Web-based technologies for the modern astronomer

Davide Ricci*, Luciano Nicastro

INAF/Istituto di Astrofisica Spaziale e Fisica Cosmica, Bologna,
Via Gobetti 101, 40129 Bologna, Italy.

December 12, 2012

Modern web-based technologies are a unique opportunity for the astronomical community to simplify and modernize their work. Building webtools and rich internet applications can be, for example, useful in traditional, remote and robotic observatories to simplify data handling, management, reduction and access. We present an overview of the top internet technologies and we explain how to apply them in astronomy. We also provide very basic code examples and invite the inrested reader to try and modify them to become more familiar with the issues discussed in this document.

Contents

*Post-doc in the framework of the GLORIA project (GLObal Robotic-telescopes Intelligent Array)

1 Introduction

Advanced web technologies, born to manage applications and provide services through the internet, are growing quickly to offer new capabilities and an always better user experience. Web 2.0 sites are typical examples.

A researcher spends a lot of time in front of a computer, programming and running stand-alone software and expecially using the internet: in fact, to accomplish her/his work, it is necessary to surf the web looking for basic information, catalogs, reference papers, solutions for programming issues. Moreover it is very common the use of webmails, translators and social media.

For these reasons, a researcher is not simply a computer power user, but also and expecially an experienced internet user.

Being such kind of a user means to have developed a "taste" for web design that overrides the pure concept of graphic presentation, because it is simpler and faster to find something on the web if the site is user friendly, or if the application is well conceived. Designers working for private companies know very well this problem: they study how the users (or potential clients) react to a banner, how many times they spend on a page, where they click and why. Furthermore, they design the internet pages to enhance the user experience and follow what we have learned to consider a "natural" action: if we use the mouse wheel on a text, we expect to scroll it; if we use the same wheel on an image, we expect to zoom it.

The aim of a website or a web application for astronomical researches is not to sell a product, but like in the previous example, to accomplish tasks in a simple and quick way. For example: browsing a database to search for information about astronomical objects, catalogs and images, scheduling and even launching a remote observation, performing simple statistics on a dataset, reducing the data and inspect for the result, saving the results or a summary in several formats for further purposes/analysis.

A typical problem is that of managing a large amount of data produced every night by a set of ground based optical or infrared telescopes. But large amount of data are also produced by space-based telescopes, in the same or in other wavelengths. In this framework, it is natural to associate this matter with the concept which is behind the Virtual Observatory [1] [2] [3].

It is for these reasons that we suggest to develop modern web-based instruments to manage all these tasks, in particular to browse in a graphical and intuitive way data archives and processing results.

[1] http://www.ivoa.net/
[2] http://www.euro-vo.org/pub/
[3] http://skyview.gsfc.nasa.gov/

2 The actual instruments

Seen through the eyes of an external observer, the computer instruments that astronomers use everyday to accomplish their work are a jam of relatively new web sites (ADS[4], astro-ph[5]), old-fashioned sites (Vizier[6], Simbad[7]) incepted with heavy web Java applications (Aladin[8]), custom routines personally programmed to solve very specific tasks, and finally a bulk of freakish stand-alone outdated and counter intuitive programs or suite of programs (ds9[9], IRAF[10], MIDAS[11], paw[12], smongo[13]).

But why are we glued to such a **steampunk** mess? For one main reason:

it works: it is better to use something that already exists, works, and can be used to produce scientific results, instead of spending a lot of time developing a new tool that maybe will save time in the long term but surely will postpone the results;

and other but not less important reasons:

tradition: people are used to those programs and "pass down" them. For the teacher this behavior has the advantage that he will spend a relatively short time to train a student to use a well known and trusted software tool;

support: some of these programs are exremely stable and still supported[14]. Although they were written in Fortran77 thirty years ago or with a user interface in Tcl/Tk, these tools are platform-independent and largely supported from old Solaris machines to the latest Windows release.

However, this does not apply to all these programs: most of them are abandoned (the last release of paw dates 2002 and the first one available on the internet[15] is dated 1988), hard to compile, expand and mantain. They are complex and difficoult to use, they require specific or outdated libraries, and they were designed for another era (command-line terminals, single processor machines).

