Patterns in the world of Web Application Development

Philip Roche (student number: 08299234 email: phil@philroche.net)

In the world of software, a pattern is a tangible manifestation of an organization's tribal memory.

-Grady Booch

Most of my work involves web development with a strong focus on front end development using frameworks to do most of the back-end work. Just as a software developer encounters recurring problems so too does a front end web developer.

This paper will be discussing the patterns, standards, conventions and best practices that I have come across in different areas of development, and that I now adhere to and use in my work. Each of the concepts or technologies are a direct result of someone seeking a solution to a recurring problem, just like design patterns in software.

Web Standards

Web standards is a somewhat misleading term as there are no defined standards only recommendations by organisations like W3C. These recommendations cover

- Valid HTML/XHTML code
- Semantically correct code
- Separation of content (HTML/XHTML), presentation (CSS) and interaction/ behaviour (JavaScript)

The problems that these recommendations are trying to solve are many, the main ones being maintainability of work, dealing with web browser inconsistencies and making websites accessible to users that are not using a regular desktop web browser (Web Accessibility).

During the early 90s there were two competing browser: Internet Explorer and Netscape Navigator. These two browsers each had different takes on how certain HTML elements should look and behave, they had their own proprietary tags and technologies. This was an absolute nightmare for developers. Standards like HTML4, XHTML CSS and Javascript were proposed and slowly the browsers began to develop to those standards/ recommendations. It is only now that (with the exception of Internet Explorer 6) developers can safely ignore the inconsistencies and develop to the standards and be sure everything will work as it should.

Another problem that kept cropping up was the difficulty in maintenance of websites. The main reason for this was the mixing of presentation, content and behaviour. Having inline Javascript and CSS and font tags in your HTML is definitely going to cause problems down the line. One of the goals of Web Standards is to solve this problem by separating presentation and content and behaviour.

What this means is that the HTML will contain only content, the CSS will apply the presentation and the Javascript will add the behaviour. This makes it very easy to maintain.

Web Semantics

'Web Semantics' is a term used to describe content with metadata. Previously when building a web page you would use tags like strong, b, i, font etc. These tags give no real meaning to the content contained within them making them very hard to understand, were you to try and parse that HTML.

The goal is to try and describe the content you are serving. Proper use of the standards (XHTML) will aid in this by using h1 tags for your main heading etc. But we need to be more granular about the description of our data. Best web design practice is to consistently markup common architectural components in the same way and projects like 'Web Design Practices' are there to solve the problem of inconsistent markup. If you have an architectural component (like a navigation section) on your page, you should use the markup to describe it as such. This applies to branding, footer and search etc.

There are ways to markup up non-architectural content so as to describe it in more detail. Microformats are a solution to the problem of inconsistent markup on commonly occurring data, for example the Microformat hCard is designed to help describe address or contact details. The hCalendar is to aid in the description of calendar events. XFN (XML Friends Network) lets you describe which links on your page are links to people you know, have met etc. All these “patterns” make a web page a lot more machine readable too which is the desired result of the semantic web.

Machine readability is the eventual goal but it will take a long time until all developers are adhering to the same recommendations and markup patterns. Standards like RDF (Resource Description Framework), RSS (Really Simple Syndication), ATOM , OPML (Outline Processor Markup Language), FOAF (Friend of a Friend), OpenId (dealing with digital identities), Oauth (protocol to allow secure API authorisation), XMPP (Extensible Messaging and Presence Protocol),APML (Attention Profiling Markup Language) and the Data Portability initiative are an attempt to standardise the publishing of machine readable content and data.

These standards/ conventions are guides on how to describe common content and data and each solve a problem that developers have been having for a long time.

Applications exist that implement and use this data, News readers like NetNewsWire and Google Reader read OPML, RSS and ATOM feeds. Social network sites use the APML, OpenId, Oauth and FOAF standards. Jabber chat clients and other online chat clients use XMPP.

The Data Portability initiative (http://www.dataportability.org/) is a portal to all these varying standards and conventions that have been proposed. It has been very successful and more and more developers are implementing the proposed guidelines.

Data Access

Most web applications require a database and as such a data access layer – to write the data access layer each time would be very tedious. Projects like the Subsonic Project, SQLObject, SQLAlchemy and Propel exist solving the problem of creating data access layers as they do all this for you. These projects also abstract the data access so that you can use any number of databases which is very useful.

Template Engines

There is also the issue of web page generation, should a developer have to deal with the generation of each page and each of it's components? Template engines like Cheetah, Kid and Smarty solve these problems. As these projects emerged it became apparent that another recurring problem needs solving – namely how to separate the different layers (data access logic, business logic, and presentation logic) of a web application, much the same as the need for separation of presentation, content and behaviour in web standards.

The Data Access can be handled by the projects mentioned previously and the front end rendering of pages handled by the template engines – but how do you join them together.

Separation of Layers

The most common design pattern for separating the three layers of a web application the MVC architectural design pattern. MVC is an acronym for Model View Controller. The most popular application frameworks that use this pattern are Ruby on Rails and the ASP.NET MVC framework. In the MVC pattern, “Model” refers to the data access layer, “View” refers to the part of the system that selects what to display and how to display it, and “Controller” refers to the part of the system that decides which view to use, depending on user input, accessing the model as needed. The real power lies in the combination of the MVC layers, which is something that the frameworks handle for you.

The Python framework Django uses a variation on MVC called MTV. “M” stands for “Model,” the data access layer. “T” stands for “Template,” the presentation layer. “V” stands for “View,” the business logic layer. This layer contains the logic that accesses the model and defers to the appropriate template(s).

