1.1 Introduction to Web Development

What is Web Development?

The 1950s were a hobbyist's paradise with magazines such as Mechanix Illustrated and Popular Science showing the do-it-yourselfer how to build a go-kart for the kids and how to soup up a lawnmower with an actual motor!

Sixty years later, we're now firmly entrenched in the age of do-it-yourself tech, where folks indulge their inner geek to engage in various forms of digital tinkering and hacking. The personification of this high-tech hobbyist renaissance is the maker, a modern artisan who lives to create things, rather than merely consume them. Today's makers exhibit a wide range of talents, but the skill most sought-after not only by would-be makers themselves, but by the people who hire them, is web coding and development.

Have you ever visited a website and thought, "Hey, I can do better than that!"? Have you found yourself growing tired of merely reading text and viewing images that someone else has put on the web? Is there something creative in you — stories, images, expertise, opinions — that you want to share with the world? If you answered a resounding "Yes!" to any of these questions, then congratulations: You have everything you need to get started with web coding and development. You have, in short, the makings of a maker.

How the web works

Before you can understand web coding and development, you need to take a step back and understand a bit about how the web itself works. In particular, you need to know what happens behind the scenes when you click a link or type a web page address into your browser. Fortunately, you don't need to be a network engineer to understand this stuff, because you can alone understand the basics without much in the way of jargon. Here's a high-level blow-by-blow of what happens:

You tell the web browser the web page you want to visit .You do that either by clicking a link to the page or by typing the location — known as the uniform resource locator or URL (usually pronounced "you-are-ell," but also sometimes "earl") — into the browser's address bar (see Figure 1-1 One way to get to a web page is to type the URL in the browser's address bar).

The browser decodes the URL. Decoding the URL means two things: First, it checks the prefix of the URL to see what type of resource you're requesting; this is usually http:// or https://, both of which indicate that the resource is a web page. Second, it gets the URL's domain name — the something.com or whatever.org part — and asks the domain name system (DNS) to translate this into a unique location — called the IP (Internet Protocol) address — for the web server that hosts the page (see F
Figure 1-2 The browser extracts the prefix, domain, and the server address from the URL).

The browser contacts the web server and requests the web page. With the web server's unique IP address in hand, the web browser sets up a communications channel with the server and then uses that channel to send along a request for the web page (see Figure 1-3 The browser asks the web server for the web page)

The web server decodes the page request. Decoding the page request involves a number of steps. First, if the web server is shared between multiple user accounts, the server begins by locating the user account that owns the requested page. The server then uses the page address to find the directory that holds the page and the file in which the page code is stored (see Figure 1-4 The server uses the page request to get the account, directory, and filename)

The web server sends the web page file to the web browser (see Figure 1-5 The web server sends the
requested web page file to the browser.).

The web browser decodes the web page file. Decoding the page file means looking for text to display, instructions on how to
display that text, and other resources required by the page, such as images and fonts (see Figure 1-6 The web browser scours the page file to see if it needs anything else from the server).

If the web page requires more resources, the web browser asks the server to pass along those resources (see Figure 1-7 The web browser goes back to the server to ask for the other data needed to display the web page.)

For each of the requested resources, the web server locates the associated file and sends it to the browser (see Figure 1-8 The web server sends the browser the rest of the requested files).

The web browser gathers up all the text, images, and other resources and displays the page in all its digital splendor in the browser's content window (see Figure 1-9 At long last, the web browser displays the web page).

How the web works, take two

Another way to look at this process is to think of the web as a giant mall or shopping center, where each website is a storefront in that mall. When you request a web page from a particular site, the browser takes you into that sites store and asks the clerk for the web page. The clerk goes into the back of the store, locates the page, and hands it to the browser. The browser checks the page and asks for any other needed files, which the clerk retrieves from the back. This process is repeated until the browser has everything it needs, and it then puts all the page pieces together for you, right there in the front of the store.

This metaphor might seem a bit silly, but it serves to introduce yet another metaphor, which itself illustrates one of the most important concepts in web development. In the same way that our website store has a front and a back, so, too, is web development separated into a front end and a back end.

» **Front end:** That part of the web page that the web browser displays in the browser window. That is, it's the page stuff you see and interact with.

» **Back end:** That part of the web page that resides on the web server. That is, it's the page stuff that the server gathers based on the requests it receives from the browser.

As a consumer of web pages, you only ever deal with the front end, and even then you only passively engage with the page by reading its content, looking at its images, or clicking its links or buttons.

However, as a maker of web pages that is, as a web developer your job entails dealing with both the front end and the back end. Moreover, that job includes coding what others see on the front end, coding how the server gathers its data on the back end, and coding the intermediate tasks that tie the two together.

Reference: Web Coding & Development All-in-One For Dummies®
Published by: John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030-5774, www.wiley.com

Terms and Definitions

Here are some basic web development terms you need to know:

- **API**
  - API stands for "Application Program Interface." This is how computers and applications can communicate with one another.
- **Back-End Web Development**
  - Back-End web development is the development of the behind-the-scenes activities that occur when performing any action on a website. It focuses primarily on databases, backend logic, APIs, and Servers. It is also called **server-side development.**
- **Bandwidth**
  - This is the amount of data that can be transmitted over a connection in a given time.
- **Cache**
  - Cache is temporary data storage that helps site speed by storing relevant information on your computer the first time you visit a website. Thanks to cache, when you visit the website again, your computer does not have to reload all the website information as its already saved.
- **Cookies**
  - This is data saved by your web browser on to your computer. It identifies you -- from how often you visit a site, to what parts you visit the most, even your preferences when browsing and more.
- **Classes**

    - An identifier used to define different sections of a website to help target for styling purposes in your CSS file.
- **CSS**
    - CSS stands for “Cascading Style Sheets.” This code that tells the browser how to display your website. This includes global styles for fonts, colors, images, menus, etc.
- **Encryption**
    - Data encryption translates data into another form, or code, so that only people with access to a secret key (formally called a decryption key) or password can read it. Encrypted data is commonly referred to as ciphertext, while unencrypted data is called plaintext.
- **Framework**
    - A framework is a set of software tools that helps people build websites easier. (Examples: Bootstrap, Skeleton)
- **Front End Web Development**
    - Front-End web development is the development of the graphical user interface of a website, through the use of HTML, CSS, and JavaScript, so that users can view and interact with that website. It is also called **client-side development.**
- **FTP**
    - FTP is short for “File Transfer Protocol.” When it comes to websites, this is how files are uploaded to the internet.
- **Full-Stack Web Development**
    - Full-Stack web development is the development of both front-end and back-end portions of a web application.
- **Hosting**
    - Hosting refers to a web server where the files for your website are stored.
- **HTML**
    - Also known as “Hypertext Markup Language,” this is the language that is used to build your website pages and display content like content, images, video, and links on the web.
- **HTTP & HTTPS**
    - Hyper Text Transfer Protocol Secure (HTTPS) is the secure version of HTTP, the protocol over which data is sent between your browser and the website that you are connected to.
    - The 'S' at the end of HTTPS stands for 'Secure.' It means all communications between your browser and the website are encrypted. HTTPS is often used to protect highly confidential online transactions like online banking and online shopping order forms.
- **Javascript**
    - Javascript is a programing language that is commonly used to create interactive effects within web browsers. For example, clicking on a frequently asked question and having the answer appear below.
- **Protocol**
    - A set of rules or procedures for transmitting data between electronic devices, such as computers. In order for computers to exchange information, there must be a

preexisting agreement as to how the information will be structured and how each side will send and receive it.