In this framework, we would like to emphasize the effort of the virtual observatories which are targeted to put together and make accessible the astronomical data, both with stand-alone and webtools, and most of all to promote common protocols and exchange formats (keeping in mind the cautions of Fig. 1).

[4] http://www.adsabs.harvard.edu/
[5] http://arxiv.org/archive/astro-ph/
[6] http://vizier.u-strasbg.fr/viz-bin/VizieR
[7] http://simbad.u-strasbg.fr/simbad/
[8] http://aladin.u-strasbg.fr/java/nph-aladin.pl
[9] http://hea-www.harvard.edu/RD/ds9/site/Home.html
[10] http://iraf.noao.edu/
[11] http://www.eso.org/sci/software/esomidas/
[12] http://paw.web.cern.ch/paw/
[13] http://www.astro.princeton.edu/~rhl/sm/
[14] The last version of ds9, which provides for the first time 3D features in the image management, was released some months before the compilation of this document, (summer 2012).
[15] http://wwwasd.web.cern.ch/wwwasd/cgi-bin/listpawnews.pl

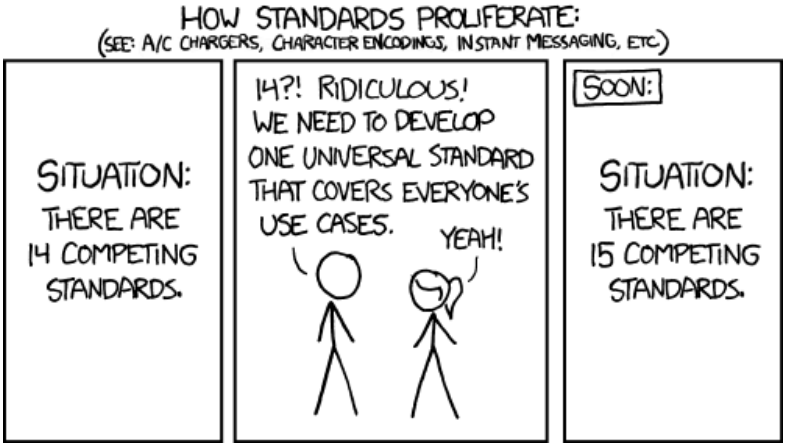

Figure 1: The risks deriving from suggesting a new approach. Credits: xkcd.

3 Web approach

The main advantage for switching from stand-alone to web-based applications is that a web page can be thought as an already accepted universal standard to interact with a computer (referring to the comics in Fig. 1, the web page is one of the initial 14 standards, and it is almost not in competition with other standards). Such a "site" hides to the end user all the low-level components that can evolve independently. In particular it provides:

easy access: every device (pc, laptop, tablet, smartphone, smart tv) has a browser and it works more or less the same way everywhere in terms of interactions and page rendering. A remarkable exception is represented by Microsoft Internet Explorer, due to the fact that it was largely dominant until some years ago, so it had no interest in developing and accepting shared rendering standards. With the success of Firefox and Chrome (along with Safari and Opera), Internet Explorer is losing more and more users[16], and it is accepting more and more standards;

user experience: the components of a "website" are well known (hyperlinks, checkboxs, buttons, etc.), and everyone knows how to browse and work on it even without any kind of training. Moreover, a skilled user is ready from the beginning to use shortcuts, tabbed navigation and additional features to speed its work;

modularity: behind the web layer, any kind of technology could in principle be used to manage the various tasks;

customization: if the application is well programmed, the effort to adapt it to a specific situation (from the needs of a color-blind user to the configuration for a cluster of machines) is minimal.

[16] http://www.w3schools.com/browsers/browsers_stats.asp

4 Real-time layer

Our concept of web application is composed by several modules, or layers. In particular, the user layer, that provides interactions in real-time via the interfaces, should use the top of current web technologies. Here follows a simple overview.

4.1 html5

Web pages are written in `html`, a markup language such as LaTeX, which is interpreted by the browser to show the content as we are usued to see. The last revision of this language is the version 5, and it is still under development. The standard, called `html5`, is foreseen to be released on July 2014. An example of `html` code using `html5` features can be seen in Listing 1.

This new version offers several new features[17], most of which are already supported by the major browsers with the exception of Microsoft Internet Explorer[18].

As the development of the standard is still *in fieri*, we foresee that at the time of the first release, almost all the browsers will be fully compatible.

