Vince Vacanti Overview by Steph

The challenge of building your own prototype **isn't** that any individual part is hard. The challenge is that there are so many different components you have to learn about.

So, when a user comes to Yipit, they make a request from our website by visiting a url. That request by the user is sent through the internet to a machine that runs your website. That machine is called a server which brings us to our first component: **servers**.

> **Servers** are machines that have your code, your database and run your website by responding to requests from your users. You don't buy these machines. You rent space on them.

Popular examples: 1and1, slicehost, Amazon Web Services

Learn or hire someone: You want to spend as little time on this as possible. It will have no impact on your ability to get traction and is just something you have to set up and fix when it goes down. **What we did:** We hired someone on a hourly basis, who is now our CTO. We used and still use Amazon Web Services. Just make sure the person you hire lets you know what to do when the server goes down (it will go down)

Now, the server is just a machine. On that machine, is software that actually handles the request from your user and that software is our second major component: **web servers.**

> **Web servers** are the software sitting on your server that is responsible for receiving the request from the user and sending back a response, usually in HTML, that the user's browser will display

Popular examples: Apache, nginx

Learn or hire someone: You also want to spend as little time on this as possible so hire someone to get it set up.

What we did: We used Apache at first, switched to nginx and are now back on Apache. I would recommend Apache since it's more popular. Again, we got this setup on our behalf. Just make sure you know how to restart the server.

Okay, so now that the server got the request that the user needs her New York daily deals, it needs to respond. But, the server doesn't know her deals or how to display the deals in a format her web browser will understand. So, it gets a little help. It sends the request to where all the magic happens and our third and most important component: **web application frameworks**.

> **Web application frameworks** are responsible for receiving the request and generating the html page that will get sent back to the user. They do all the work. This is where the logic of your site will sit.

Popular examples: Django (built in Python) and Rails (built in Ruby)

Learn or hire someone: This is what you have to learn and will be where you spend almost all of your time. If you learn how to develop with these frameworks, you'll be able to build almost any prototype you want.

What we did: I decided to use Python but Ruby is also great. If you have a good friend that's an expert in either one of these, I would just pick which one they know and ask them lots of questions as you learn. I first bought a book on Learning Python. It's not critical you fully understands all the ins and outs of Python. I certainly don't. You just need to know enough (for loops, data structures). I then learned Django primarily from the online django book. Will talk much more about this in a separate post.

Tools: I use Textmate to do most of my programming. I use GitHub to manage my revisions. (Ask the person you hire to set github up for you). While the web app framework will do all the work, it needs a little help because it doesn't actually have the data. For our example, it doesn't have all the new york daily deals. That data sits in our fourth major component: **databases**

Databases store all of the data for your project. Think of them as really large excel spreadsheets with rows and rows of data

Popular examples: MySQL
Learn or hire someone: You should hire someone to explain this to you and set it up. You should learn enough to run some basic queries off the database and alter the structure. The web app framework obfuscates most of the interaction with the database.
What we did: I read this book on learning MySQL. I wouldn't read past page 300 and don't worry if you don't know this stuff to well
Tools: I use Sequel Pro to view the database.

While the web app framework will handle the creation of your html pages that get sent back to the server, you still have to write the templates in **HTML and CSS** which is our fifth major component.

HTML is the format that web browsers expect for web pages. CSS is an additional file that comes along with the HTML that helps to style the html.

Learn or hire someone: You should learn this. It's actually the easiest thing to learn. If you have a co-founder that isn't technical, they should definitely learn this. It's not programming and it will be a big help to have someone else on the team doing this.
What we did: My co-founder Jim learned HTML and CSS. He read and recommends this HTML book and this CSS book. In three weeks, he was ready to go.
Tools: Firebug running in Firefox is your best friend; it's awesome. I also recommend PSD2HTML to get your photoshop files turned into HTML. You can then work off of what they did. **DO NOT BUY DREAMWEAVER**. It's a nightmare-maker.

The last major component is **Javascript**. It's a programming language that runs on your user's browsers. You know when you click on certain websites and the page doesn't reload but a little box pops up or it unhides content? That's using Javascript.

> **Javascript** is a client-side programming language that allows you to manipulate content on your site without requiring the user to reload the entire page. It's not a necessity but it can significantly improve user interface and user experience of your site. There's a library written in Javascript that you should use called **jQuery**

Learn or hire someone: You shouldn't really learn Javascript but rather you should learn jQuery which is a library written in Javascript that makes it much easier to do all the user interface things you want to do on the page. Also, I wouldn't worry about mastering jQuery, just learn enough to accomplish the user interface improvements you are looking for.

What we did: We just read through the jQuery website and did some of the tutorials. It's pretty easy to get going.
Tools: I never found a javascript debugger that I really liked. Firebug's console mode lets you see if there's something wrong.

Development and Production Environment - When you launch your site, you will have a development and production environment.

Development Environment. This is just a fancy way of saying where you work on your prototype that normal users don't have access to. Or even simpler, your laptop. You'll essentially have a working version of your website with a database, your code, etc running on your laptop. I strongly recommend you find someone who knows their stuff (pay them per hour if needed) to set this up for you. Sit right next to them while they do this and ask lots of questions. It should take them around 5 hours and it would take you a frustratingly long time to do it yourself. If you can, I also recommend switching to a Mac. It's way easier to deal with than a PC.

Production Environment. This is where a live version of your site sits where users can access it. You do work on your development environment and then you push it to the production environment where users can see that latest feature you added. Again, you should have the person you hire set this up and the mechanism in which it gets updated from your development environment.

Other Acronyms and Terms You've Heard

If you understand the major components above, you can fit these acronyms in their buckets.

PHP, Java, Perl: Just programming languages like Python and Ruby

XML: Just a format to put data in like HTML. It's just another way servers send data to users. It's typically used for API's.

API: This is just a way for your website to interact with another website. As an example, if you have a website with business listings and you want to show Yelp's reviews on your website, you use Yelp's API. That is, your website sends Yelp a request for the reviews of a business with phone number 555-1234 and Yelp returns an XML file with the reviews. Your site parses that XML file and puts the reviews on your HTML page.

JSON: *Javascript object notation.* This is just another format to put data in that's actually much easier to use than XML. Most API's give you the option of getting back JSON formatted data in addition to XML data.

AJAX: *Asynchronous Javascript and XML*. This is a way for your website to interact with your server without having to reload a full page. On Facebook, when you like someone's status update, do you notice that the page doesn't reload. That's because it's using AJAX. When you click on the "like button", the javascript sends a request to the server to let it know that the status message has been liked without having to reload the page. Lastly, remember that your goal isn't to master these things but to learn just enough to get a prototype up and running that you can iterate on. Once you get traction, you'll be able to bring on awesome developers that will have a mastery of this subject matter.