- **Server**
  - A server is a computer designed to process requests and deliver data to another computer over the internet or a local network.
- **Server Side Scripting**
  - Server Side Scripting is a technique used in development which involves utilizing scripts on a web server that produce a custom response for each user's request to the website.
- **User Interface (UI)**
  - UI is the point of human-computer interaction and communication in a device. This can include display screens, keyboards, a mouse and the appearance of a desktop. It is also the way through which a user interacts with an application or a website.
- **Wireframe**
  - A wireframe is a visual guide used to show the structure of a web page without any design elements.

Web Development Stages

WEBSITE DEVELOPMENT PROCESS: FULL GUIDE IN 7 STEPS

Despite conventional wisdom, the core part of website development and design is not necessary for the coding process. Indeed, such technologies as HTML, CSS, and JavaScript give the web we know its shape and define the way we interact with the information. But what usually stays behind the scenes and, at the same time, remains the crucial part of the website development life cycle are the stages of preliminary information gathering, detailed planning, and post-launch maintenance.
In this article, we'll take a look at how the general website development process may look like. The overall number of development stages usually varies from five to eight, but every time the whole picture stays pretty much the same. Let's choose the average value.

So, here are seven main steps of web development:

1) Information Gathering,

2) Planning,

3) Design,

4) Content Writing and Assembly,

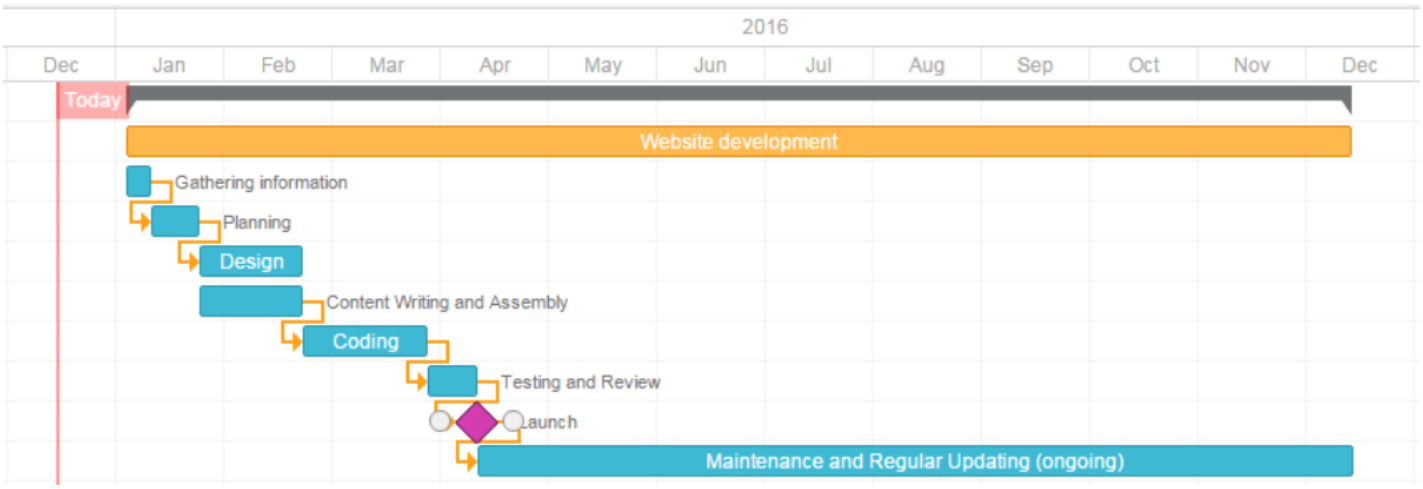
5) Coding,

6) Testing, Review and Launch,

7) Maintenance.

Website Development Timeline

When you think of building a website, your thoughts rotate around two main issues – price and time. These two values depend largely on the size and scope of the project. To outline the whole development process, you can create a website development timeline, adding tasks, and establishing milestones for your project. It is the best way to track your project implementation to make sure you keep up with the deadline.



Website Development Life Cycle

Step 1. Gathering Information: Purpose, Main Goals, and Target Audience

This stage, the stage of discovering and researching, determines how the subsequent steps will look like. The most important task at this point is to get a clear understanding of your future website purposes, the main goals you wish to get, and the target audience you want to attract to your site. Such kind of a website development questionnaire helps to develop the best strategy for further project management.

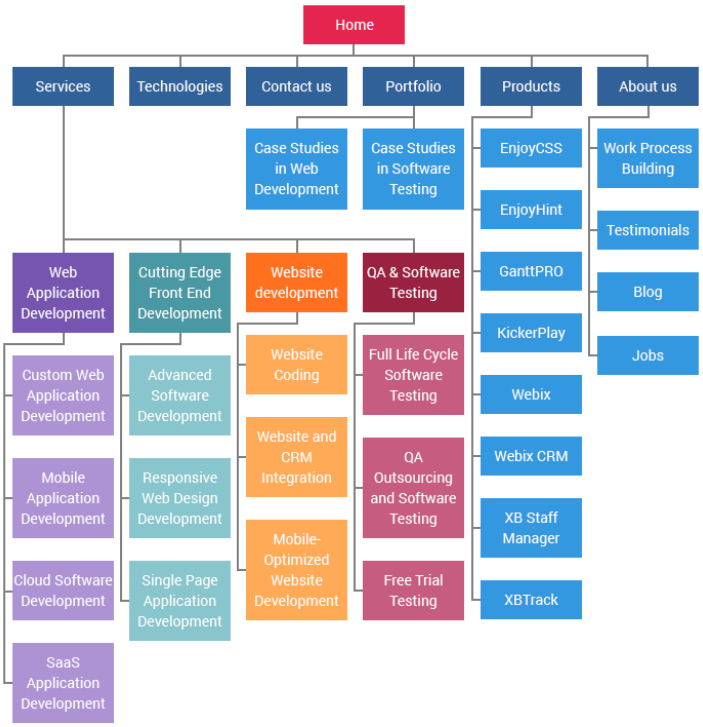
News portal differs from the entertainment websites, and online resources for teenagers look different than sites for adults. Different types of websites provide visitors with different functionality, which means that different technologies should be used according to purposes. A well-described and detailed plan based on this pre-development data can protect you from spending extra resources on solving the unexpected issues such as design changing or adding the functionality that wasn't initially planned.

Estimated time: from 1 to 2 weeks

Step 2. Planning: Sitemap and Wireframe Creation

At this stage of the website development cycle, the developer creates the data that allows a customer to judge how the entire site will look like.

Based on the information that was gathered together in the previous phase, the **sitemap** is created. Here is the sitemap of the XB Software website:



The sitemap should describe the relations between the main areas of your website. Such representation could help understand how usable the final product will be. It can show you the “relationship” between the different pages of a website, so you can judge how easy it will be for the end-user to find the required information or service if he starts from the main page. The main reason behind the sitemap creation is to build a user-friendly and easy to navigate website.

The sitemap allows you to understand how the inner structure of a website looks like but doesn’t describe the user interface. Sometimes, before you start to code or even work on a design, there’s a necessity to get approval from a customer that everything looks fine so you can begin the next phase of developing. In this case, a **wireframe** or **mock-up** is created. A wireframe is a visual representation of the user interface that you’re going to create. But it doesn’t contain any design elements such as colors, logos, etc. It only describes the elements that will be added to the page and their location. It’s artless and cheap in production sketch.

You can use any mockup for this purpose. We used Moqups. Here’s how the wireframe can look like:

The other important thing is select technology stack – programming language, frameworks, CMS that you're going to use.

