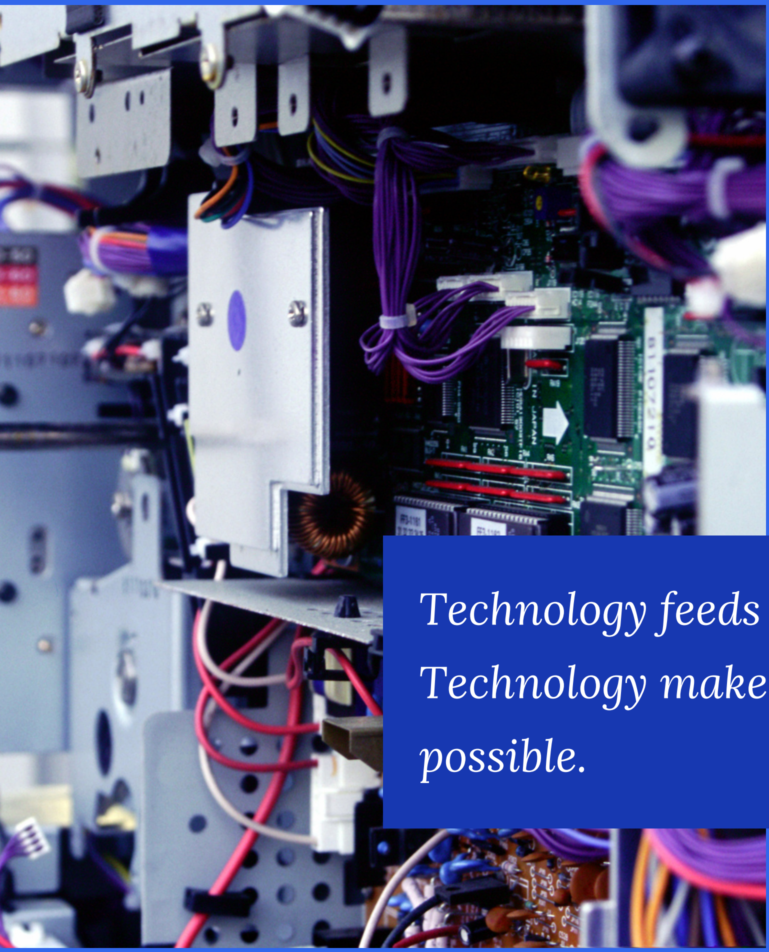
THE BASIC COMPONENTS OF SOFTWARE DEVELOPMENT

Swan Software Solutions

Reliable. Scalable. Affordable.

317-597-5535 | www.swansoftwaresolutions.com | 706 Pro-Med Ln. #101, Carmel, IN 46302

TECHNOLOGY AT A FUNDAMENTAL LEVEL



Technology feeds on itself. Technology makes more technology possible.

-Alvin Toffler

List the top technologies you use in a day. What are they? The smart fridge that orders your food? The medical device that enables you to live a healthier life? What about your car that is basically a computer on wheels?

It may seem as if technology is infinite and incomprehensible. However, it is much like construction.

The same material that can be combined to make a skyscraper might also be used to make a garden shed. A gold faucet can be just as easily attached to the pipes in a billion-dollar mansion or a small apartment.

Why?

Because the basic components of construction are the same. Same pipe sizes. Same board sizes. Same electrical wire. No matter what they are used to construct, the fundamental and most basic components are often identical.

In the same way, technology--be it social media, exercise app, or website--will on a fundamental level be created with many of the same components.

What are they?

Read on to discover the basic components of software development.

PLATFORMS: THE FOUNDATION OF SOFTWARE

Whether building a house, skyscraper, or barn, the first and most important element to be completed is that of the foundation. A builder must determine the type of land, the size of the building, and what material and design will create a strong and lasting foundation.

In the exact same way, a developer must determine the best foundation on which to build an application. Software developers must choose what systems they want to make the application available on. Software programmers write base code on top of that platform to host their code. Platforms are a combination of hardware and software that will act as a foundation for the application.

