Progressive Enhancement Vs. Graceful Degradation

Posted At : May 18, 2011 2:20 PM | Posted By : Mark Kruger
Related Categories: Project Management

I was in a fascinating seminar by Barney Boisvert about programming with "progressive enhancement". It's something I've thought about on the edges of my mind, but Barney (who is funny and really tall) put it so clearly that I could not help but blog about it. To get us in the right frame of mind, let's check in on project launch meeting at CF Webtools. Our new client, Joe Dreamer, has brought his latest idea to the table. He wants to build a special "Barbie's Facebook" - a social networking site where young girls can register their Barbie dolls and flesh out their lives in the virtual world. NOTE: This funny example came out of the Muse' head. I know of no such project (but now that I think about it....). Anyway, let's check in on our project launch meeting.

The Meeting

Meeting attended by the Muse, Laurie (our gifted project manager), and 2 developers - *Pinky* and *The Brain*.

- **Muse:** All right we have our requirements document. It's time to get to work. Laurie?
- **Laurie:** Ok... we need to flesh out our functional specs and make some decisions. Let's talk about the "Barbie's Life" page. According to Joe this page is supposed to be a sort of personal Facebook page for the doll in question. It should have links to the profile, messages etc., but the main purpose of this page as a communication's "wall" for Barbie's status.
- **The Brain:** (sonorous Orsen-Wellion tone) Well of course we'll use jQuery to enhance the user interface.
- **Pinky:** (always merrily) Why we gonna do that Brain?
- **Brain:** Because it's ubiquitous and well supported and provides a library of common functionality.
- **Pinky:** Oh... **narf**. I thought it was a part of your brilliant plan to take over the world.
- **Brain:** (hastily) Can we talk about this later? .... anyway I see lots of possibilities for enhanced Web 2.0 interaction here.
- **Laurie:** Ok... let's get started. Here's what our creative folks came up with for a story board.....

A month later. Joe Dreamer has taken a peek at "Barbie's Life" page on staging and called a meeting.

- **Joe:** (aggressively) Ok... I *love love love* what I'm seeing so far. You guys are fabulous, awesome, brilliant even! Everyone I show this too practically faints in ecstasy.
- **Brain:** (wryly) Do tell.
- **Pinky:** **narf** It makes me swoon too
- **Joe:** ....I only have a few problems with it now that I've had a peek at it.
- **Brain:** (testily) Problems?
- **Joe:** Oh nothing major I promise. See, I was over at my cousin Barney's house. He's a huge Barbie fanatic. He has a blow up Barbie in his closet... at least I think it's a Barbie... well it's blond.... Anyway he loves this idea. We pulled up the test site on his computer and I noticed a few things didn't work. I couldn't get the

profile to save for example. and the dialogue boxes don't come up. And there's a little yellow exclamation point in the lower left hand corner.
- **Brain:** Pinky - are you pondering what I'm pondering?
- **Pinky:** I think so Brain, but how do we get the frogs out of your pants when we're done?
- **Brain:** No... I was thinking we should ask what sort of browser Barney was using.
- **Pinky:** You mean the viewy thingy?
- **Brain:** (sighing) Yes Pinky - the viewy thingy. Joe, what sort of viewy thingy... (slaps forehead) .. what type and version of browser was he using?
- **Joe:** I believe it was Internet Explorer 4. Does that sound right.
- **Brain:** Unfortunately yes. Come on Pinky - we have work to do tonight.
- **Pinky:** And what we gonna do tomorrow night Brain?
- **Brain:** Not now Pinky.

Typical Process

Buried in our little sketch is the outline of one of the *typical* processes we go through when designing web applications. Note: I'm *not* talking about agile, scrum, waterfall or any other project *management* idea here. I'm talking about our choices when we actually *build* the application. Now I know many of my smart and savvy readers will say "browser compatibility should be a part of the spec." And indeed compatibility and testing usually are part of the spec. But our actual coding *choices* tend to operate in a way that is not productive. We typically build our sites as a pyramid but from the top down. This is where Barney really helped clarify it for me - using small words so I could understand. He defined the pyramid and suggested that we start with the base and work our way up to the ideal. Before we get there however let's discuss his phrase for the way we *often* do things - ***graceful degradation***.

Graceful Degradation

Here's what *often*happens. We start with a storyboard, a workflow and some design concepts. These are important because they are what the *stakeholders* are concerned about. But it's important to note that while story boards, mockups and design concepts are foundational to our project planning, they are in fact representative of the *ideal* that we are trying to accomplish. In other words they picture the top of the pyramid. But they fix in our minds a vision of how the end product should look and behave. Since we are developers we have the latest Firefox, Safari and Chrome browsers - and most of us open IE only to test (and occasionally de-test). Our goal is to build our application to the highest possible standard - to build that ideal we see at the top of the pyramid. That means nice Web 2.0 interactions, client side error traps, rich images, transparency, dialogue boxes, jQuery grids and panes etc etc - the works. In a single phrase our approach is to "build the ideal first".

Then, having constructed the "ideal" that matches our deliverable we are confronted with nit-picky reality. There are still users on IE 4 using Windows 95 or 98. There are people running security software or proxies or adware that screw up your fancy under-the-hood jQuery requests. There are folks who can't read your artsy grey 8pt font (like the Muse for example). Elderly women have nephews who work at the geek squad and are therefore computer geniuses who disable JavaScript (You don't need that Aunt Pheobe... it's dangerous). There are a hundred reasons why your code *won't* work in many environments.