Estimated time: from 2 to 6 weeks

Step 3. Design: Page Layouts, Review, and Approval Cycle

During the design phase, your website takes shape. All the visual content, such as images, photos, and videos is created at this step. Once again, all the info that was gathered through the first phase is crucial. The customer and target audience must be kept in mind while you work on a design.

The website layout is the result of a designer's work. It can be a graphic sketch or an actual graphic design. The primary function of the layout is to represent the information structure, visualize the content, and demonstrate the basic functionality. Layouts contain colors, logos, images and can give a general understanding of the future product.

After that, the customer can review the layout and send you his feedback. If the client is not sure about some aspects of your design, you should change the layout and send it back to him. This cycle should be repeated until the customer is completely satisfied.

Estimated time: from 4 to 12 weeks

Step 4. Content Writing and Assembly

Content writing and compiling usually overlaps with other stages of website creation, and its role can't be underestimated. At this step, it is necessary to put in writing the very essence you'd like to communicate to the audience of your website and add calls-to-action. Content writing also involves the creation of catching headlines, text editing, writing new text, compiling the existing text, etc., which takes time and effort. As a rule, the client undertakes to provide website content ready to migrate to the site. It is better when all website content is provided before or during website coding.

Estimated time: from 5 to 15 weeks

Step 5. Coding

At this step, you can finally start creating the website itself. Graphic elements that have been designed during the previous stages should be used to create an actual website. Usually, the home page is created first, and then all sub-pages are added, according to the website hierarchy that was previously created in the form of a sitemap. Frameworks and CMS should be implemented to make sure that the server can handle the installation and set-up smoothly.

All static web page elements that were designed during the mock-up and layout creation should be created and tested. Then, special features and interactivity should be added. A

deep understanding of every website development technology that you're going to use is crucial at this phase.

When you use CMS for site creation, you can also install CMS plugins at this step if there's a need. The other important step is SEO (Search Engine Optimization). SEO is the optimization of website elements (e.g., title, description, keyword) that can help your site achieve higher rankings in the search engines. And, once again, valid code is pretty important for SEO.

Estimated time: from 6 to 15 weeks

Step 6. Testing, Review, and Launch

Testing is probably the most routine part of a process. Every single link should be tested to make sure that there are no broken ones among them. You should check every form, every script, run a spell-checking software to find possible typos. Use code validators to check if your code follows the current web standards. Valid code is necessary, for example, if cross-browser compatibility is crucial for you.

After you check and re-check your website, it's time to upload it to a server. An FTP (File Transfer Protocol) software is used for that purpose. After you deployed the files, you should run yet another, final test to be sure that all your files have been installed correctly.

Estimated time: from 2 to 4 weeks

Step 7. Maintenance: Opinion Monitoring and Regular Updating

What's important to remember is that a website is more of a service than a product. It's not enough to "deliver" a website to a user. You should also make sure that everything works fine, and everybody is satisfied and always be prepared to make changes in another case.

The feedback system added to the site will allow you to detect possible problems the end-users face. The highest priority task, in this case, is to fix the problem as fast as you can. If you won't, you may find one day that your users prefer to use another website rather than put up with the inconvenience.

The other important thing is keeping your website up to date. If you use a CMS, regular updates will prevent you from bugs and decrease security risks.

Estimated time: ongoing

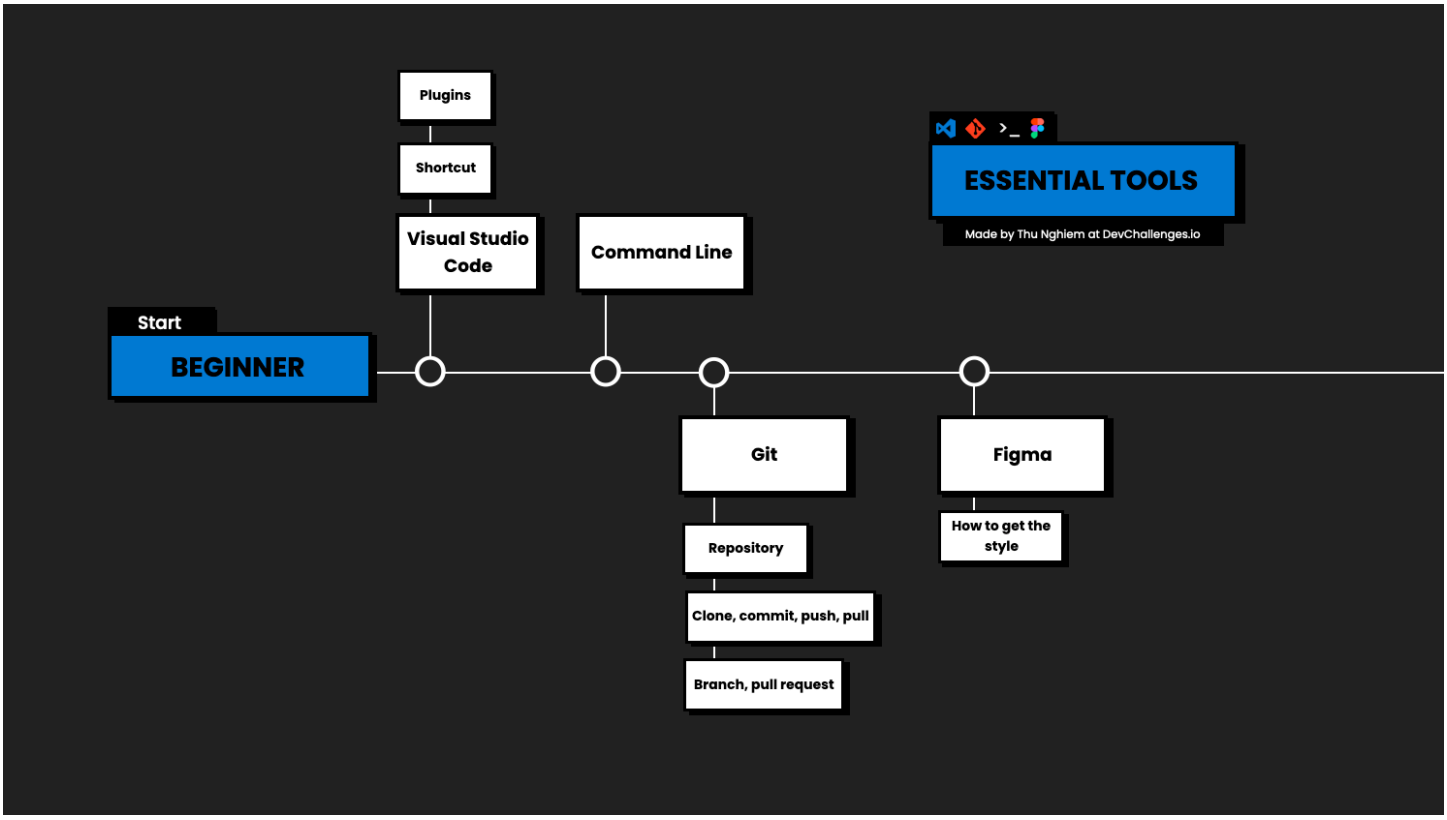
Conclusions

You should always keep in mind that the website development project doesn't start with coding and doesn't end after the day you finally launch your website. The phase of preparation affects all subsequent stages, defining how productive the development process will be. A profound and in-depth discovery of such aspects like age, sex, and interests of your end-user may become the key to success. The post-launch period is rather significant. Your project should be agile and flexible enough to have a possibility to change your website according to users' feedback or spirit of the time. Keeping in mind that there's no such thing as insignificant website development phase will prevent you from unexpected troubles and give you confidence that everything flows as it should, and you have full control over the project.