We investigated in some details and tested, probably for the first time, the possibility to take advantage of these innovations in the framework of the astronomical research, which makes a large use of internet and browsers, as we have explained above.

Only a few of the new features are not directly interesting for our purposes (for example some semantic tag), but even in this case their use can turn out to be useful in a wider context. Some of them are moderately interesting for the management of a web form or in general for input fields, for example:

input types: `html5` provides a set of new attributes for input forms such as `datetime`, `datetime-local`, `number`, `range`, `url`, `email` that could help to check client-side the input values for a query in a database. Moreover, several new touch devices (such as tablets) use these attributes to show to the user the more suitable keyboard layout (alphabetical, numerical, custom-defined);

output tag: `<output>` represents the output of a calculation and could be useful to mark a result in an application;

form attributes: new attributes such as `step` (defining the allowed intervals of an input field), `required`, and `pattern` (which uses regular expressions) could help to avoid errors in the submission of a form in the context, astronomy, where the nomenclature is sometimes very complex.

These new improvements are also useful to avoid server-side controls or even client-side script checks; it is the browser itself that performs this task.

Other moderately interesting components of `html5` are the new media tags and the semantics of the content management. They resemble some of the very well known LaTeX commands and environments that astronomers use everyday:

[17] http://www.w3schools.com/html5/default.asp
[18] http://www.findmebyip.com/litmus/

```
<!DOCTYPE html>
<head>
  <meta charset="utf-8"/>
  <link rel="stylesheet" href="mystyle.css" />
  <script src="http://code.jquery.com/jquery-latest.js"></script>
  <script src="http://d3js.org/d3.v2.js"></script>

  <body>
    <h1>mypage</h1>
    <h2>Test of html5, css3 and jquery capabilities</h2>
    <p>Try to submit an empty form, or an invalid email.</p>
    <form>
      <label for="mytest">email: </label>
      <input id="mytest" type="email" placeholder="a@b.com" required />
      <button>Submit</button>
    </form>
    <strong>Here the result will be shown</strong>

    <h2>Test of svg and jquery capabilities</h2>
    <script src="myscript.js"></script>
    <p>Click on the circle to change its color.</p>
    <svg id="circletest">
      <circle cx="20" cy="20" r="20" style="fill:gray" />
    </svg>

    <h2>Test of svg and d3.js capabilities</h2>
    <script src="myplot.js"></script>
```

Listing 1: An example of `html` code containing elements of `html5`. It calls `javascript` and `css` files which will be listed and described in the following sections. We suggest to copy the content of this code in a simple text file and to call it `mypage.html`. Opening the file with an internet browser (normally it is sufficient double-clicking on it) it will be possible to see its content.

semantic tags: the `<article>`, `<section>`, `<figure>`, `<figcaption>`, `<summary>` and `<details>` can help astronomers to present their work following the scheme already used for their scientific publications, and ad-hoc tools could be designed to produce with no efforts online or printable telescope logs, department bulletins, conversion from/to LaTeX, getting through creepy UNIX commands such as `textohtml`;

mathml: this technology provides an `xml`-based markup, already supported by major browsers excepted Chromium, to write high quality mathematical equations. Even now, browsing phisics or astronomy wikipedia articles, the formulas are generated as gif images by the server, if a LaTeX source is provided. `mathml` override this limitation by directly rendering in the text mathematical elements, signs and symbols as astronomers are used to see in their documents. The markup is very heavy, but several converters from/to the LaTeX syntax are already available through the internet[19].

media tags: the `<audio>`, `<video>` and `<track>` will provide a valid cross-browser alternative to flash-based plugins for the online access of multimedia content as conferences and talks, and most of all to control cameras of a remote or robotic telescope. There is no need for an external plugin to see or hear the media content. The browser is responsible of the audio/video rendering;

streaming: the `getUserMedia()` method, used together with the `<video>` tag, will be a valid alternative for the streaming in live conferences without custom software such as Skype;

geolocation: the communication of the user position with a simple click can be used for survey statistics and to perform several checks in a telescope remote management system.