Often our approach (the "graceful degradation" approach) is to build the ideal and then handle the exceptions as they arise. So we build a fancy pants interface, test, congratulate ourselves and then *after the fact* we have to account for Aunt Martha's custom Gateway IE 4.0. It's either that or teach her how to use an new operating system and browser and we are too conscientious to do that to the local suicide rate. The result? Our code ends up with hacks and work arounds and widgets all designed to solve *specific issues* we encounter after we build the ideal.

Note, we are talking about web applications here. Rich Internet applications (RIA) using something like Flex or Silverlight are designed for a single runtime that is *required* - so there's no degradation at all. It either works or it doesn't.

Progressive Enhancement

Contrast what we have just observed (build the ideal and then degrade it for exceptions) with the idea of "Progressive Enhancement". In this model we create the baseline functionality. We think about accessibility, basic functionality without JavaScript, basic design elements that work in all browsers etc. Then at some point we add "enhancements" to bring our application up to the "ideal".

Recently my wife and I went to an upscale steakhouse (**Mahogany's**) for a meal. You know you are in a luxury restaurant when the hostess looks carefully at your wife and then substitutes a black napkin for a white one to better match her outfit. It seems like there were about 3 wait staff who hovered a few feet away and the minute I dropped a crumb on the table they were there to sweep it up with the little crumb blade thingy. This restaurant had a mostly a la carte type menu. You choose your meat first (that's why you are there after all). Then you add a side or two and a bottle of wine and possibly desert. The focus of the meal is the meat. The waiter spends 2 minutes just describing the process of selecting and preparing your steak - "First we raise a cow lovingly with chocolates and bubble bath, then we slaughter it carefully with a 200 year old Katana, then we yada yada yada and....", you get the idea. It's clear they are quite proud of their meat - which is after all an American tradition (at least among men). The side dishes and salad were great too. In fact I had a "salt encrusted" baked potato that was absolutely delectable. But the sides only *enhance* the main event - the main course. In this place steak rules and it's very very good. If all you ordered was the steak and a glass of water you would have a memorable meal - but the sides, salad, appetizers and wine take it from being a meal to a grand evening event.

If you are following my metaphor, I mean to say that using progressive enhancement there is a basic utility that should be considered and created *first* in each web application. Think of it as the "main course" - the basic functionality you are trying to deliver *without the bells and whistles*. This baseline should be able to provide the functionality of the application to the lowest common denominator device or browser you intend to support. How do you get to the "lowest common denominator"? Here's where Barney had some specific advice for us:

- Accessibility - make your first draft follow the accessibility rules. This means friendly to screen readers, effective (and semantic) labeling of elements and images, content centric layout etc. There are a many online resources that will help you flesh out your knowledge of accessibility rules and practices. Following these rules will benefit the lowest common denominator browser environments *and* make your site accessible to those with visual disabilities. It's a win win.

- JS Disabled - Do your first draft with *no JavaScript*. Test it with JavaScript disabled and make sure it still provides the basic functionality you are after. An added benefit of this is that it will force you to include server side validation of input variables. Too often the Muse sees sites with client side only validation. It's worth remembering that while client side validation *does* enhance the user interface it does *nothing* to enhance your security. If you are truly concerned about double checking the user inputs to insure they match the pattern or rule required then server side validation is absolutely a must. Of course if you programming your first draft without JavaScript then you pretty much *have* to do it this way anyway - another win win.
- CSS - Uh-Oh... here's where you have a bit of a decision. You have to compromise your CSS so that browsers who do not support the full CSS set will still render your content. Depending on how far down in the weeds you want to go with this you may not be willing to do that. most browsers *do* support a broad set of CSS. The good news is that while CSS may not be supported or fully supported by the "lowest common denominator" browser, it will likely be *ignored* by that browser. So your page will render... it just won't look as you intended. The trick here is to maintain some order and functionality even if it's butt ugly.

Step 2 of PE - Enhancement

Once you have your main course it's time to start adding the sides. Semantic Markup, HTML 5, fancy pants CSS, and of course the rich tapestry of jQuery (or one of the other JS libraries) that allow for page injection, DOM manipulation, and functional targeting. Continue assembling the pieces until your meal has turned from a simple main course into an evening event that reflects your "ideal" - that thing you sold the stakeholders on. But the good news is you will have far less to do now for browser compatibility and you won't be caught with your pants down on a browser that doesn't work. They will all work (although they will not all reflect the ideal).

Overall I like this approach. There are times when building for graceful degradation (building the ideal first) is still ok, like an application for a well-defined audience. If you *know* all your users are internal and have a certain browser, or can be required to use a certain browser, then you can program to the capabilities of that browser. But for public sites that are designed to cast the widest possible net and have a wide demographic (or a demographic of "late adapters") the progressive enhancement approach will mean better regression and less headaches overall.

I'll leave you with this quote from Barney's presentation (which can be found at this link).

> *With Graceful Degradation you build it awesome and pray it works. With Progressive enhancement you build it to work and then make it awesome.*

I'd say that sums it up nicely.