Not every application will work on every device or operating system. When an error of "Not Supported on this Device" pops up, it is because the developers did not choose to build on that foundation.

Many applications are built on computing systems such as--

- Linux
- macOS
- Windows
- Android
- OS
- and more.

However, there are also other kinds of software systems with a variety of purposes on which an application can be constructed.

Think of it like someone deciding to build a tiny home on a chassis in order to enable the completed house

to be towed behind their vehicle to new locations. Another example would be the construction of a houseboat. Changes in the foundation would enable the house to have water as its foundation.
Although one could build a similar floorplan with similar fixtures and similar colors on land, wheels, or water, the foundation would categorically change the building.
An application platform will also change the way that the software can be used.
Deciding on the foundation of the application will depend on what one wants the program to do.
Some of the main types of platforms beyond the computing platforms mentioned above are--

01 **UTILITY**
Platforms, such as a search engine, provide a useful service, generally for free.

02 **CONTENT DISTRIBUTION**
Platforms connect the owners of places users are found with marketers placing ads.

03 **MARKETPLACE**
Platforms connect buyers with the sellers providing the product or service they need.

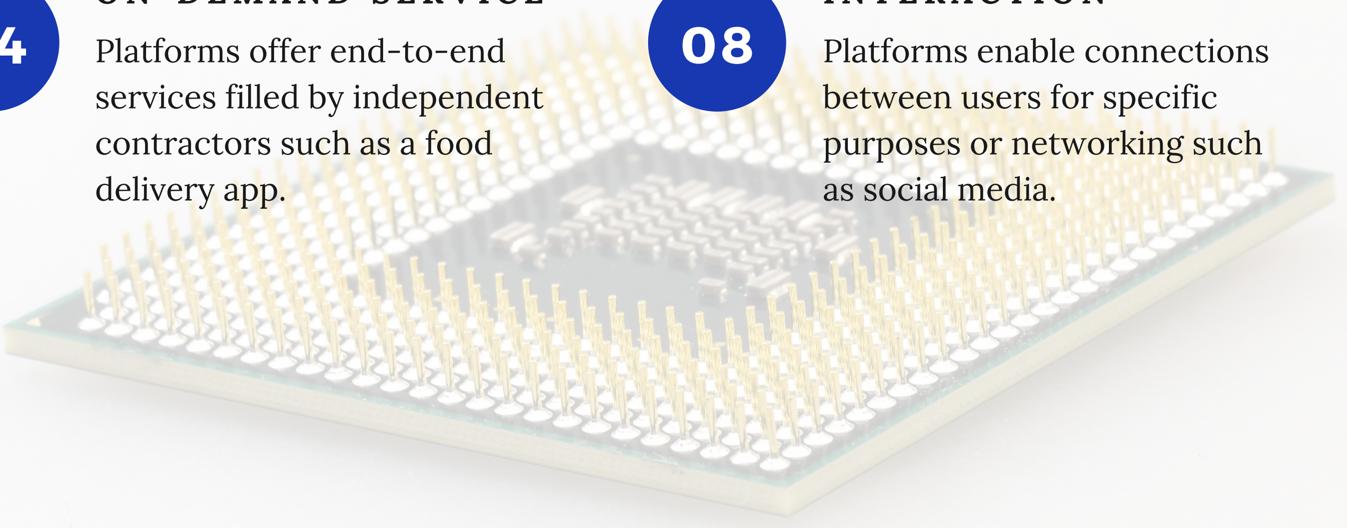
04 **ON-DEMAND SERVICE**
Platforms offer end-to-end services filled by independent contractors such as a food delivery app.

05 **TECHNOLOGY**
Platforms provide building blocks or services, which are then sold to developers.

06 **CONTENT CROWDSOURCING**
Platforms connect users with content created by other users such as videos or restaurant reviews.

07 **DATA HARVESTING**
Platforms harvest data from users and share it with other users in order to provide a useful service such as maps.

