allows interactions between the client and server parts of an application to be organized within a unified architectural model.

React gives developers the freedom to choose the most suitable libraries for working with external services. This approach provides greater flexibility but requires more careful design of the data exchange architecture.

Search engine optimization requirements for web applications should also be considered. For a long time, single-page applications (SPAs) were inferior to traditional websites in terms of search engine indexing. This issue has now been largely resolved thanks to advances in server-side rendering technologies.

Angular utilizes the Angular Universal engine, which enables HTML pages to be generated on the server before being delivered to the user's browser. In the React ecosystem, similar tasks are most effectively accomplished using the Next.js framework, which supports server-side rendering, static page generation, and hybrid data display models. As a result, both technologies can provide a high level of search engine optimization, provided the project architecture is properly organized.

Another practical aspect is mobile development. Angular can be used in conjunction with the Ionic and NativeScript frameworks, allowing for the creation of cross-platform mobile apps. React, in turn, is actively developing thanks to React Native, which has become widely used in mobile software development for Android and iOS.

As practice shows, React Native has a more developed ecosystem and a larger number of ready-made components, which makes React an attractive choice for organizations planning to simultaneously develop web and mobile versions of a software product.

In recent years, integration with artificial intelligence, machine learning, and generative language models has become increasingly important. Modern web applications increasingly utilize intelligent chatbots, automated data analysis systems, recommendation services, and natural language processing tools. Both Angular and React successfully integrate with such solutions through APIs from various cloud platforms. However, due to its flexible architecture, React is somewhat more commonly used for creating innovative digital products that incorporate AI technologies.

Thus, the results of the analysis indicate that the choice between Angular and React should be determined by a comprehensive assessment of the specific project's specifics. Angular is most appropriate for developing large corporate information systems that require a rigorous architecture, long-term maintenance, and a high degree of standardization. React, on the other hand, demonstrates high effectiveness in creating dynamic web applications, startups, interactive user interfaces, and projects focused on the rapid implementation of new functionality. The lack of a one-size-fits-all solution confirms the need for a customized approach to selecting a technology platform, taking into account the technical, economic, and organizational factors of software development.

To provide a visual comparison of the main characteristics of Angular and React, the results of the comparative analysis are presented in Table 1.

Table 1 - Comparative characteristics of Angular and React based on key criteria

Criterion	Angular	React
Type of technology	A full-fledged framework	User Interface Library
The main development language	TypeScript	JavaScript / TypeScript
Architectural approach	Strictly regulated component architecture	Flexible component architecture
Performance of complex corporate projects	High	High
Speed of initial development	Average	High

The data in Table 1 demonstrates that Angular and React have different architectural concepts. Angular focuses on a unified development framework and includes a built-in toolchain, while React offers developers significantly greater freedom in architectural design choices. While both technologies deliver high performance for modern web applications, they differ in terms of learning curve and development process organization.

Current Trends in Angular and React Development. Modern web development is evolving at an extremely rapid pace, driven by ever-increasing demands on the performance, security, and functionality of digital services. In this context, Angular and React continue to rapidly improve, adapting to new technological challenges. Developers of both platforms strive to improve application performance, simplify the development process, and ensure compatibility with modern architectural approaches. An analysis of recent trends suggests that the competition between Angular and React drives the continued development of both technologies and positively impacts the quality of the software they create.

One of the most noticeable trends of recent years has been the shift toward more efficient reactive programming mechanisms. While Angular previously relied heavily on the RxJS library as its primary tool for managing asynchronous processes, modern versions of the platform are gradually developing the Signals mechanism. Its implementation has been one of the most significant changes to Angular's architecture in recent years.

Signals enable UI updates based on explicitly tracking dependencies between data and components. Unlike traditional change detection, which sequentially checks a large number of components, Signals triggers updates only for those elements that actually depend on the changed data. This approach reduces computational overhead, improves performance, and makes application behavior more predictable.

As experts note, the implementation of Signals reflects a general trend in front-end technology development aimed at reducing unnecessary computation and more efficient use of client device resources. This makes Angular more competitive with other modern solutions that use reactive data processing models [10].