Finally, we present the several `html5` feautures which we consider as the most interesting in the framework of astronomical research[20]. In particular:

application cache: it offers an opportunity to create an offline version of a web application. It can be used to reduce the server load, as the browser will only download updated or changed resources from the server. This is of particular interest in the context of astronomy, telescopes being often in places where bandwidth and traffic are subject to restrictions;

local storage: web pages can store data locally within the user's browser and overriding the limitations of the cookies. Used together with the application cache, this feature will be useful for data management and reduction: the progress of the work is trustly saved on the client at disconnection and it can be continued offline thanks to the application cache; it will be eventually resumed at the next connection;

[19] http://latex2mathml.freewebmaster.fr/demo.php
[20] http://html5demos.com/

```css
body{
    width: 50%;
    height: 450px;
    border: 1px solid gray;
    border-radius: 15px;
    margin: 10px 25%;
    padding: 20px;
}
form{
    border: 1px dashed gray;
    padding: 5px;
}
strong, h1{
    color: gray;
}
h1{
    position: absolute;
    right: 25%;
    -webkit-transform: rotate(-45deg);
       -moz-transform: rotate(-45deg);
        -ms-transform: rotate(-45deg);
         -o-transform: rotate(-45deg);
}
h2{
    font-size: 1.1em;
    text-shadow: 5px 5px 5px rgba(70,70,50,0.4);
    color: teal;
}
#circletest{
    width: 40px;
    height: 40px;
}
.okmail{
    color: green;
}
rect:hover, #circletest:hover{
    opacity: 0.8;
}
```

Listing 2: An example of `css` code containing elements of `css3`. We suggest to copy the content of this code in a simple text file named `mystyle.css`. If the file is placed in the same directory of the `mypage.html` (see Listing 1), it will be possible to see the style changes in `mypage.html` by opening this latter in a browser.

server-sent events: this technology provides the possibility to obtain real-time events as emitted by the server without a specific request from the client (e.g. with a click, a mouse movement, a polling or a websocket). It is a kind of communication that reminds the old concept of television. In observational astronomy it is for example useful to continuously monitor atmospheric parameters in a local or remote telescope site, or to follow the evolution of the light curve during a transient event such as a planetary transit or a microlensing event; in general it is useful to perform any kind of real-time control without specifically asking the browser to contact the server to receive these information;

canvas: this is probably the most important innovation for our purposes, as the `<canvas>` element allows to draw graphics and display data, on the fly, on a web page. It can be used for dynamic plots, for displaying FITS images or, for example, to superpose to the FITS file an additional layer with labels and marks. The potential is huge and on the internet several different application can be found that show its adaptability;

svg: inline vector graphics management can be used for plots and data representation. Simple or complex vector shapes can be directly included between the `<svg></svg>` tags using a syntax analog to `html` and `xml`, and it can be seen either as alternative or complementary to the `<canvas>` element. Another opportunity of this element is to design user intefaces for particular kind of devices (such as touchscreens), as there is the possibility to make any element clickable or draggable;

webgl: although it is not an `html5` element, it can be seen as a side technology of it. It allows the rendering of interactive 3D graphics without the use of browser plug-ins via the `<canvas>` element (another proof of its potential), and it is another opportunity to manage in real time telescope observations and astronomical data in two or three dimensions to create a software-free, plugin-free, and web-based alternative to visualization applications such as ds9. Webgl is a low-level technology and requires good programming skills. However it can have a huge role in the simplification and modernization of the astronomer's work.

It is crucial to understand that the only thing requested to the **user** to be able to work with all these kind of technologies is nothing but an **updated browser**.

4.2 css3

In modern web design, it is essential to separate the content (written in `html`) from the layout (defined by the `css` stylesheet[21]). It is the same concept that researchers apply every day while using TEX or LATEX: the graphic presentation of their scientific content is defined by the stylesheet provided by A&A, ApJ, MNRAS, etc, which is normally described in a .sty file. An example of `css` stylesheet can be seen in Listing 2.

[21] http://www.w3schools.com/css/default.asp

The improvements coming with the version 3 of the `css` language, or `css3`, are still in development and they do not present peculiar features for our purposes, or at least they are not so crucial as the new `html5` features described in Sect. 4.1: at the end it is a technology which is merely related to the graphical presentation. However, there are good reasons to spend a few line about `css`. We explained in Sect. 1 that astronomers are, generally, experienced internet users, and, like any other web surfer, they expect a comfortable experience from a site that they have to use several hours per day.