08 **INTERACTION**
Platforms enable connections between users for specific purposes or networking such as social media.



FRAMEWORKS

Software Developers might mention frameworks when discussing programming. What are they?
Frameworks are existing pieces of code that will enable a developer to perform a specific task, which can be taken and altered into doing what the developer needs it to do. When using a framework, some parts of the code will always remain the same while other parts can be altered to fit application needs.
By plugging existing code into an application, a developer can create an application in a shorter period of time. Another benefit is that the code is proven to work and has been improved by other developers so there will be fewer errors than creating from scratch.
Essentially, a framework takes care of the mundane and repetitive tasks involved in software and enables the programmer to use his skills on the unique characteristics of the application.
Think of it a bit like a builder creating a custom kitchen. The client has specific requirements that they want to be integrated into the kitchen.
Now, the builder could create each and every cabinet from scratch. However, this would take longer and he might run into issues not previously thought about.
Most likely the builder would take existing cabinets and layouts and put them together to create a functional kitchen. He may need to build a few cabinets to connect them and perhaps alter some features, but in the end, he will have created a custom kitchen at a fraction of the time and price that he would need to charge otherwise.
Frameworks enable developers to do the same. By filling in the gaps in the framework with custom code, they can create better applications in shorter periods of time and usually at cheaper prices.

LIBRARIES

In software development, the use of libraries is common. Although similar to frameworks, libraries do differ.
While frameworks provide a skeleton of code upon which developers can hang the flesh of their application, libraries are pieces of code that perform an operation. Frameworks are less capable of customization or change than libraries and cannot be changed midway through a project. If a framework is the cabinets that provide a basic outline on which to create a custom kitchen, the library is the window that is plugged into the hole in the wall and will provide the same function whether that hole is in the wall of a mansion or the wall of a shack. Customization isn't the point in a library. Functionality is.

TOOLS

Another important component of software development is tools. From Jira to Azure, the tools of software development enable programmers to accomplish tasks, test applications, and manage projects. Just as a builder would struggle to construct something without tools, a developer's use of tools can save them time and empower them to succeed.
Software Suites that include the tools needed to create software are called Integrated Development Environments (IDEs), which we will look at next. Tools are an important component of software development

