

Best Practices for Web Development

by Sophia



WHAT'S COVERED

In this lesson, you will learn about the twelve best practices for web developers. These best practices will help you address the complexities of web development by working more efficiently and producing more consistent output.

Specifically, this lesson will cover the following:

1. [Complexities of Web Development](#)
2. [Twelve Web Development Best Practices](#)

1. Complexities of Web Development

The world of web development has a wide range of technologies, languages, and facets that can make any developer's work life hectic, stressful, and even potentially unsuccessful. You may have multiple projects going on at once. You may have a client who wants to make a change to a project that you haven't touched in years. You may find that clients are consistently or inconsistently unhappy about different aspects of the work you produce.



BIG IDEA

It is important for all developers to learn and apply a variety of best practices and strategies to help them produce more consistent results and products and reduce complexity and wasted time.

2. Twelve Web Development Best Practices

Let's take a look at the twelve web development best practices.

1. Maintain clean and consistent code. Write clean, readable, and maintainable code. Use comments to help identify the purpose of or dependencies to lines of code. Follow consistent coding conventions, use proper indentation, and adopt a modular approach to organizing your code. Additionally, keep your code sets

organized in their own directories and use a consistent naming convention for your project folders. Poor code organization and management make it difficult to locate and maintain code and ultimately reduce your overall efficiency.

2. Use **semantic tags** to structure page contents. Structure your webpages using semantic HTML elements (e.g., `<header>`, `<nav>`, `<main>`, and `<footer>`) to provide meaning and organization and to improve accessibility. Not only does this help you as the developer to easily organize and locate code specific to certain sections of the page, but it also helps accessibility software in making better decisions on how to present the information to the user.




3. Use **version control** software. Use a version control system such as Git to track changes in your codebase, collaborate with other developers, and easily roll back to previous versions if needed. If you collaborate with other developers, version control software is a must. Version control software keeps an exact log of all the changes made to each and every code file in the project. Furthermore, when multiple users fork a single code file (to fork means to make a copy) to start their own branch and add their own contributions to the project, version control software is able to use the change logs to reconcile the changes from both edited versions and provide a new final merged version. The other major benefit is, of course, being able to roll back to a previous version of the project if needed.

4. Incorporate responsive design. Ensure your websites are responsive and adapt well to different screen sizes and devices. Use CSS media queries or, better yet, take advantage of responsive frameworks like Bootstrap to create a mobile-friendly user experience. If you design a site with responsiveness in mind from the beginning, your site will be better received by viewers on mobile devices. Additionally, doing this from the start will save you extra time and effort at the end making the site responsive.

5. Build efficiencies into your work. One of the best ways to increase your productivity is to become familiar with and make use of frameworks and time-saving tools provided by IDEs. While frameworks like Bootstrap do take some time to learn and understand how to apply effectively, they can absolutely increase your development efficiency by reducing the amount of time spent on the details and intricacies of implementing responsive design, for example. Frameworks can also reduce errors by abstracting away the smaller details, reducing the opportunities for errors. Additionally, IDEs often provide tools and utilities for managing your project in a variety of ways. Visual Studio Code, for example, can provide a pre-formatted HTML page structure, generate Lorem Ipsum, autocomplete code, and even generate reusable CSS rule structures. There is a lot that a good IDE can do for a developer; make sure to explore those features and get into the habit of using them.

6. Conduct cross-browser testing. Test your websites or web applications on multiple browsers (i.e., Chrome, Firefox, Safari, and Edge) to ensure they work consistently across different platforms. Different browser applications are developed by different organizations and thus behave differently when presented with the same code set. Consider using tools like BrowserStack or Sauce Labs to conduct cross-browser testing and to ensure the best compatibility among the popular browsers. This will increase acceptance of your website or application and can avoid costly compatibility issues.

7. Optimize website performance. Optimize and compress images without sacrificing quality. Images can be saved in different file formats, with each format possessing different average file sizes and overall quality. For example, a simple icon image, like the one below, at the time it is saved is 8 KB as a JPEG, 6 KB as a PNG, and only 4 KB as a GIF, all without sacrificing quality.

PNG: 8 KB	JPEG: 6 KB	GIF: 4 KB
		

The following image is an example of an image that has suffered quality loss; thus, the GIF format would not be ideal. Instead, the software used to make the image can use compression techniques to reduce the file size without affecting quality, such as JPEG and PNG. Furthermore, the GIF format, which usually has a smaller file size, did not provide any benefit, but it greatly reduced the quality of the image. Again, the image sizes noted are the size of the image file at the time it was created.

PNG: 168 KB



JPEG: 76 KB



GIF: 87 KB



Use **browser caching** to enable faster subsequent visits. This means incorporating Cache-Control headers into the web server to tell the browser what to cache, when the cache is no longer valid, and when to use the cached version.

Another technique to improve loading performance is to use a technique called lazy loading to defer the loading of noncritical resources. For example, decorative images located in the footer of a page do not need to be loaded with the page itself since they will never be seen on screen until the user scrolls down. As such, these images can use lazy loading, and thus the page will only request the image files once the user scrolls down.

Avoid excessive HTTP requests and optimize database queries. When your website pulls data from a database, it uses an additional HTTP request each time. However, if the data that is initially pulled can be reused to make later decisions or generate views, reuse the data in memory and avoid an additional HTTP request.