React also continues to actively develop its own architecture. One key area of focus has been the improvement of the Concurrent Rendering mechanism, which allows for more efficient allocation of the browser's computing resources. Unlike traditional sequential execution of operations, the new mechanism can suspend execution of lower-priority tasks and allocate resources to actions directly related to user interaction with the interface.

The practical value of this technology is particularly significant in the development of complex information systems that involve a large number of concurrent operations. For example, when working with analytical dashboards, interactive maps, document editors, or large-scale data visualization systems, the user continues to interact with the interface without noticeable delays, even while resource-intensive calculations are being performed.

The spread of cloud technologies is significantly impacting the development of front-end development. More and more modern web applications operate within distributed information systems, interacting with numerous external services via APIs. Consequently, Angular and React are focusing on integration with cloud platforms, microservice architectures, and containerized runtime environments.

An equally important trend is the rapid development of artificial intelligence technologies. Modern web applications increasingly utilize intelligent assistants, automatic text generation systems, machine learning algorithms, and natural language processing tools. In most cases, such functionality is implemented through interaction between the client application and external artificial intelligence services.

Angular and React offer extensive integration capabilities with similar platforms thanks to their use of modern APIs. However, experience shows that React is more often used in the development of innovative digital products involving generative artificial intelligence. This is due to its highly flexible architecture and the large number of specialized libraries focused on integration with modern cloud services.

Another important trend is the development of microfrontend architecture. In large organizations, individual parts of the user interface are increasingly being developed by independent teams and combined into a single application only at the deployment stage. This approach allows for faster development, reduced coupling between individual modules, and increased independence between teams.

Angular provides built-in tools for organizing large modular projects, making it a convenient tool for implementing microfrontends in enterprise environments. React is also actively used in building such systems due to its highly modular components and the ability to independently deploy individual application parts.

It's also worth noting the changing approaches to building web applications. Modern tools strive to minimize compilation time and speed up the development process. Traditional build systems are increasingly being replaced by new, high-performance tools that ensure virtually instant application updates after changes to the code. Angular and React are gradually adapting to these technologies, which has a positive impact on the productivity of the development environment and the ease of use for programmers.

Much attention is paid to optimizing the final size of applications. Modern mechanisms for code splitting, lazy module loading, automatic removal of unused dependencies, and intelligent build optimization significantly reduce the amount of data transferred to the user. This is especially relevant

given the widespread use of mobile internet and the need to support devices with limited computing resources.

Another persistent trend is the development of automated testing tools. The increasing complexity of software products necessitates the earliest possible detection of errors. Angular traditionally includes a well-developed testing infrastructure, while the React ecosystem offers a large number of specialized tools for unit, integration, and functional testing. In both cases, automated quality assurance is becoming a mandatory element of the modern development process.

The development of a professional community around both technologies deserves special attention. Angular and React boast vast ecosystems, including thousands of libraries, tools, educational materials, and ready-made components. Regular documentation updates, active developer participation in open-source development, and broad support from the global community contribute to the continued adoption of these technologies.

To quantitatively assess the advantages of the technologies under consideration, they were compared based on the most significant characteristics (Table 2).

Table 2 – Expert assessment of Angular and React based on key indicators

Indicator	Angular	React
Performance	4.8	4.9
Scalability	5.0	4.7
Ease of learning	3.8	4.8
Flexibility of architecture	4.4	5.0
MVP development speed	4.2	4.9

The results presented in Table 2 demonstrate that Angular offers advantages for developing large, scalable information systems due to its high degree of architectural standardization. React, on the other hand, scores higher in terms of architectural flexibility, speed of creating a minimum viable product, and ease of initial learning. These results support the conclusion that the choice of technology should be determined by the specifics of the web application being developed and the requirements of the specific project.

Conclusion

The study showed that Angular and React are mature and technologically advanced solutions, widely used in creating modern web applications of varying complexity. Despite their common scope, these technologies are based on different architectural principles, which determines the specifics of their use in specific development environments.

The study found that Angular is a full-featured framework, incorporating virtually all the necessary tools for developing client-side applications. Its use of component architecture, strong TypeScript typing, built-in dependency injection, advanced routing mechanisms, reactive programming, and

modern optimization tools makes Angular an effective tool for developing large-scale enterprise information systems. The high degree of architectural standardization helps reduce software maintenance costs and facilitates collaborative work on large-scale projects.