The new modules of `css3` help to reach this result by introducing new methods for backgrounds, borders, effects and animations. In particular:

transforms: the 2D transform methods provided by `css3` can be helpful to rotate or flip an image on the screen to match the orientation of another image taken with a different telescope or instrument;

transitions: the new opacity methods allow to control the transparency of the elements, making easy the superposition of two images previously flipped or rotated with the transform method mentioned above, to check astrometry, orientation and any kind of visual comparison.

`css3` can also help to adapt the layout of a web application for different kind of devices. Moreover, similarly to `html5`, version 3 brings some new rule, expecially new selectors to access the several page elements, which are useful to replace script calls and keep the content cleaner and more readable.

4.3 javascript

In Sect. 4.1 and 4.2 we presented an overview of the latest technologies for the web content and its design, and we mentioned that their new features can replace some particular type of script call.

Normally in web pages these scripts are written in `javascript`, a language not to be confused with `java`, which is a software platform based on a virtual machine which runs programs written in its specific object-oriented *programming* language, also named `java`.

Astronomers working on linux or UNIX machines use everyday scripting languages such as `bash` or `python`, or the IRAF scripting language known as `cl`, to perform simple operations such as managing table columns, renaming files, or, typically, to call and interact with more complex programs responsible of the core of the operation.

The same way, `javascript` plays an essential role in web programming: it performs simple operations by manipulating the page elements (showing/hiding content, changing features); and most of all it works as a bridge between the site content and the complex programs that astronomers need to produce their results.

4.3.1 AJAX

The double use of `javascript` as both page manipulator and link with the server-side routines underlies the “philosophy” that modernized the web experience in the last few

years.

In an old conception of the internet site, still used every day in our field (Vizier, Simbad), a request of information such as the submission of a research form is sent to the server, which calculates a response, writes it in `html` and reloads the page to show the result to the user. This is pefectly useful in sites such as ADS which are targeted, basically, to provide two main things: a link to a paper and a BibTeX file. However it turns very annoying to have a page reload any time that an astronomer searches a set of objects, refines a result, browses it, selects a subset of those information, searches something on this subset and so on.

In the new conception of web site, the request is called via `javascript` which points to a hidden server page responsible to produce the output, then takes the response when it is ready and finally manipulates the content of the front page to show it. As the communication occurs in background and the extra content is dynamically added to the foreground page, there is no reload. This technique, called AJAX (Asynchronous JavaScript and XML[22]), can be seen in action in modern webmails, webchats, social media, and sites with highly dynamic content.

Modern webtools for astronomy should definitely use a technology of this kind to be perceived as real useful applications to interact with a database or a distant telescope.

4.3.2 jquery

Several `javascript` libraries for any kind of purpose are available through the internet: from generic development tools to complete sets of functions for very specific tasks. In the last years the `jquery` library[23] grew in popularity due to its simplified way to access the `html` elements (using a syntax analogue to that of the `css` selectors) and to manage the AJAX requests. The functions are chainable and cross-browser.

An example of `javascript` code written using the `jquery` syntax can be seen in Listing 3.

The diffusion of `jquery` in internet sites is currently very wide[24], and a quick research allows a developer to find on the internet the necessary resources to perform a particular task. We consider this library very suitable for astronomical web-based applications because it is enough generic to be adapted to our aims and enough simple to be used for the most common operations.

Another advantage of `jquery` is that it is easily expandable with custom plugins. We use this technology in the astronomical webtools with the aim that other astronomers might contribute with their own plugins to improve the list of possible tasks.

[22] Despite the name, the use of Javascript as a scripting language and XML (eXtensible Markup Language) as interchange format for the information is not mandatory. The technique simply consists in the background elaboration of the request and in the further manipulation of the foreground page when the response is ready.

[23] http://jquery.com/

[24] http://w3techs.com/technologies/details/js-jquery/all/all

```javascript
$(document).ready(function(){

    // First test
    $('form').submit(function(){
        var myemail = $('input').val();
        $('strong').fadeIn(900).text('You inserted: '+myemail);
        $('strong').addClass('okmail'); // styled in mystyle.css
        return false;
    });

    // Svg test
    $('svg>*').click(function(){ //each direct svg child element
            $(this).attr('style','fill:green');
    });

});
```

Listing 3: An example of `javascript` code using the `jquery` library. We suggest to copy this code in a simple text file named `myscript.js`. If this script is in the same directory of the `mypage.html` file (see Listing 1), it will be possible to see the effect of clicking on the Submit button and on the svg circle. Please note that the page is not reloaded and that there is no need to set up a webserver on the user machine to run this example. To automatically include the `jquery` library, it is sufficient to have an internet connection.