Lastly, use tools and services like Minify. Minify is a utility that will compress your CSS, JavaScript, and HTML files to reduce their file sizes and improve loading times. This is done by removing unnecessary white space, comments, as well as repetitive or unused code.

8. Ensure security. Protect against common web vulnerabilities such as **cross-site scripting (XSS)**, **SQL injection**, and **cross-site request forgery (CSRF)** by implementing security measures like **input validation** and **output sanitization**. These types of attacks are common and easily fixed by employing input sterilization techniques. It is a best practice for all web developers to become familiar with these types of attacks, how they work, and how and when we need to protect against them.

Regularly update and patch software, frameworks, and libraries to address security vulnerabilities. These tools offer a lot to developers and users. However, as with any piece of software being added to a project, they can introduce vulnerabilities. When these vulnerabilities are discovered, the developers will update their software and release a patch or updated version. It is a good practice to keep a list of the software, frameworks, and libraries in use and to regularly check for security patches or new versions that address any security concerns.

9. Provide accessibility. Design and develop websites that are accessible to users with disabilities. Become familiar with the Web Content Accessibility Guidelines (WCAG) to ensure the proper use of headings, alt text for images, keyboard accessibility, and more. It may also be worth examining the guidelines of Section 508 of the U.S. Rehabilitation Act in case you ever develop for federal organizations.

10. Maintain documentation. Document your code, APIs, and project setup to facilitate future maintenance and collaboration. Clear documentation helps other developers understand your code and onboard new team members. Documentation also helps developers jump back into a previous project. Configuration and setup documentation is also important to bringing a past project back to life. In some scenarios, you may have been required to use a specific tool set configured in a particular way to facilitate development for a specific client. If that client ever comes back with a new project or changes to an older one, good documentation will save you a lot of time and headache trying to get your system set up exactly the way it was.

11. Carry out **performance monitoring**. When you are responsible for maintaining a site's code set, it is good practice to continuously monitor your website's performance using tools like Google PageSpeed Insights,

Lighthouse, or GTmetrix. Identify bottlenecks or slow-loading assets, and use techniques to reduce the load time.

12. Keep learning. Web development is a rapidly evolving field, and it is important to stay updated with the latest web standards, frameworks, tools, and best practices. Participate in online communities, attend conferences, and follow industry blogs and forums to stay informed and abreast of the latest trends and technologies. For example, to stay up to date with current trends, subscribe to reputable publications such as Forbes Innovation. Dig deeper into computer languages and code using sites like Mozilla Developer Network (MDN), w3schools.com, and TutorialsPoint. You can also take a more structured approach and utilize or subscribe to online learning platforms such as Coursera, LinkedIn Learning, Udemy, edX, and many more. These online learning platforms, some free and some paid, allow you to take structured courses of your choosing and earn certificates and digital badges, and some even offer credits toward college credit. Certificates and digital badges are valuable to you as a professional as they demonstrate accomplished skills and knowledge to employers and increase your value as an employee or independent contractor.

TERMS TO KNOW

Semantic Tags

HTML tags that reinforce the meaning of the information in the webpage as opposed to only defining its position and appearance.

Version Control System

A tool that tracks changes in your codebase and can easily roll back to previous versions if needed.

Browser Caching

A browser process wherein web content is temporarily stored on the local system to improve performance.

Cross-Site Scripting (XSS)

A type of web attack when the attacker inserts malicious code into a site page that victims visit.

SQL Injection

A type of web attack when malicious SQL code is inserted into a web form and then executed by the server.

Cross-Site Request Forgery (CSRF)

A web attack when a malicious attacker tricks the victim into submitting an HTTP request that allows the attacker to perform state-changing operations on behalf of the victim.

Input Validation

A mechanism for verifying that the data values provided are in the expected format.

Output Sanitation

The process of examining the data being returned by a request and removing or encoding data to reduce security risks.

Performance Monitoring

The ongoing process of creating a benchmark of performance and then using current metrics to manage and improve performance deficiencies.



SUMMARY

In this lesson, you learned about some of the **complexities of web development** and were introduced to **twelve best practices for web developers**. These best practices will go a long way in helping you work more efficiently and produce more consistent output.

Source: This Tutorial has been adapted from "The Missing Link: An Introduction to Web Development and Programming " by Michael Mendez. Access for free at <https://open.umn.edu/opentextbooks/textbooks/the-missing-link-an-introduction-to-web-development-and-programming>. License: [Creative Commons attribution: CC BY-NC-SA](#).



TERMS TO KNOW

Browser Caching

A browser process wherein web content is temporarily stored on the local system to improve performance.

Cross-Site Request Forgery (CSRF)

A web attack when a malicious attacker tricks the victim into submitting an HTTP request that allows the attacker to perform state-changing operations on behalf of the victim.

Cross-Site Scripting (XSS)

A type of web attack when the attacker inserts malicious code into a site page that victims visit.

Input Validation

A mechanism for verifying that the data values provided are in the expected format.

Output Sanitation

The process of examining the data being returned by a request and removing or encoding data to reduce security risks.

Performance Monitoring

The ongoing process of creating a benchmark of performance and then using current metrics to manage and improve performance deficiencies.

SQL Injection

A type of web attack when malicious SQL code is inserted into a web form and then executed by the server.

Semantic Tags

HTML tags that reinforce the meaning of the information in the webpage as opposed to only defining its position and appearance.

Version Control System

A tool that tracks changes in your codebase and can easily roll back to previous versions if needed.