Dimitrios Vythoulkas

Answer 1

The project I am mostly proud of is the master thesis I had to implement which was a blog application. I'm proud of it because I managed to set up an application using tools ant technologies I had no former experience with. So I taught myself PHP and MySQL and build it in 3 months time. I also had to implement OOP practices. So I consider myself to be a stubborn fast learner.

Answer 2

1) Unclear initial requirements. If requirements are poorly written, unclear, incomplete, too general, or not testable there will be problems.

2) Schedule work and track of Project Timelines. Sometimes a task seeming relatively easy may exceed the allocated time.

3) Having to add new features while development is underway.

4) Testing. Testing is not only to check for a bug free frontend but also testing user experience. It's vital to understand how the user perceives the final product.

Answer 3

1) Environment: According to your final product you should consider the technologies you would want to work with. Is it mobile application development, web app development ext. If you want to develop native iphone apps let's say, then you might want to focus to Objective C.

2) Familiarity: In my opinion one should always start with the technologies that are familiar to him or to the starting team. There is a quote saying that "Anything will work in the beginning" so, dealing with technologies you are familiar to, will result to faster and more efficient delivery of the final product.

3) Maintainability: Maintainability is a huge factor when choosing a technology stack. As product is growing more challenges appear. Being able to adapt and upgrade your product is crucial. So you should choose technologies that allow you to maintain your code easily

4) Performance: Some technologies are better at performing certain tasks than others. Also some help to scale more efficiently.

5) Community: Active communities support fast adaptation of software and quick bug fixes. So if you are dealing with limited resources the community support should be strong.

Answer 4

UX, most of the times, is perceived to be the techniques used for the better visual representation of a software product. Visualization is of course part of the UX process but not just that. UX focuses on how a user reacts with a product and how this

reaction can be more efficient and enjoyable. In that aspect UX tends to be more "human-centric". In order to achieve this, UX depends on testing the product during production with real users.
Using UX principals and UX tests can contribute to the implementation of a product in a positive way. Requirements might be reconsidered and goals might become more clear. Testing product with real users make the final product more appealing to customers and overall a greater experience.

Answer 5
There are many different tools that would replace others so here my answer would have to be git. I cannot imagine developing something without version control. The reason is that it allows me to be bolder with newer versions and not being afraid to make mistakes while developing. With git you are always sure you can return to a previous version. With how git collaborates with gitHub is also great. Sharing, contributing and merging code is much easier.

Answer 6
An element can have only one id and each page can have only on element with this id. An element can have multiple classes and you can have many multiple elements using the same class.
It's a good idea to use id when you want to specify a certain unique behavior to a single element on the page. It's a good idea to use class when you want to have the same behavior to multiple elements on the page.
It is a general rule in performance optimization to make your rules as general as possible. Saying that, you should avoid using id when you could use class

Answer 7
Javascript does not use Classical Inheritance. Instead it implements inheritance through Prototypal Inheritance. So every object in Javascript has this prototype property that is a reference to an object called proto{}. You can directly set the prototype which is greatly discouraged like the example:

```
var person = {
   firstname: 'Johnny',
   lastname: 'Bravo'
}

var john = {}

john.__proto__ = person;
```

if we console.log(john.firstname) we 'll get Johnny as a result since it is inherited from person.

Or you can create am object constructor function and use the prototype method that all functions have in Javascript. So when you use the new operator this creates an empty object and sets the prototype of that empty object to the prototype property of

the constructor function called (functions being objects in Javascript). So here is an example also using the prototype property all functions have:

```
function Person(firstname, lastname) {
    this.firstname = firstname;
    this.lastname = lastname;
}

Person.prototype.getfullName = function() {
    return this.firstname + ' ' + this.lastname;
}

var john = new Person('John', 'Doe');
console.log(john.getfullName());
```

We set the prototype of john object to the prototype property of Person constructor so the john object inherits allproperties and methods of the constructor. In other words john.__proto__ points to Person.prototype. if we use the console and enter john.__proto__ the result is Person {} which proves the point.

Answer 10)
1) I have a strong feeling I qualify for the requirements you set for the certain position.
2) I consider myself friendly, easygoing and great to communicate with, and I value teamwork.
3) The 3rd reason is quite personal actually. I feel like I have lost time in my life making bad career decisions. Getting this job will be the next step for me, and a chance to prove things to myself and to others. So I'm full of energy and ready for hard work in order succeed there.