Reference: Website Development Process: Full Guide in 7 Steps - XB Software

Web Developer Roadmap

Tools You Need to Know to Become a Web Developer



VS Code (or other code editor)

First, you should learn how to use **Visual Studio Code** (Links to an external site.), which is a source code editor. It is a free and powerful tool.

In the beginning, I recommend learning some of the basic shortcuts and installing some of the extensions like ESlint, Prettier, or Live Server.

Here's a free, full-length course (Links to an external site.) on the freeCodeCamp YouTube channel to get you started.

The Command Line

Next, you need to know about the **Command-Line (Links to an external site.)**. You should learn what it is, and some of the basic commands like how to move around directories, how to create a new directory, or how to create a new file.

Here's a great article on how to use Bash (Links to an external site.), the Linux command line. And here's one one that'll help you use the command line more effectively (Links to an external site.).

Version Control - Git and GitHub

No matter what you do, as a developer, you need to know about **Git (Links to an external site.)**. Git is a Version Control system used for tracking changes. It is usually used with **GitHub**, which is a code hosting platform.

In the beginning, learning Git might be overwhelming, therefore you just need to know some of the basics like how to create a new *Repository*, how to *clone* a project, how to make a new *commit*, and how to *pull* and *push* the new changes.

One of the best ways to practice Git is by working in a team. There, you need to know how to create a new *branch*, how to make *pull requests*, and how to resolve *conflicts*.

Here's a great Git and GitHub crash course (Links to an external site.) on the freeCodeCamp YouTube channel to get you going with version control.

Design tools - Figma

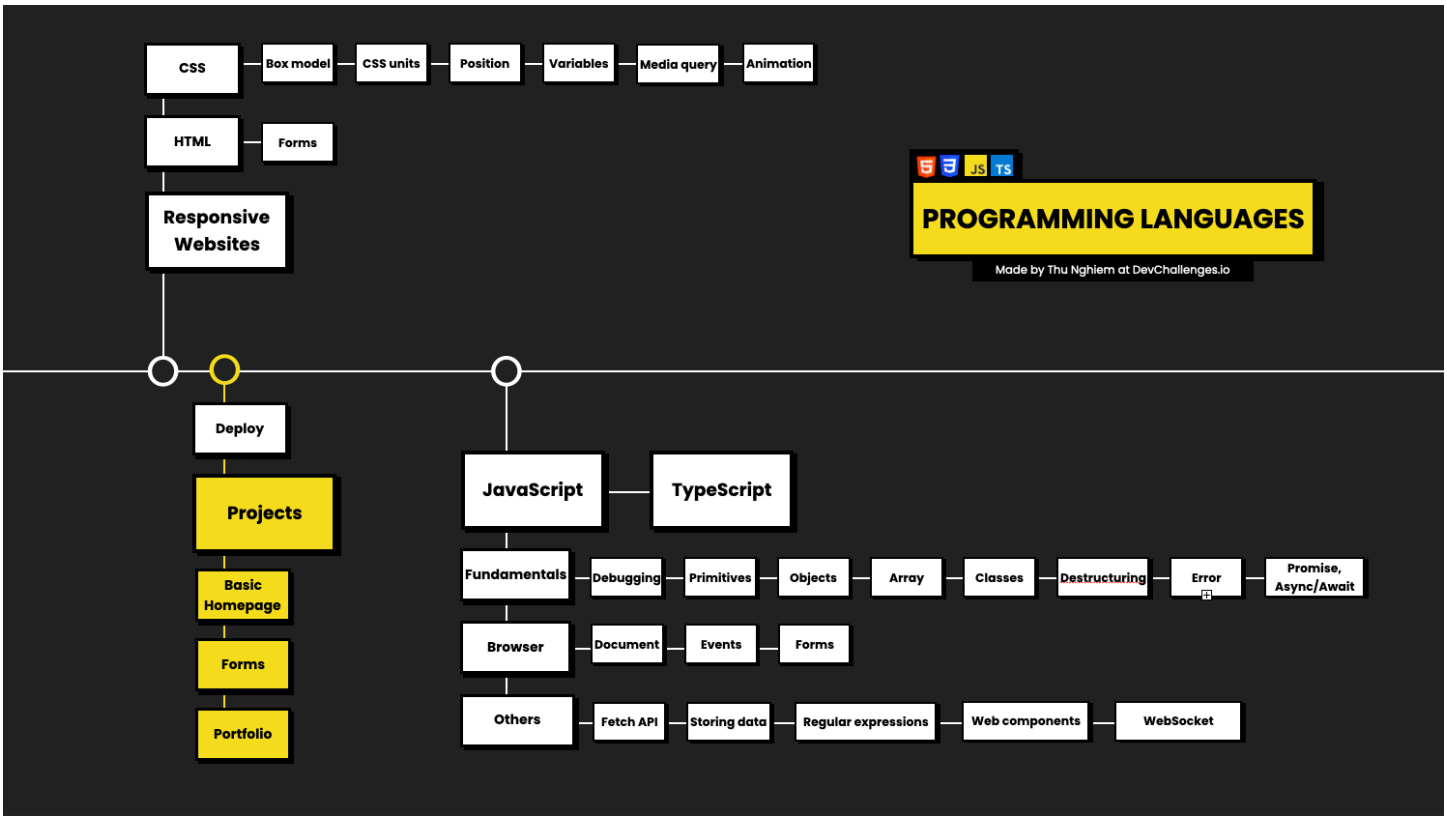
The last tool is **Figma (Links to an external site.)**. Figma is a design tool and is free to use for individuals. But here I want to talk about how to use Figma as a developer.

As a developer, you might get designs from designers on your team. With Figma you can inspect the code of the elements and measure the layout. Therefore, you need to know how to read Figma design, like how to get the color, typography, or spacing right.

Here's an article about various design tools, Figma included (Links to an external site.). And here's a fun tutorial about creating 3D sketches in Figma (Links to an external site.).

Alright, so these are the tools that you need to know of when getting started. **You don't need to know everything**, but be sure that you understand the basics so that you can improve while learning to code.

Programming Languages Web Developers Should Learn



HTML and CSS

Next, let's continue with programming languages. And let's start off by talking about **Responsive Websites**.

Responsive websites are sites that look good and are usable on all devices or screen sizes. You probably know how important it is to build a responsive website, as people use so many different devices these days.

So let's jump to the first two languages that you need to know to build a website: **HTML** and **CSS**.

HTML stands for *Hypertext Markup Language*. It it is used to build the skeleton for your website. HTML is not difficult to learn, but you might want to pay more attention to HTML forms as they will be fundamental in the future.

CSS stands for *Cascading Style Sheets*. This is a markup language but I also consider it to be a programming language. CSS is not necessarily difficult to learn but it's difficult to master.

There are few topics that you want to pay more attention to like:

- **Box model** - how margins, padding, and borders work together.
- **CSS units** - used for expressing lengths (e.g: rem, vh, and vw).
- **Position** - specifies the type of positioning method. It also confuses many people so make sure you spend some time on it.
- **Variables** - or **Custom properties** are entities that can be reused throughout a document. This is my favorite feature in CSS. They make working with CSS so enjoyable and you can create themes with just a few lines of code.
- **Media query** - decides what to show on different screen sizes. They are a key component of responsive design (Links to an external site.).
- **Animation** - lets an element change from one style to another. If you know how to use animation correctly, it makes your site stand out. Otherwise, it will make your website look unprofessional, so be careful.
- **Flexbox, CSS Grid** - used for building Responsive layouts (I forgot to mention this in the video).

