Basic Concepts

The world of web development orbits around a galaxy of interconnected concepts, techniques, and technologies which are constantly evolving. This can sometimes hinder the ability of newcomers to learn and practice the craft effectively.

To help build a solid foundation for learning web development, we're going to go over some of the most basic concepts. The objective of this chapter is to present a broad picture of the kind of things involved in the development of web applications. In the hopes that the following chapters don't carry too much cognitive weight.

Remember that programming is not about memorization. Where in doubt, you can always refer to this chapter to refresh your memory and review the concepts.

Locating Sites on the Web

The internet is an ever growing web of machines connected between each other. Similar to how houses are connected in a city, and cities are connected to each other, if you don't know which direction to take, you're going to get lost.

The most usual and high level way to locate a website is by it's **URL**, this term stands for *Uniform Resource Locator* and it's an encoded string which let's us identify a location in a computer network and the means of retrieving a resource on that machine. Let's take a look at the programming subreddit URL to identify its basic parts: https://www.reddit.com/r/programming/.

First we have the *scheme* `https` which denotes the protocol used for communication. Then we have the *hostname* `www.reddit.com` which is basically the domain to which the URL is referring to. And lastly we have a *path* to the `/r/programming/` resource, which identifies that of all the content in that website, we want to get the programming subreddit feed.

As it's very common with technical definitions, there's more details and complexity in the precise definition of URL, but the simplified definition from above can help us untangle other concepts, in particular regarding the hostname.

It turns out that `reddit.com` is not the direct location of the machine where the site is hosted. Similar to how house addresses are not the latitude and longitude

coordinates, but a more easily to remember combination of names. Websites locations are directly identified by an *IP address*.

The structure of IP addresses differs from version to version, the most widely use version is IPv4 which reads as a four numbers from 0 to 255 separated by dots like:

- 127.0.0.1 (localhost)
- 208.80.153.224 (Wikipedia)
- 172.217.2.238 (Google)

Go ahead and enter http://172.217.2.238 into your browser's address bar. You'll see google popping up. But how do we know `www.google.com` refers to this address?

Well, the Google domain is registered in a **DNS** (which stands for *Domain Name System*). These DNS are like tables that translate memorizable names to the numerical IP addresses needed for locating a machine.

The Languages of the Digital Hyperspace

For two programs or machines to communicate over a network with one another, there needs to be a set of rules that mandate how the information should be interpreted. These sets of rules are called network protocols.

There are a vast number of these protocols in use today, some of them are improvements over predecessors, other ones shine in particular use cases.

One of the most used protocols is **HTTP** (which stands for *Hypertext Transfer Protocol*), most web applications and online services rely on HTTP. It's the foundation of data communication for the World Wide Web. This protocol is based on a client sending requests to a server, and the server sending responses to the client, where each response is associated with a request.

Each HTTP request has an associated *method* which is a semantic construct design to indicate what the request is for. There a quite a few methods, but here we'll only discuss the most used. The **GET** method requests a representation of a particular resource, and denotes that data should only be retrieved. The **POST** method requests that the server accept and process a piece of information associated with a resource. The **PUT** method requests that a piece of information should be stored under the supplied resource. The **DELETE** method requests that the specified resource should be deleted.

There are other useful protocols which are commonly used for specific use cases:

- The File Transfer Protocol (**FTP**) is used for transferring files between two machines in a network.

- The Secure Shell protocol (**SSH**) is used for remotely logging in to a computer system.
- The Post Office Protocol (**POP**), Internet Message Access Protocol (**IMAP**), and Simple Mail Transfer Protocol (**SMTP**) are mostly used to send and retrieve emails.

The Ends of a Web Application

A web application can be conceptualized and broken apart in several ways, a common separation of concerns from an architectural point of view is the identification of the **front end** and **back end**. When someone refers to the front end, they usually are talking about the part of the application that runs in the client (for example, in the browser). In the same vein, the back end is mostly associated with the part of the application that runs in the server (the machine that communicates with the browser). However this is not always the case.

Other ways in which the front end and back end differ is in the level of interaction with the user. Most of the time, the user of the application is going to interact exclusively with the front end, which in turn, communicates with the back end. It's common to also refer to the visual/interactive aspects of an application "the front end", and to the data-handling/business-logic aspects "the back end".