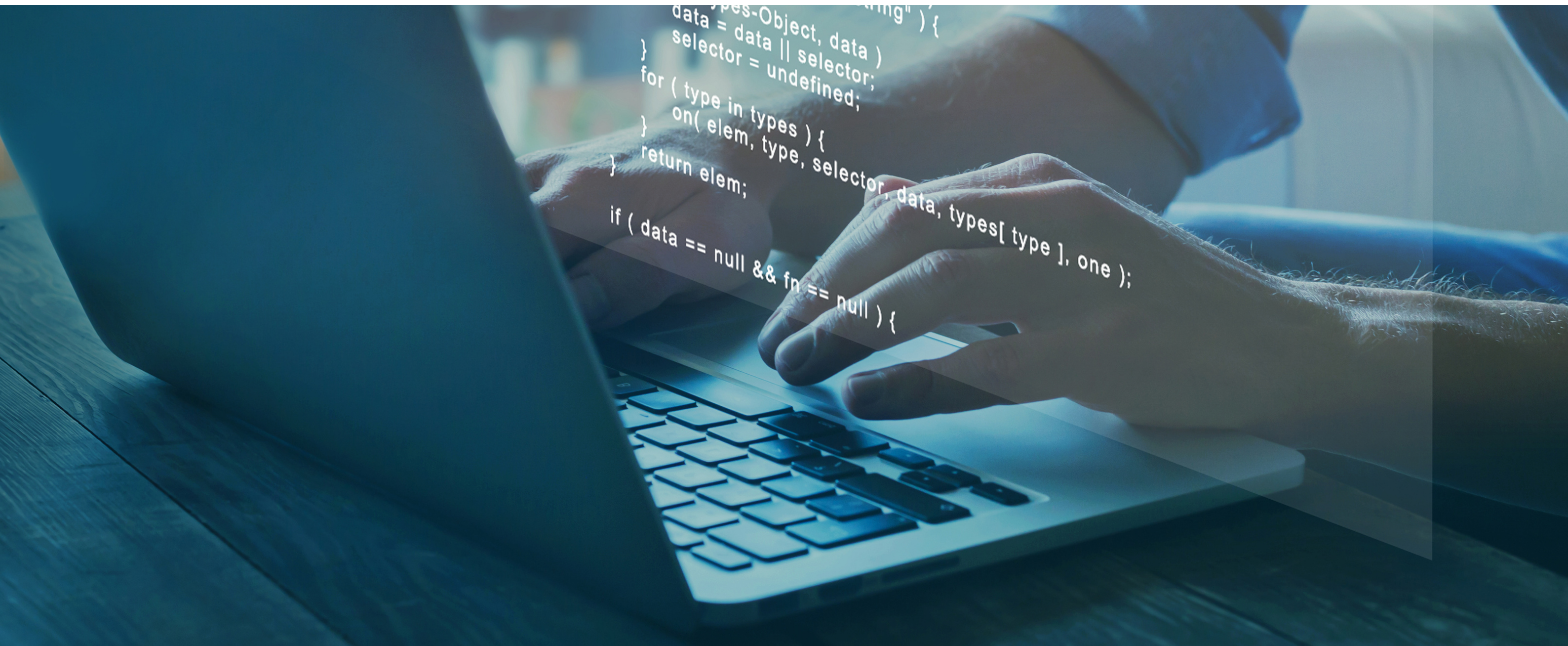
INTEGRATED DEVELOPMENT ENVIRONMENT

Integrated Development Environment (IDE) is an application that makes development easier.
Although not necessary, an IDE allows developers to work better and more efficiently.
IDEs allow code to be written, debugged, and compiled in one place.
Technically, one can write code in any word processing program, but it will not be as easy as creating it in an IDE, nor will the word processing program debug or compile the code.
Words with special meanings within the syntax might be highlighted in colors to allow a developer to read the code better.
Additionally, autocomplete will save a developer keystrokes.

Think of it like a factory that builds sheds to sell.
The factory is set up to build sheds and sheds only. Wood left over from one project can be used on another. The perfect tools are available to do the specific tasks required to create a shed. Some employees will have the specific task of checking to make certain the buildings are correct.
Although one could build the shed from scratch on site, the factory provides an environment to make it better, easier, and to eliminate errors.
This is what an IDE does for software development. The same task might be accomplished without the IDE, but it will be accomplished better with one.

PROGRAMMING LANGUAGES

One of the most important decisions to be made before development can start is the choice of programming languages.
Programming languages are the way that computers communicate with each other and humans communicate with computers. Depending on the source, it is estimated that there are somewhere between 150-9,000 programming languages in the world, although many are obsolete, esoteric, or unused.
Not all programming languages can do the same thing. Front-End Languages are used for Front-End programming. Back-End Languages are used for Back-End programming. Some languages can be used to do both.
When planning which languages to use for Front and Back ends, the developer must make certain that the two will "talk" to each other. Some programming languages will and some will not.
Just like with human languages, programming languages perform the same purpose but have a different structure and benefit. When looking at human languages, one can see a vast difference in structure and detail. One language is very particular in its details while another requires the speakers to rely heavily on intuition or context to discern meaning.
Depending on the application being created, a developer must take into account what programming language will best suit the project's needs.
In construction, programming languages are like building materials. A builder must decide what size of wood, what kind of wood, or even whether to use wood at all. He or she must determine if the steel studs they wish to use will work with the concrete foundation or if they could choose better materials to do a better job. Each part of the building will connect with the other parts, just as an application will connect a variety of components to accomplish a specific purpose.



```
                                   ing" ) {
    es-Object, data )
    data = data || selector;
    selector = undefined;
  }
  for ( type in types ) {
    on( elem, type, selector, data, types[ type ], one );
  }
  return elem;
}
if ( data == null && fn == null ) {
```

POPULAR PROGRAMMING LANGUAGES

Although there is a myriad of programming languages, most developers will use one of the following. Also to be noted, the popularity of programming languages is transient. Even measuring the comparative popularity of coding languages is difficult as it, too, varies by which metric one employs to determine the numbers. Here are a few of the most popular programming languages.