Here's a full handbook (Links to an external site.) that'll teach you all the basics of HTML. And here's a complete course on CSS (Links to an external site.) on the freeCodeCamp YouTube channel that'll get you started beautifying your sites.

When you know the basics of HTML and CSS, the next step is to build some basic websites. For example, you can try a *Homepage*, *a Form* like a login page or a checkout page. You can even build a *Portfolio*. You can find example projects on DevChallenges.io (Links to an external site.).

Website deployment

Once you have your website, you need to put it on the internet, so people can see it. Deployment is the process of deploying your code to a hosting platform.

Back in the day it was a lot more difficult to do. But now, it's super easy and you can use tools like GitHub Pages (Links to an external site.), Netlify (Links to an external site.), or Vercel (Links to an external site.).

Here's a full YouTube course (Links to an external site.) on how to get your site online that covers the entire process from start to finish.

JavaScript - Fundamentals

Alright, the next topic is **JavaScript** (Links to an external site.). JavaScript is a popular programming language and is widely used for Web Development, among other things.

You will need to learn some of the basic features of the language like **Data Types (Links to an external site.)**, **Loops (Links to an external site.)**, and **Conditionals (Links to an external site.)**.

Then there are topics that you will want to dive deeper into.

First, we have **debugging.** This is the process of finding and fixing errors. Here's a great in-depth guide to bug squashing (Links to an external site.) to get you started.

Then there are other topics like **Objects**, **Primitives**, and **Arrays**. Especially when working with Arrays, you need to know about **Array methods (Links to an external site.)** as well.

Functions are the main building blocks in your program. So make sure that you don't overlook them.

One of my favorite features in JavaScript is **de-structuring (Links to an external site.)** – it is easy to write and makes the language super powerful.

Like C#, Java, or other programming languages, in modern JavaScript we also have **classes**. These are useful when it comes to Object-Oriented Programming and the SOLID principles (Links to an external site.).

And no matter how good you are with programming, you will have errors in your scripts. This means you'll want to know about **Error handling** as well (Links to an external site.).

Asynchronous programming is important, especially when you need to communicate with the server. So spend some time learning about **Promises** and **Async/Await** (Links to an external site.).

JavaScript - Browser

Let's move on to how JavaScript is used in the browser.

First, you need to know what a *Document Object Model* or **DOM** is. Then you need to know how to get elements, how to change the classes, or how to change the style with JavaScript.

Here's a good introduction to the JS DOM (Links to an external site.), and here's a guide on how to manipulate the DOM (Links to an external site.) (you'll learn by building a project).

You also need to learn about different **User Interface Events (Links to an external site.)** like click, mouse over, mouse down, and so on.

And, you'll also want to pay more attention to **Forms** in JavaScript (Links to an external site.) as they have many events and properties

JavaScript - Other features

The **Fetch API** is an important topic (Links to an external site.). It lets you send network requests to servers. This is useful, for example, when we need to submit a form or get a user's information.

Another important topic is **Storing Data** in the browser (Links to an external site.). Here you need to know what the differences are between Cookies, LocalStorage, and sessionStorage.

Other less important topics when you are just getting started are Regular expressions, Web Components, and Websockets.

Regular expressions (Links to an external site.) are used to search and replace text. **Web Components**, are a new thing but you should totally check them out.

Lastly, we have **Websockets (Links to an external site.)**. They are useful when you need to have continuous data exchanges like in chat applications.

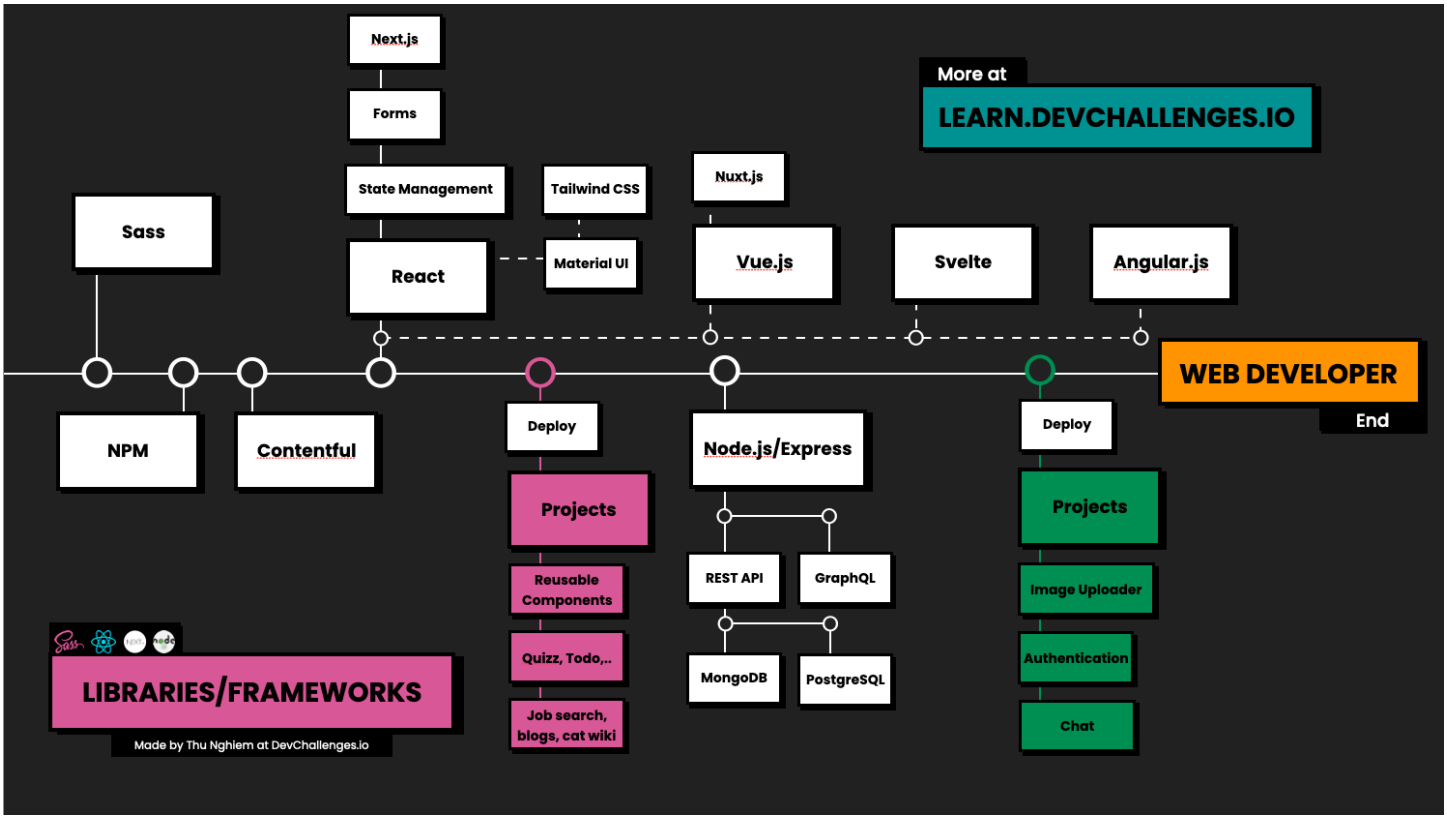
So after learning Javascript, you might want to spend some time to learn about **TypeScript** (Links to an external site.). I love TypeScript as it gives me a safe feeling while writing code.

TypeScript provides static typing, that allows you to catch errors earlier in the debugging process. It also saves your time as it finds bugs before the code runs.