4.4 canvas versus svg for plots

Easy to understand and custom defined plots are crucial in scientific communication and obiviously in astronomy. The importance of effective communication is growing more and more, and several works[25] explain different techniques based on the communication medium (presentations, posters, papers).

Unlike communications in printed or even in pdf format, in the web context a user always expects to have a visual and interactive feedback on the graph, and we have seen that the `<canvas>` tag and the `<svg>` element can be in principle seen as two technologies to perform this task: displaying plots and vector graphics in web pages and provide user interactions.

We deeply investigated the capabilities of the `<canvas>` and the `<svg>` elements for our purposes, and we suggest when it is better to use one technology or the other.

The `<canvas>` element is rendered pixel by pixel, so it is *strongly dependent on the size of the screen*, but *weakly dependent on the number of canvas elements*. In canvas, once the graphic is drawn, it is forgotten by the browser. If the position of an element should be changed, the entire canvas needs to be redrawn. The resulting image can be easily saved as a .png or .jpg image. For these reasons, even if vector drawing is provided, it matches more the concept of raster, resolution dependent graphics (as for example astronomical images). Then we consider that it should be used for for

- interactive image editing (rotating, resizing, changing image cuts);
- generating raster graphics (from FITS or binary images);
- reading pixels to gather data for histograms, and superposing data (star names, circles, areas).
- anything related with raster graphics, pixel-based analysis or display;
- plots and charts providing a **huge** amount of elements.

The `<svg>` elements, being based on `xml`, can be considered as `html` tags. It is independent from the size of the rendered area, but it is *dependent on the number of objects*. (as basically the webpage weight increases). as turns slower if a huge number of elements is present, it should be used for:

- resolution-independent user interfaces.
- interactive or animated data charts and plots, even with a large amount of data, if a suitable technique to limit the number of elements is used[26].
- anything is related to vector graphics with visual data analysis interaction.

Frameworks aimed at implementing data visualization through these elements are extremely new and in rapid growth. For example

[25] http://www.treesmapsandtheorems.com/

[26] if a query on a database returns millions of astronomical objects, it is suitable to implement a visualization based on color or saturation instead of a scatterplot composed by millions of svg nodes.

```
var w = 350;
var h = 100;
var barPadding = 1;

var dataset = [ 5, 10, 13, 19, 21, 25, 22, 18, 15, 13,
                11, 12, 15, 20, 18, 17, 16, 18, 23 ];

var svg = d3.select("body")
    .append("svg")
    .attr("width", w)
    .attr("height", h);

svg.selectAll("rect") // they still do not exist
    .data(dataset)
    .enter()
    .append("rect") // they are created here
    .attr("x", function(d, i) {
        return i*(w/dataset.length);
    })
    .attr("y", function(d) {
        return h-(d*4);
    })
    .attr("width", w/dataset.length-barPadding)
    .attr("height", function(d) {
        return d*4;
    })
    .attr("fill", "teal");
```

Listing 4: an example of `javascript` code using the `d3.js` library and inspired by an internet tutorial (see the text). we suggest to copy this code in a simple text file named `myplot.js`. if this script is in the same directory of the `mypage.html` file (see listing 1), it will be possible to see a vectorial bar chart. as in the `jquery` example, to automatically include the `d3.js` library it is sufficient to have an internet connection.

flot: a `jquery` plugin which uses the `<canvas>` element to provide simple usage, attractive looks and interactive plots[27]. Several clones/forks providing more or less the same capabilities can also be found on the internet (`flotr`, `gflot...`).

d3.js: (Data-Driven Documents), not a `jquery` plugin although it has a similar syntax. It is thought to interact with any `html` element to draw a graph, but methods based on the `<svg>` elements are implemented to reach this goal[28]. (see Sect. 4.4.1).

For the considerations explained above, and even if we consider `flot` a valid alternative, we suggest `d3.js` over `flot` for implementing data charts and plots, and we give a brief overview in the next section.