An analysis of React's architecture revealed that this library is primarily focused on building high-performance user interfaces. The use of a virtual DOM, a declarative programming model, functional components, and the Hooks mechanism ensures high development speed, flexible architectural solutions, and the ability to effectively reuse code. Thanks to its modular approach, React is successfully used in creating interactive web applications, digital platforms, online stores, analytics systems, and innovative services.

A study of current development trends confirmed that both technologies continue to rapidly improve. Angular is evolving toward increased reactivity and performance through the implementation of Signals and optimization of its change detection mechanism. React is improving its server-side rendering, concurrent task execution, and integration with modern cloud platforms. This demonstrates the high adaptability of these technologies to the new demands of the digital economy.

The choice between Angular and React should be determined not by technology popularity, but by the specific project's specifics, customer requirements, the development team's expertise, the required level of scalability, and the software's future development prospects. A comprehensive analysis of technical, organizational, and economic factors allows for the selection of the most effective technological platform and the creation of reliable, high-performance, and maintainable web applications.

References

1. Baidybekov A. A., Gilvanov R. G., Molodkin I. A. MODERN FRAMEWORKS FOR DEVELOPING WEB APPLICATIONS // Intelligent technologies in transport. 2020. No. 4 (24).

2. Baynov Artem Maratovich, Krivonogova Anastasia Evgenievna, Nikolaev Alexey Sergeevich, Bogomolova Olga Igorevna Review of modern frameworks and tools used for developing web applications // Science without borders. 2020. No. 1 (41). URL: https://cyberleninka.ru/article/n/obzor-sovremennyh-freymvorkov-i-instrumetov-ispolzuemyh-dlya-razrabotki-web-prilozheniy

3. Bondarenko Sergey Olegovich Modern interactive web applications - building a user interface with React // Bulletin of science and education. 2018. No. 5 (41). URL: https://cyberleninka.ru/article/n/sovremennye-interaktivnye-web-prilozheniya-postroenie-polzovatelskogo-interfeysa-s-react

4. Danilin Dmitry Andreevich, Zinoviev Yaroslav Vyacheslavovich, Kuzmin Kirill Maksimovich Analysis of web programming technologies for creating modules for visualization and unloading data from information systems // PenzSU Bulletin. 2019. No. 3 (27). URL: https://cyberleninka.ru/article/n/analiz-tehnologiy-veb-programmirovaniya-dlya-sozdaniya-moduley-vizualizatsii-i-vygruzki-dannyh-informatsionnyh-sistem

5. Web Frameworks: An Introduction for Beginners. — August 2, 2018 / Translated from English // Tproger — a typical programmer. URL: http://tproger.ru/translations/web-frameworkshow-to-get-started

6. Frane, B. HTML5 and CSS3. Website development for

any browsers and devices. Responsive Web Design with HTML5 and CSS3. Second Edition / Translated from English by N. Vilchinsky. — St. Petersburg
7. Danjou, D. The Python Way. Black Belt Advice on Deployment, Scalability, Testing, and More = Serious Python. Black-Belt Advice on Deployment, Scalability, Testing, and More / Translated from English by P. Kovaleva. — St. Petersburg: Piter, 2020. — 256 p. — (Programmer's Library).
8. Zend Framework 3 // Zend Framework - Documentation. URL: http://framework.zend.com/learn
9. Reisig, D. JavaScript. Professional programming techniques = Pro JavaScript™ Techniques / Translated from English by N. Vilchinsky. - St. Petersburg: Piter, 2008. - 352 p. - (Programmer's Library).
10. Ilyin Dmitry Yuryevich, Nikulchev Evgeny Vitalievich, Kolyasnikov Pavel Vladimirovich Selection of technological solutions for the development of software for distributed information systems // Modern information technologies and IT education. 2018. No. 2. URL: https://cyberleninka.ru/article/n/vybor-tehnologicheskih-resheniy-dlya-razrabotki-programmnogo-obespecheniya-raspredelennyh-informatsionnyh-sistem