Using frameworks like these makes application development more about the logic of the application rather than dealing with the implementation. Simple application like a blogging system or a wiki can be rapidly built and deployed and because of the separation of layers it makes it a lot easier to scale such applications by adding database clusters, load balanced servers etc.

Transfer of Data

Most web applications have some sort of CRUD (Create, Read, Update and Delete) operations. Patterns exist to standardise these operations.

The REST (Representational State Transfer) architecture aids in all these operations. REST suggests the use of web services and the use of the HTTP verbs POST (for create operations), GET (for read operations), PUT (for create and update operations) and DELETE (for delete operations).

REST also promotes the use of readable (pretty urls) urls. The data sent to REST services can be of any type XML, JSON or YAML. Other methods exist solving the same problem like SOAP and XML RPC but REST or “RESTFul services” are the most popular especially in the building of application APIs.

The Atom format is an extensible format and one of it's extensions is the ATOMPub (or APP) application-level protocol for publishing and editing Web Resources.

Protocols like ATOMPub and architectures like REST will hopefully bring an end to custom created CRUD applications introducing a pseudo-standard for applications like that.

Responsiveness of web applications

A problem that faced developers was building web applications that would feel like desktop applications without having page reloads after every action.

Javascript and Ajax (Asynchronous Javascript and XML) are now used to achieve this effect. Essentially it means that instead of reloading the whole page after every action you only need to update part of the page making the application a lot more responsive and usable.

Instead of the browser making a hit to the server for the whole page, Javascript uses the XMLHttpRequest method to publish or request data. This data is then served in either XML, JSON (a plain text serialisation of data which is less verbose than XML) which the Javascript parses and uses to update the page. This is extremely popular method of building applications now but it does have a requirement that the user must have Javascript enabled which goes against Web Accessibility.

To solve this the concepts of “graceful degradation” and “unobtrusive Javascript” were introduced where an application is built without any of these enhancements with the enhancements to be added post page load using Javascript. This means that the requirement for Javascript is removed and the application remains accessible to those using clients without Javascript.

Front end Libraries

As mentioned before, Javascript is used a lot in application development, as such many developers would be writing the same functionality again and again. Libraries were written to make it easier for developers. Combined, libraries like JQuery, Prototype, YUI, Mootools and MochiKit provide solutions to practically any Javascript problem a developer might have encountered. JQuery is the most advanced providing simplified methods for nearly all Javascript operations.

The same happened with CSS too as developers would find themselves writing the same CSS again and again. Frameworks like Blue Print CSS, YUI, YAML and Tripoli provide ways to rapidly develop CSS layout without having to write each line of CSS.

Interaction/ User Interface design

A well designed and intuitive user interface is key to an applications success. A lot of research has been carried out on how users interact with user interfaces. Heatmaps that analyse user interaction can be used (like ClickHeat from LabsMedia) to help in optimising your design.

Tried and tested solutions to common design problems are available to use. Examples would be site navigation, display of search results, a checkout process, form layout, captcha for forms, calendars, shopping carts, product listings, language selection, site maps, tag clouds, wizards, bread crumbs and many more.

Sites like webdesignpractices.com, welie.com and ui-patterns.com provide comprehensive lists and example of visual design and user interface patterns.

Conclusion

As with design patterns in software development, it is up to the software engineer to recognise problems and be aware that there are patterns that can be applied to solve these problems, so too it is the onus of the web developer to be aware of the interaction and user interface, semantic patterns etc. and to apply them appropriately to problems they encounter.

I have been involved in the web development and standards community for a few years now and it is a testament to “open source” philosophies that the patterns described above have emerged – In Booch's quote above – it is this community that are the organisation and the standards, patterns and best practices that are their tribal memory.

Resources and references:

- Microformats (http://microformats.org/)
- RDF (http://www.w3.org/RDF/)
- RSS (http://www.rssboard.org/)
- ATOM (http://www.atomenabled.org/)
- OPML (http://www.opml.org/)
- FOAF (http://www.foaf-project.org/)
- OpenId (http://openid.net/)
- Oauth (http://oauth.net/)
- XMPP (http://xmpp.org/)
- APML (http://www.apml.org/)
- Data Portability (http://www.dataportability.org/)
- SubSonicProject (http://subsonicproject.com/)
- SQLObject (http://sqlobject.org/)
- SQLAlchemy (http://www.sqlalchemy.org/)
- Propel (http://propel.phpdb.org/trac/)
- Cheetah (http://www.cheetahtemplate.org/)
- Kid (http://www.kid-templating.org/)
- Smarty (http://www.smarty.net/)
- Django (http://www.djangoproject.com/)
- Ruby on Rails (http://rubyonrails.org/)

- ASP.NET MVC (http://www.asp.net/mvc/)
- ATOMPub (http://bitworking.org/projects/atom/rfc5023.html)
- REST (http://en.wikipedia.org/wiki/Representational_State_Transfer)
- JQuery (http://jquery.com/)
- Prototype (http://www.prototypejs.org/)
- Mootools (http://mootools.net/)
- MochiKit (http://mochikit.com/)
- YUI (http://developer.yahoo.com/yui/)
- BluePrintCSS (http://code.google.com/p/blueprintcss/)
- YAML (http://www.yaml.de/en/home.html)
- Tripoli (http://devkick.com/lab/tripoli/)
- Web Design Practices (http://www.webdesignpractices.com/)
- UI – Patterns (http://ui-patterns.com/resources)
- Interaction Design Patterns (http://www.welie.com/index.php)
- YUI Design Patterns (http://developer.yahoo.com/ypatterns/)