C
One of the oldest and most popular programming languages.

JAVA
An object-oriented and general use programming language.

PYTHON
A general purpose language that offers interactive qualities.

C++
General Purpose language with generic, functional and object-oriented features.

C#
A member of the C-family and general-purpose programming language.

VISUAL BASIC
An event-driven programming language known for its Component Object Model

JAVASCRIPT
A programming language often used in web development.

PHP
A general purpose language geared toward web development.

ASSEMBLY LANGUAGE
A generic term for a specific type of low level programming languages.

SQL
A highly functional programming language used to manipulate databases.

DATABASES

Although not necessary for software development, the creation of databases and a Database Management System can be key for a successful application.
Merriam-Webster defines a database as "a usually large collection of data organized especially for rapid search and retrieval (as by a computer)."
First used in the 1960s, databases have evolved in their structure in order to allow faster and more accurate data retrieval.
At its core, a database is simply a collection of data. For instance, an online store's database might include data on stock, prices, customers, costs, and more.
Yet, all this data is useless unless one can find it and use the information.
That is where a Database Management System (DBMS) is used.
The online store might have the information on the products in stock written down somewhere, but the Database Management System organizes this information into a structured format that allows it to be searched and accessed.
Imagine a contractor purchasing materials to build a house. All the materials are thrown into a large bin and mixed together. Nails, screws, wood, and more are all jumbled.
Now, the contractor needs sheetrock screws. However, he or she will have to dig through the confused mess and hope they come across the screws they need.

"Getting information off the Internet is like taking a drink from a fire hydrant."

-Mitchell Kapor

Even though all the materials are there, they are not easy to access and the result is a contractor relying on luck to find what he needs to succeed.
Having the information in a database without any way to access or search through it is just as useless as a jumbled bin of construction materials.
A database keeps the information safe, and a DBMS allows that information to be useful. While not every application needs an extensive database, in some applications, both a database and a DBMS can be crucial for success.

QUALITY ASSURANCE

"It's not at all important to get it right the first time. It's vitally important to get it right the last time."

-Andrew Hunt and David Thomas

Although quality assurance might not seem like a basic component of software, the reality is that without it, all other software components would be nearly useless.
One of the most fundamental requirements of software development is the testing and re-testing of each component--individually and collectively--to ascertain that they work as designed and developed. Quality Assurance employs both QA Engineers and automated processes to find and eliminate bugs or other issues with the code.
Anyone who has ever constructed a building or renovated a house is aware of the permits required to do so. Throughout the process, inspectors will come and sign off on various aspects before a final inspection will declare that the project is done and safe.
Quality Assurance is the building inspector of the software development world. Their goal is to ultimately ensure that the application is secure, issue-free, and completed to specifications.
Even though the inspector is not a construction material, his or her presence is essential to the completion of the building, just as a QA Engineer's is to software development.