Here's a helpful post on TypeScript types (Links to an external site.) to help you think about them the right way.

You can stop here and start working on some projects. But personally, I would continue by learning at least one framework. Then you can practice JavaScript at the same time.

Libraries and Frameworks to Know



Libraries and Frameworks for Front-end Developers

So let's start with **Sass**, which is a preprocessor scripting language. I use Sass in almost every one of my projects. It makes the CSS look cleaner and makes it faster to develop.

Here's a full course on Sass (Links to an external site.) that'll teach you how to give your CSS superpowers.

Next, we have **NPM** which is a package manager for JavaScript programming. This allows you to install different packages on your machine quickly.

Here's a good beginner's guide to NMP (Links to an external site.) that'll help you get started downloading packages right away.

One of my regrets is that I didn't know about **Contentful (Links to an external site.)** sooner. Contentful is a Headless Content Management System or CMS. It's different from a traditional CMS, as you can store data in Contentful and use it for your front end.

And to get a frontend job, you need to know at least one frontend Framework. I personally would choose **React**. We'll talk about other frameworks later.

Besides learning the basics of React (Links to an external site.), you also want to pay attention to how **State** is managed in React applications and how **Forms** work in React.

Once you've learned some basics, you can solidify your skills by building a bunch of projects with React in this YouTube course (Links to an external site.).

Another regret that I have is that I didn't know about **Next.js (Links to an external site.)** sooner. Next.js is used for *server-side rendering* or *generating static websites*. And yes,

Next.js is still quite new, but I do believe that this is the skill we *must* have as React developers.

You can learn all the Next.js basics in this in-depth handbook (Links to an external site.).

After learning these tools, you also might want to take a look at **Material UI (Links to an external site.)** which is a React component library, or **Tailwind CSS** (Links to an external site.) which is a CSS framework that helps you speed up your work when prototyping or freelancing, for example.

So after learning JavaScript and a framework, you need to **practice** by building projects.

You can start by building some simple reusable components to understand how React works. Then you can build more complex applications like a *Quiz App* or a *Todo App*.

After that, you will want to build even more difficult applications like a *Job Search tool,* a *blog,* or a *Document page*. Again you can find example projects on DevChallenges.io (Links to an external site.).

Full-stack Developer

Now you are ready to apply for a Front-end developer job. If you want to continue to become a Full-stack developer, you can start by learning **Node.js** (Links to an external site.) and **Express (Links to an external site.)**.

Here, you need to how to build a *RESTful API* and you can use **MongoDB (Links to an external site.)** when working with databases. Which is quite simple to learn when you already know JavaScript.

After that, if you want to learn more, you can look into **GraphQL** (Links to an external site.) which is a data query and manipulation language for APIs. Personally, I would also spend some time learning **PostgreSQL**. Compared to MongoDB, it's a bit more difficult to learn as you need to learn about SQL (Links to an external site.) as well.

After learning these tools, you can practice by building applications like an *Image Uploader*, *Authentication*, or *Chat Room*. You can also find these projects on Devchallenges.io (Links to an external site.).

Now you are ready to apply to a Full-stack position. If your job requires you to know **Vue.js (Links to an external site.)**, **Angular (Links to an external site.)** or even **Svelte (Links to an external site.)** you can spend some time learning those tools, too. It should not be too difficult once you already know React.

Web Development Roles and Skills

Web developer job description

Web developers play a crucial role in our society. Responsible for developing and maintaining web pages, they're behind all of our favorite websites, from Wikipedia to YouTube.

Today, practically every organization needs web developers to keep up with competitors. According to the U.S. Bureau of Labor Statistics (Links to an external site.), web developers have a rapidly expanding job market, expected to continue rising by up to 8% by 2029. But why are web developers so essential? What exactly do they do? Below, we'll walk you through a web developer's responsibilities and skills, with tips on where you can get started.

Web developers build websites that meet their company's or client's needs. To do so, they need two things:

1. The ability to create websites
2. The ability to discern how to satisfy a client's needs using a web page

Think of a web developer as a "solutions architect." First, they determine the best way to address the challenges faced by their client. Then, they design a website that meets their objectives.

This is one of the key differences between web development (Links to an external site.) and web design. While web designers focus on a page's layout and aesthetic, web developers are more concerned with structure, functionality, and efficiency and ensuring everything stays within budget.

The U.S. Bureau of Labor Statistics (Links to an external site.) explains how web developers "develop, create, and test website or interface layout, functions, and navigation for usability." To achieve this, web developers need to:

- Meet with clients and IT teams.
- Understand the company's web hosting and security architecture.
- Work with decision-makers to attain more resources if needed.

What are web developer skills?

Web developer skills refer to the specific abilities needed to get hired and successfully perform in a web developer role. Having these skills in abundance can help you stand out from other candidates vying for the same role and in your everyday tasks on the job. As with any job, you also have the ability to develop your skills as a web developer over time. Some web developer skills include understanding HTML, general design skills and being analytical.

Examples of web developer skills

To become a web developer, you'll need a variety of hard and soft skills to succeed. Here are some of the most common skills to have in this role:

- HTML/CSS skills
- Analytical skills
- Responsive design skills
- JavaScript skills
- Interpersonal skills
- Testing and debugging skills
- Back-end basics
- Search engine optimization

HTML/CSS skills

If you hope to become a web developer, you must understand HTML and CSS. HTML is the most used markup language across the web, and web developers use it to create web pages on the Internet. CSS, on the other hand, is used to style the website. It's responsible for the choice of fonts, colors and layout in a website's design. Whereas HTML lays the foundation for a webpage, CSS styles it.

Analytical skills

As a web developer, you'll need to have keen analytical skills to create a variety of successful websites. This involves ensuring the coding is correct so the website is functional. Also, it's important to understand the needs of your customers or clients. You'll need to analyze their wants and needs and be ready to adapt to any changes they'd like to see implemented on the website. The better you understand your client, the better your website will turn out for them in the end.

Responsive design skills

If you work as a web developer, it's important to understand how responsive design works. Because many people view webpages both online and on their mobile devices, the websites you create need to adapt to the changing screen sizes. If your website isn't responsive, this means it won't be compatible with either the desktop site or the mobile site. In other words, it won't function or show up properly on either of these platforms. Your ability to create a responsive design can also impact the number of visitors a website gets since most people are deterred from a website when it's not displaying properly.

JavaScript skills

You'll also need to understand JavaScript as a web developer. JavaScript is a high-level programming language that implements complex features onto a website. This involves things like moving elements, search bars, videos or interactive maps on a webpage. In knowing JavaScript, you'll be able to meet higher expectations from your clients in the web development process. Using these skills, you'll be able to reel in more webpage viewers, as well, since the majority of people tend to prefer interactive web pages that grab their attention. Special features go a long way in engaging viewers, and your job as a web developer plays a large role in seeing them come to fruition on a digital screen.

Interpersonal skills

Since you'll be working with a variety of clients, you must have good interpersonal skills. This includes various social skills like good body language, active listening skills, the ability to collaborate and an overall good work ethic and positive attitude. The more you get along and can effectively communicate with your clients, the smoother the web development process will be for both of you.

Testing and debugging skills

As a web developer, it's important to know how to continuously test your webpage for any possible errors. Should any arise, you should also know how to debug it. Testing makes sure your coding is doing what it's supposed to and will come down to either functional testing or unit testing. Being able to consistently and effectively test and debug your webpage is essential in this role.