4.4.1 d3.js and derivates

`d3.js` is a `javascript` library for manipulating documents based on data[29], using in particular `html` and `<svg>` elements. Its syntax is similar to the `jquery` syntax, and an example inspired by an internet tutorial[30] can be seen in Listings 4.

A didactic set of examples is provided by the author[31] and daily updated. The library provides a set of methods for plot and charts (pie chart, scatter plot, bar chart, area and line chart), and a set of cartographic projections (Equirectangular, Mollewide, Aitoff, Orthographic) which would be particularly useful to mark observational data on the celestial sphere.

An inconvenient of `d3.js` is that it is a quite low-level library, although it is deeply developed and daily updated. To circumvent this limitation, several projects, both free and open source or commercial, are growing in these days to provide an higher level abstraction:

datameer: is a commercial framework[32] providing a WYSIWYG editor to edit the graphs;

rickshaw: is an open source object oriented `d3.js` implementation [33];

nvd3: provides a collection of re-usable plots and chart components[34];

dc.js: focuses on the `d3.js` **crossfilter plugin** developed by the same author, and allows highly efficient exploration on large multi-dimensional dataset [35].

Given the youth of these project, it is impossible to point out the best solution to implement a web tool for astronomical data analysis.

[27] http://www.flotcharts.org/
[28] http://d3js.org/
[29] http://d3js.org/
[30] http://alignedleft.com/tutorials/d3/making-a-bar-chart/
[31] http://bl.ocks.org/mbostock
[32] http://www.datameer.com/product/data-visualization.html
[33] http://code.shutterstock.com/rickshaw/
[34] http://nvd3.com/
[35] http://nickqizhu.github.com/dc.js/

However, due to its flexibility (the main concept is to use the `html` and `<svg>` elements to emphdraw the graph on the web page), we forsee that the `d3.js` library will grow in polularity in the next months and years for this kind of purposes.

4.5 json

Several kind of techniques and formats can be used to exchange data and requests between the browser and a web server: `xml`, plain text, or `json` (JavaScript Object Notation). This latter is often used in AJAX exchanges, and recent browsers now either have native encoding/decoding.

With respect to `xml`, it is meant to reduce the complexity of the markup and provides native support for data types such as boolean, strings, numbers, arrays and objects. Moreover, it is easily manipulated in `javascript` as it is derived from it.

4.6 Managing fits files

As this document is related to the use of new web technologies in astronomy, it is important to spend some word about the available webtools to manage the standard format for astronomical images: `fits` files.

Those files are not basic image formats and unfortunately they cannot be displaied in websites with the usual syntax (`<img src="myfile.fits" />`).

In traditional websites, `fits` images are converted on-the-fly or in batch mode to `.png`, `.gif` or even `.jpg` format, and this solution is very useful to create static thumbnails. However, dynamic interaction with the displaied image (zooming, changing cuts, checking counts) turns a cumbersome task and may require continuous server-client interaction.

Our suggested approach, which is already implemented in several tools available online, takes advantage from the new `html5 <canvas>` tag. As we explained in Sect. 4.6, the `<canvas>` element can be used to display and interact with raster graphics. Here below we list what we consider the best projects concerning `fits` managers, readers and viewer:

jsfits is an example of `fits` library[36] useful to build canvas-based viewers[37] and is developed in the framework of the Las Cumbres Observatory Global Telescope Network[38];

astro.js provides an advanced `fits` library which also manage compressed `fits` files. The project also proposes being a space to develop and consolidate all astronomical `javascript` resources of the astronomical community.

Moreover, the astro.js project share the idea, presented in this document, that building a base set of `javascript` libraries would be extremely useful to perform quick preliminary analyses via browser.

[36] https://github.com/slowe/jsFITS
[37] http://lcogt.net/user/slowe/dev/fits/
[38] http://lcogt.net/

4.7 Summary

In this section, we explored the possibility to apply the top web technologies in modern webtools and rich internet applications for the astronomical research. The only request to the user (*researcher*) is to use an updated browser. These programs should take advantage of:

html5 to boost the contents;

css3 to improve the user browsing experience;

jquery to interact with the server-side via AJAX requests.

Moreover, concerning the content management, these programs should take advantage of the following libraries:

d3.js to build plots and charts in a `<svg>` element;

astro.js specific tools to manage `fits` files in a `<canvas>` element.