It's worth noting that these distinctions are not universal. Some applications do all the business logic in the client code, handling only the data persistence in the server code. Some applications don't have a visual layer and their back ends consist of clients which communicate with other back ends.

In the next two section we'll talk about the usual knowledge a front end developer and a back end developer have.

An important thing to note is that there are a lot of tools which can make your life easier if you lack knowledge or skill in either front or back end. There are frameworks which provide good configuration defaults and allow you to focus your attention to the business logic of your application.

Learning both front end and back end development at the same time can be quite overwhelming, so if you ever want to get an app up and running with no hassle you could for example:

- Ignore the back end details and use Google Firebase or implement a thin back end layer to just persist information.
- Ignore the front end details and write a standalone API or use Bootstrap for putting together a nice looking simple interface.

An example of a type of system that requires a tight integration between a front and a back end is a *Content Management System* (**CMS**) which is an application that supports the creation and modification of digital content, usually supporting multiple users in a collaborative environment and the means for those users to publish and consume content.

Behind the Website

An important concept in back end development is the *Application Programming Interface* (**API**), this refers to a set of subroutine definitions, protocols, and tools for building application software. However, in the context of web development, an API is a programmatic interface consisting of one or more publicly exposed endpoints to a defined request–response message system, typically expressed in JSON or XML, which is exposed via the web.

The concept of an endpoint is tightly related to the concept of a URL, as the later is the usual representation for endpoints for identifying the objects or entities of the API. There is not a universal rule which determines how these URLs should be written, as it depends on the specific design of the API. For example, from an architectural point of view, one principle which determines the way endpoints are design is whether the API is oriented towards *services* (processes/computations) or *resources* (data).

An important functionality of an API is managing data, while you can have in-memory data containers to satisfy your data needs, the most reliable and cost effective way to handle any type of information is the *database* (**DB**). Databases come in all shapes and colors, but the most common type of DB are the relational databases. These are structured as tables and relations between them. The most fundamental aspects of databases is that they allow the persistence of data, that is, information that outlives the process that created it.

A classic use of a DB is the management of users in a web application, in this case there could be a *users* table in which the columns denote information of users such as the name, the gender, the age, etc. The rows would then correspond to a particular user's record.

An HTTP cookie (also called web cookie, Internet cookie, browser cookie, or simply cookie) is a small piece of data sent from a website and stored on the user's computer by the user's web browser while the user is browsing.

The Website is in the Eye of the Beholder

- HTML (Hyper Text Markup Language) is the standard markup language for creating web pages and web applications. It constitutes the structure

of a web site.

- CSS (Cascading Style Sheets) is the standard language for describing the presentation and visual aspects of a website.
- JS (JavaScript) is the lingua franca of the web. It handles the behavioral aspects of a website.
- AJAX (Asynchronous JavaScript and XML) is a set of front end techniques to create asynchronous web applications, allowing websites to send and retrieve data from a server asynchronously (in the background) without interfering with the display and behavior of the existing page.
- A grid system is a layout paradigm based on containers, rows, and columns used to determine the placement and alignment of content.
- Responsive web design (RWD) is an approach to web design which makes web pages render well on a variety of devices and window or screen sizes. With the advent of mobile phones and tablets, making your website display well regardless of what size screen or type of device it is viewed on is crucial.
- Search engine optimization (SEO) is the process of affecting the online visibility of a website or a web page in a web search engine's unpaid results—often referred to as "natural", "organic", or "earned" results. The application of SEO in your web application ensures that your website is very easy for Google to crawl so your intended audience can find their way there.

The Tools of the Craft

- A WYSIWYG editor is a system in which content (text and graphics) can be edited in a form closely resembling its appearance when printed or displayed as a finished product,such as a printed document, web page, or slide presentation. The advantage of using these editors is that most of them support a drag-and-drop and clickable configuration paradigm which directly translates to the intended design of a web site.
- A text editor is a program specialized in editing plain text. Editors with good support for programming allow the easy manipulation of code.
- An integrated development environment (IDE) is a program that provides features to computer programmers specifically for software development. An IDE normally consists of a source code editor, build automation tools, a debugger, and intelligent code completion.
- Browser Developer Tools are a set of web debugging tools built into the browser, providing developers with deep access to the running state of their applications.