Back-end basics

You'll also need to understand some of the fundamentals of back-end web development. This is because you'll be working in collaboration with back-end developers or might be tasked with using these skills yourself. In this regard, you'll need to know how servers work as well as APIs, SQL, databases and cybersecurity. All of these are important skills to know as they relate to the server-side of a webpage.

Search engine optimization

As a web developer, you must know all about search engine optimization. This is because website traffic is becoming more and more important to companies. If a website has good SEO, it'll attract a greater number of webpage visitors by showing up high enough in search engine result pages. Because more and more companies are looking to have good SEO, it's an important skill to have as a web developer.

How to improve web developer skills

As a web developer, there are several ways to hone in on the skills necessary for this role. Here are some tips to guide you toward improving your web developer skills:

1. Practice coding

The more you write code, the more experienced you'll become. This will not only help your typing skills, but it'll help you catch mistakes that you can avoid in the future. You can practice through various free online programs. As long as you're consistent, you'll see improvements.

2. Learn something new

If there's a programming language you don't have experience in, consider taking a course in it. This will help you get out of your comfort zone and immerse yourself in a new skill

that could work hand-in-hand with your current skillset.

3. Take a break

Taking the time to let your mind rest could be the boost you need when you come back to work. If you're constantly working, this could lead to you feeling run down and burned out.

4. Practice improving your interpersonal skills

You can improve your interpersonal skills by being more mindful of your body language, the way you respond in certain situations and being a better listener overall.

Web Application Architecture

Fundamentals of Web Application Architecture: How Web Apps Work?

Any internet user views the web page on their screens by means of numerous interactions that occur in the backend web development. A chain of interactions takes place amongst various components like database middleware system, server, browser, and user-interface. For managing all such interactions, the **web application architecture** takes place.

A web application functions well on remote web servers with the aid of the internet. This application is kept on a remote server and gets transported over the browser interface. But to develop a robust and impeccable web app, the crucial part is to choose the right web application architecture and its components.

What is Web Application Architecture?

Web application architecture is a framework that outlines its key external and internal components, the interaction and relationship between those app components, like **user interfaces, middleware systems, and databases**. In this manner, the web experience will be facilitated which can be utilized by consumers.

Its chief task is to ensure that all these components function all together and exist as a sturdy foundation to develop and fix up everything subsequently.

This application development architecture approaches and connects dissimilar elements to empower web experience. It is considered as the mainstay of our everyday browsing. Entering a URL, viewing, and interacting with the site in the meantime, the browser

interconnects with the server is one of the methods to term what is a web application architecture.

Below is the representation of this architecture:

Most of the developers might neglect the phase of generating the blueprint and simply develop a web application. However, if you wish to form a scalable product that has great competence and is robust, reliable, and secure, a web application architecture must not be something that you can overlook.

All the stakeholders comprising clients, developers, or end users must join in the scheduling of this architecture to modify it as per the anticipations.

Characteristics of a Sturdy Web Application Architecture

- The visual aesthetic is supported
- Resolves development company problem
- Guarantees speedy user experience
- Offers security
- Facilitates A/B analytics and testing
- Rules out and log errors in the simplest way
- Ensures elevated automation
- Self-regulating and sustainable

Important Components of Web Application Architecture

Web application architecture encompasses various components that are separated into two main categories – user interface app and structural components. Let's understand each of these application components in brief.

1. User Interface App Components
2. User interface components are connected to the **settings, display, and configurations** of a web application. They are accountable for producing a web application's experience and interfaces. They contain a number of elements, for instance, statistical data, dashboards, layouts, configuration settings, activity tracking, notification elements, to name a few.
3. Structural Components
4. The structural components have an important role in fabricating the performance of a web application. Therefore, these components allow users to network with the web app. This essentially includes the web browser or client, the database server, and the web application server.
    - **The web browser or client:** The users can interact and communicate with web applications via this platform. HTML, CSS, and JavaScript are the programming languages that are useful to form this component.
    - **The database server:** It offers relevant data or information or business logic that is managed and stored by a web app server. It stores, retrieves, and delivers the data.
    - **The web application server:** It manages the fundamental hub that supports multi-layer applications and business logic, and is usually developed utilizing PHP,

Python, .NET, Java, Node.js, and Ruby on Rails (Links to an external site.).

Types of Web Application Architecture

The web application architecture is an arrangement of dealings between many web application components. A safe architecture of an application is determined by how the logic of any web app is dispersed amongst the client and server-side. The model view controller pattern serves to divide the presentation layer from business logic.

Contingent on varied factors like the choice of application logic, purposes, features, functions, business's priorities, developers will select one from the below mentioned three main types of web application architecture to begin configuring the architecture.

1. Micro Services
2. These services are small and lightweight that render single functionalities. Microservice architecture offers many benefits for users as well as developers. This modern web application architecture enhances productivity and creates the deployment process faster.
3. One of the best things about such a service architecture is that the developers can choose any technology stack as components that make the app dependent on each other. Hereafter, the components are established by employing different technologies. Thus, microservices architecture of web application makes the app development process faster and easier.
4. Serverless Architecture
5. In such a type of contemporary web server architecture, the third-party cloud infrastructure is accountable for computing and maintaining web servers. A software developer will employ third-party cloud infrastructure for infrastructure management and servers. With the assistance of serverless architectures, you can implement the app code without any requirement of infrastructure-related responsibilities.
6. It is best suited to the companies who do not want to manage or support the web server and hardware of the web application. This web development architecture for web application is also termed as serverless computing or '**Function as a Service**' (FaaS). Some might even call it the prospect of software architecture patterns.
7. Single Page Applications (SPAs)
8. Single Page Application (Links to an external site.) is a modern web application architecture that aims on augmenting experience. SPA will process the complete web page once. Then when the user clicks on something on the web page, the requested data will get rationalized. And the rest of the web page will stay as it is.
9. In other terms, a single page web application empowers vigorous interaction by only upgrading the requested content on the page. SPAs became a reality because of AJAX – a small form of XML and Asynchronous JavaScript. SPA architecture just averts the excessive interruptions and delivers a collaborating user experience.

How Web Apps Work? An Example

Let's understand how web application architecture works and is applied in single page

apps and multi-page apps.

The front end of the software application can serve either dynamic or static content. In most use cases, it is a mixture of both. Static web pages are present on the web server and encompass information that does not amend. Dynamic web pages alter the information each day or in reply to the user request – like your Twitter feed or any news website.

The amalgamation of static and dynamic content creates up the web application. Single Page Application is an example of a web app with dynamic content.

1.Single Page Applications

The foremost goal of the SPA is the capability to access all the data from an HTML page. The software architect can create the site run quicker and ease out the load from the web server by moving the app logic to client side and utilizing the server side as a data storage. Apart from CSS and HTML, the front end is transcribed on a single framework, which energetically produces content and conveys it to the end user – like your Gmail or Facebook feed.

The dependencies amid components are close-fitted. This states that generating changes to one of the UI or UX elements requires re-writing the entire front end coding. From the time when SPA carried the logic to client side, it has to be transcribed by employing client side scripting.

If you are utilizing the client-side scripting tools, then you are fundamentally building templates, so that whenever the user requests any content, a web server merely transfers this information back to the browser.

This considerably decreases the server load, in contrast to the server-side scripting. The primary technology of this client-side scripting is JavaScript. This language permits the formation of both, small and vigorous apps, besides its several popular web frameworks (Links to an external site.).

When the part of the web server is decreased to the data services, this is at times known as the thin server architecture. We cannot just talk about the SPA without citing the traditional model – Multi-Page Applications.