DEVELOPERS: THE BUILDERS OF SOFTWARE APPLICATIONS

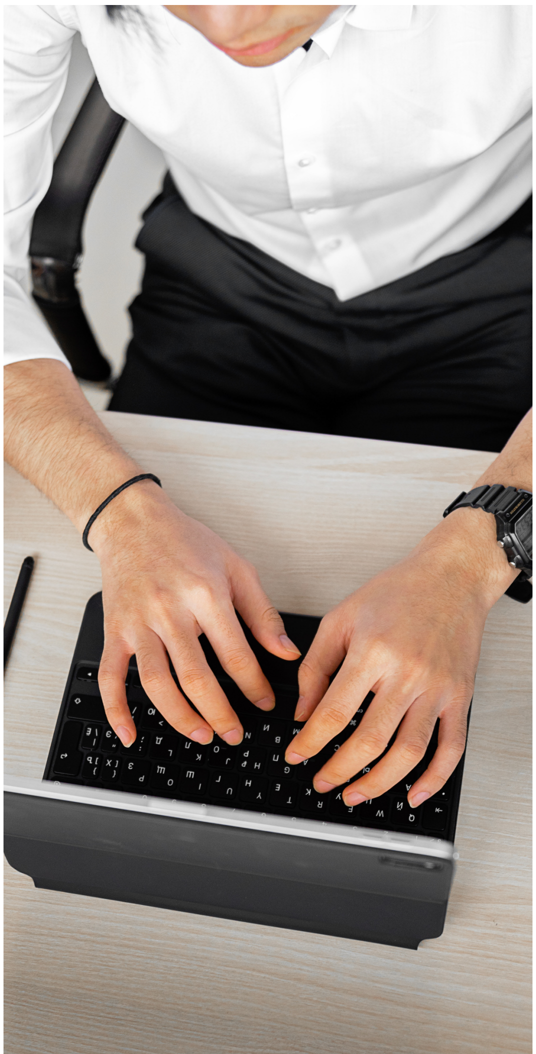
Perhaps the most important of all software components are the developers. Although one could not have an application without the materials needed to build it, one also couldn't have one without the builders capable of taking the materials and creating a necessary product.
Many kinds and levels of developers work together in the creation of software, just as many kinds and levels of tradespeople come together to construct a house.
An architect creates the plan.
Senior-level builders/developers assign the tasks and guide the other developers/builders throughout the process.
Developers/ builders complete assigned tasks to take them closer to the completion of the application or house.
Tradespeople specialize in specific skills such as wiring, plumbing, or tiling.
Developers specialize in specific skills such as front-end programming, back-end programming, or User Interface/User Experience design.
The reality is that each component in software or construction is necessary for the completion of the whole. The best construction materials without builders to use them are useless. The best programmers without platforms or programming languages would be hard-pressed to create anything of value.
Both users and materials are necessary.

FROM COMPONENTS TO APPLICATIONS

When boiled down to the basic materials, most things seem simple. What matters is the way things are combined.
The same materials used to create a great work of art can also be used by a child to fingerpaint.
Only by combining high-quality materials in the best way possible can excellent programming be created.

“In some ways, programming is like painting. You start with a blank canvas and certain basic raw materials. You use a combination of science, art, and craft to determine what to do with them. You sketch out an overall shape, paint the underlying environment, then fill in the details."

-Andrew Hunt



Why Swan Software Solutions

Swan Software Solutions was founded in 2005 and prides itself on delivering reliable, scalable, and affordable solutions that exceed our clients' expectations.

With over 140 developers, we have many skilled team members. From QA Engineers to Project Managers, we can provide the skill set you need for your project. Our team also specializes in a variety of programming languages, and we provide a free discovery process. Clients only pay for the developer's time, but we also provide the Project Manager, QA Engineer, Business Analyst, and other roles necessary to keep your project running smoothly.

Swan believes in the Agile Principles and Values, which enables us to pivot to meet a client's needs. We believe in communication and encourage the input of our clients with real-time communication. We hold daily, weekly, or bi-weekly team meetings based on the project and client's needs.

We provide both Fixed Bid contracts and Full-Time-Equivalent (FTE) Developers in order to best meet the needs of our clients. Some of our FTE Developers have outsourced to the same company for over twelve years, but other companies have found that a Fixed Bid project --such as a new design for their website-- has been the best choice for them.

No matter what sort of development help your company might need, Swan is committed to helping you succeed.

Because when you succeed, we succeed.

To find out more or set up a free assessment, contact us at swansoftwaresolutions.com

Swan Software Solutions

Reliable. Scalable. Affordable.

Free Discovery Process