All the content of the section was focused on the user, and in particular on the user experience with the web interface, which must provide real time feedbacks to the requested tasks. In Sect. 5 we focus on the server-side, in particular for what concerns the management of the client-side requests coming from the user's browser.

We suggest to copy in separate files the code contained in Listings 1, 2, 3 and 4, following the instructions provided by the respective captions. It is possible to open the `html` page in a browser to see in action these technologies without setting up a web server on the local machine. We invite the reader to play and modify them following "inspiration" and searching examples on the internet.

5 Management layer

Linux-based systems are in the astronomical research the reference systems used to implement a server (for example for a department mail server, a data reduction cluster or a backup system), and for that reason we decided to base our concept on the so-called LAMP platform (Linux operating system, Apache http server, a database software as for example `MySQL` and a server-language as for example `php`). Anyway, the concept is enough generic to be ported to other operating systems without any particular effort.

This platform aims at managing the requests of the user arriving from the real-time layer and at interacting with a database to gather the information.

In the following sections we focus on the server-side and its technologies, and we call this module "management layer".

The management layer consists in the set up of a class of `php` methods to interact with the web page and a class of `php` methods that use the `mysqli` extension to communicate with the database based on `MySQL`.

In the following sections we give an overview of these two components.

5.1 php functions for the interaction with the web page

`php` (PHP: Hypertext Preprocessor) is an object-oriented programming language which is meant to act as a server-side scripting language[39]. It is largely used to develop server applications. It takes client side calls, elaborate information and produces `html` code which is sent back to the browser. It can be called directly within the `html` code or by using `javascript`.

The simplicity to perform AJAX calls using `jquery` to add changes in the `html` elements suggested us the idea to keep completely separated the "foreground" `html` code from a class of high-level `php` methods. That is to say, we decided that the `html` will not have any direct `php` call in the code. This choice can be seen as an extreme AJAX-based approach and it is justified by two reasons:

portability: by completely separating the server-side content, it is simpler to provide `html` templates that can be used to implement the same server functions in languages other than `php`, such as for example `asp`. In principle, even C++ routines could be directly called. The same way, the upgrade of this layer with new functions or new versions of the same function do not affect directy the real-time layer.

team development: in Sect. 4.1 we mentioned **modularity** as an advantage for web-based tools in the astronomical research. Here we extend this idea suggesting that the separation of the modules and most of all an object-oriented approach is very convenient for a multi-partner development in an optics of interaction and collaboration.

Concerning this last item, the modularity between the layers will benefit from the experience of people not directly related with the research field (for example an extremely skilled web designer), and the modularity within the management layer will help to extend the collaborations between the astronomic fields to provide the best solutions for a specific task.

For example, in the framework of a network of telescopes observing the same field but with different instruments, times or wavelenghts, it will be advantageous to have specific methods to treat all these different kind of data before sending the results to the real-time layer. In this context we aim at developing communication standards as mentioned in Sect. 2 about virtual observatories.

Finally, even if it can seem in contrast with the layer/module concept, this `php` methods can integrate all the functions which are too expensive for the client in terms of cpu or memory usage. An example is the calculation of a `css` style parameter of any element of a huge set of results. This choice is based on the priority of the user experience.

5.2 php/mysqli functions for the interaction with the database

We started from the idea that webtools in astronomy should profit from databases, not only for what concerns the mere storage of scientific data, but also, e.g. in a virtual

[39] http://www.php.net/

observatory, for the data production logging, their processing and exploitation, users management and so on. Besides the high-level `php` classes, it is then also necessary to implement a lower level class with methods aimed at the interaction with the database. Several languages and possibilities exist. For homogeneousness with the previous choices, we suggest to use `php` in conjuction with `mysqli` (see the PHP website). `mysqli` (MySQL Improved) is an object-oriented library used in `php` to communicate with the `MySQL` database server (see [40]).

The class will handle all the aspects related to the database interaction, and it is enough indipendent from the upper levels to provide an interface with any kind of scripting language (e.g. not only our suggested real-time level based on `html` and `jquery`).

6 Archive layer

The current layer is responsible not only to collect and archive datai but also to manage user accounts preferences. Moreover it can perform automatic or requested actions on the data.

6.1 mysql database

6.2 C++ routines, sextractor, astrometry...

[40] `php.net`