2.Multi-Page Applications (MPAs)

In multi-page apps, some of the content requests require a completely new page to be recovered from the server. These are enormous apps with multi-layered and multifaceted UI. AJAX technology resolves the problems of composite applications by transporting a vast quantity of data between browser and server, stimulating only selected elements of the app.

Simultaneously, the specified method brings more complications, as it becomes tougher to build when compared to that of single-page applications.

Contrary to client-side scripting of SPA, old-fashioned applications are transcribed utilizing both server and client side languages. Server-side scripting states that all actions are accomplished on the server's end, so whenever you request any content, it processes the script, repossesses the data from storage, and selects the content to exhibit.

For any server-scripting language and programming language, you must be acquainted with PHP, Java, C#, Ruby on Rails (Links to an external site.), to name a few.

Web Application Architecture

Fundamentals of Web Application Architecture: How Web Apps Work?

Any internet user views the web page on their screens by means of numerous interactions that occur in the backend web development. A chain of interactions takes place amongst various components like database middleware system, server, browser, and user-interface. For managing all such interactions, the **web application architecture** takes place.

A web application functions well on remote web servers with the aid of the internet. This application is kept on a remote server and gets transported over the browser interface. But to develop a robust and impeccable web app, the crucial part is to choose the right web application architecture and its components.

What is Web Application Architecture?

Web application architecture is a framework that outlines its key external and internal components, the interaction and relationship between those app components, like **user interfaces, middleware systems, and databases**. In this manner, the web experience will be facilitated which can be utilized by consumers.

Its chief task is to ensure that all these components function all together and exist as a sturdy foundation to develop and fix up everything subsequently.

This application development architecture approaches and connects dissimilar elements to empower web experience. It is considered as the mainstay of our everyday browsing. Entering a URL, viewing, and interacting with the site in the meantime, the browser interconnects with the server is one of the methods to term what is a web application architecture.

Below is the representation of this architecture:

Most of the developers might neglect the phase of generating the blueprint and simply develop a web application. However, if you wish to form a scalable product that has great

competence and is robust, reliable, and secure, a web application architecture must not be something that you can overlook.

All the stakeholders comprising clients, developers, or end users must join in the scheduling of this architecture to modify it as per the anticipations.

Characteristics of a Sturdy Web Application Architecture

- The visual aesthetic is supported
- Resolves development company problem
- Guarantees speedy user experience
- Offers security
- Facilitates A/B analytics and testing
- Rules out and log errors in the simplest way
- Ensures elevated automation
- Self-regulating and sustainable

Important Components of Web Application Architecture

Web application architecture encompasses various components that are separated into two main categories – user interface app and structural components. Let's understand each of these application components in brief.

1. User Interface App Components
2. User interface components are connected to the **settings, display, and configurations** of a web application. They are accountable for producing a web application's experience and interfaces. They contain a number of elements, for instance, statistical data, dashboards, layouts, configuration settings, activity tracking, notification elements, to name a few.
3. Structural Components
4. The structural components have an important role in fabricating the performance of a web application. Therefore, these components allow users to network with the web app. This essentially includes the web browser or client, the database server, and the web application server.
    - **The web browser or client:** The users can interact and communicate with web applications via this platform. HTML, CSS, and JavaScript are the programming languages that are useful to form this component.
    - **The database server:** It offers relevant data or information or business logic that is managed and stored by a web app server. It stores, retrieves, and delivers the data.
    - **The web application server:** It manages the fundamental hub that supports multi-layer applications and business logic, and is usually developed utilizing PHP, Python, .NET, Java, Node.js, and Ruby on Rails (Links to an external site.).

Types of Web Application Architecture

The web application architecture is an arrangement of dealings between many web application components. A safe architecture of an application is determined by how the

logic of any web app is dispersed amongst the client and server-side. The model view controller pattern serves to divide the presentation layer from business logic.

Contingent on varied factors like the choice of application logic, purposes, features, functions, business's priorities, developers will select one from the below mentioned three main types of web application architecture to begin configuring the architecture.

1. Micro Services
2. These services are small and lightweight that render single functionalities. Microservice architecture offers many benefits for users as well as developers. This modern web application architecture enhances productivity and creates the deployment process faster.
3. One of the best things about such a service architecture is that the developers can choose any technology stack as components that make the app dependent on each other. Hereafter, the components are established by employing different technologies. Thus, microservices architecture of web application makes the app development process faster and easier.
4. Serverless Architecture
5. In such a type of contemporary web server architecture, the third-party cloud infrastructure is accountable for computing and maintaining web servers. A software developer will employ third-party cloud infrastructure for infrastructure management and servers. With the assistance of serverless architectures, you can implement the app code without any requirement of infrastructure-related responsibilities.
6. It is best suited to the companies who do not want to manage or support the web server and hardware of the web application. This web development architecture for web application is also termed as serverless computing or '**Function as a Service**' (FaaS). Some might even call it the prospect of software architecture patterns.
7. Single Page Applications (SPAs)
8. Single Page Application (Links to an external site.) is a modern web application architecture that aims on augmenting experience. SPA will process the complete web page once. Then when the user clicks on something on the web page, the requested data will get rationalized. And the rest of the web page will stay as it is.
9. In other terms, a single page web application empowers vigorous interaction by only upgrading the requested content on the page. SPAs became a reality because of AJAX – a small form of XML and Asynchronous JavaScript. SPA architecture just averts the excessive interruptions and delivers a collaborating user experience.

How Web Apps Work? An Example

Let's understand how web application architecture works and is applied in single page apps and multi-page apps.

The front end of the software application can serve either dynamic or static content. In most use cases, it is a mixture of both. Static web pages are present on the web server and encompass information that does not amend. Dynamic web pages alter the

information each day or in reply to the user request – like your Twitter feed or any news website.

The amalgamation of static and dynamic content creates up the web application. Single Page Application is an example of a web app with dynamic content.

1.Single Page Applications

The foremost goal of the SPA is the capability to access all the data from an HTML page. The software architect can create the site run quicker and ease out the load from the web server by moving the app logic to client side and utilizing the server side as a data storage. Apart from CSS and HTML, the front end is transcribed on a single framework, which energetically produces content and conveys it to the end user – like your Gmail or Facebook feed.

The dependencies amid components are close-fitted. This states that generating changes to one of the UI or UX elements requires re-writing the entire front end coding. From the time when SPA carried the logic to client side, it has to be transcribed by employing client side scripting.

If you are utilizing the client-side scripting tools, then you are fundamentally building templates, so that whenever the user requests any content, a web server merely transfers this information back to the browser.

This considerably decreases the server load, in contrast to the server-side scripting. The primary technology of this client-side scripting is JavaScript. This language permits the formation of both, small and vigorous apps, besides its several popular web frameworks (Links to an external site.).

When the part of the web server is decreased to the data services, this is at times known as the thin server architecture. We cannot just talk about the SPA without citing the traditional model – Multi-Page Applications.

2.Multi-Page Applications (MPAs)

In multi-page apps, some of the content requests require a completely new page to be recovered from the server. These are enormous apps with multi-layered and multifaceted UI. AJAX technology resolves the problems of composite applications by transporting a vast quantity of data between browser and server, stimulating only selected elements of the app.

Simultaneously, the specified method brings more complications, as it becomes tougher to build when compared to that of single-page applications.

Contrary to client-side scripting of SPA, old-fashioned applications are transcribed utilizing both server and client side languages. Server-side scripting states that all actions are accomplished on the server's end, so whenever you request any content, it processes the script, repossesses the data from storage, and selects the content to exhibit.

For any server-scripting language and programming language, you must be acquainted with PHP, Java, C#, Ruby on Rails (Links to an external site.), to name a few.