a3ab206056a0b6c68f0789337945512d
ebrary

11 <?php>

"Invisible" Code and the Mystique of Web Writing

Helen J. Burgess

PHP is a popular scripting language used by hundreds of custom Web applications, widely used software such as WordPress and MediaWiki, and popular Web sites such as Facebook and Digg. Though it was originally designed for personal home page production (hence the acronym), PHP is most often used in server-side scripting. A core part of the LAMP stack (Linux, Apache, MySQL, PHP), it is probably the language most often used for database-driven Web applications, thanks to its easy support for MySQL. PHP code is also popular because it can be mixed freely with HTML, via simple opening and closing `<? php ?>` *tags. Even novice users can cut and paste code into Web pages and perform complex database operations—or create security and privacy problems: PHP is involved in a huge percentage of the vulnerabilities tracked by the National Institute for Standards and Technology.*

In an age of writing and code, we are all mystics. In my capacity as a teacher, I am the imparter of mysteries, the encoder of information. My favorite moments have neatly bookended activity in my Web writing, design, and development classes. The first is the moment of revelation, the pulling aside of the curtain: I stand at the podium, enter a URL, and then choose "View Source" in the browser. Students gasp. It's very satisfying.

My second favorite moment occurs at the other end of the class. A student (not a Web designer; perhaps a human development major, or an English student, or a psychology intern) is working on her tags, trying to figure out what is happening on the page. We work together, heads not quite touching, poring over the symbols on the screen, looking for (and finding) the code that will enable the page to display. She utters a small sound, makes a change, and previews the page. And the curtain falls, the page appears—but with a difference. For now, we understand the trick.

In new media studies, we have (or should have) gone beyond simply

From A to : Keywords of Markup. Minneapolis, US: University of Minnesota Press, 2010. ProQuest ebrary. Web. 12 March 2017.
Copyright © 2010. University of Minnesota Press. All rights reserved.

a3ab206056a0b6c68f0789337945512d
ebrary
pointing to a Web page and commenting on its structure, development, and language. But the current trend in focusing on "visual literacy" tends to emphasize what's on the screen, rather than what lies beneath. Our understanding of code has gotten lost under the layers of GUI and WYSIWYG; graphic design teachers (trained primarily in the visual register) pale when students click on the <> button in Dreamweaver, and the mysteries of markup appear. At the same time, though, teaching plain vanilla markup is somewhat old-fashioned. Sure, there is the magic moment of revelation, but in an age of database-driven pages and invisible scripting languages (PHP, Perl, ASP.NET, the usual laundry list of server-side applications), it seems rather quaint to be teaching the `<p>` tag (or even worse, the deprecated `<font>` and `<table>` tags). Server-side scripting languages complicate markup by writing code for us; if we try to view the source, all we see are the traces left behind in the shape of inert HTML tags. The magic happens elsewhere. Markup, then, has come full circle: from the mystery of the Web page, to the revelatory moment of "View Source," to the invisible scripting we know is there but can't quite get to. Unable to scan the page and take an educated guess at what's going on under the hood, students face a key disadvantage as they try to understand how the Web works. This is especially true in the world of the social Web, which is dependent on database calls and the run-time restructuring of pages for its vitality. "View Source" is no longer sufficient as a mechanistic view of the way information appears on the screen.

In this essay I want to talk about the history of invisible code and its relationship to the mystique of writing. N. Katherine Hayles notes that code has a tendency to operate through "practices of concealing and revealing" (54), in which the programmer chooses which code to leave visible for the purposes of authoring and debugging. But the process of revelation and concealment in markup goes much further back than electronic texts—in fact, it is an essential part of the hidden history of print and writing itself. Long before the magical moment of "View Source," print and book producers were already using their own forms of hidden markup: the symbols written on texts that contained instructions or marked points for the purposes of textual reproduction. These printers' marks are the antecedents of today's markup schema: they are marking up manuscripts in the same way we mark up electronic texts. Therefore, I want to start out by looking at the origin of printers' marks in the era of the manuscript, with the understanding that these marks represent an ancestry of sorts for Web writing.

From A to : Keywords of Markup. Minneapolis, US: University of Minnesota Press, 2010. ProQuest ebrary. Web. 12 March 2017.
Copyright © 2010. University of Minnesota Press. All rights reserved.

a3ab206056a0b6c68f0789337945512d
ebrary

The history of the marks that constitute any text has long been a field of study. D. F. McKenzie, in his study of the history of bibliography, notes that the body of the text has long held primacy over the other material that accretes to it. He argues for a "sociology of the text," suggesting that bibliography, marginalized (sometimes quite literally) to a supporting role in the text, actually has its own complex social history (15). Jerome McGann, similarly, has made arguments for the crucial and constitutive role of "bibliographic codes" (57): ways in which the material construction of the book (using leading, typeface, layout, gutter, and printers' marks) fundamentally changes the nature and meaning of any given text. Each text thus generates its own history as it passes through multiple marked-up printers' editions.

Markup works similarly in the formulation of historical (electronic) texts. It has its own history (the versioning of SGML/HTML/XHTML), its own grammatical lineage (the development of some tags over others), its own narrative (the archaeological layers of comments attached to shared code), and even its own politics (language choices, browser compliance, and the choice to share code or retain its mystique as the writing of an invisible professional). Markup thus becomes a kind of ghostly writing dependent on context and history, rather than merely a means of formatting text.

It's my belief that we are going through yet another stage in the history of markup. The second part of my essay will look at the functioning of server-side scripting as another invisible hand in the process of electronic writing. Sometimes practices of hiding and revealing, as Hayles describes, are necessary for the benefit of the programmer and/or reader—she cites on the one hand the practice of collapsing/hiding code into object chunks for ease of use, and on the other hand the revealing of HTML and comments for the explication of the marked-up text (54). The HTML tags we can see in our browser's "View Source" window are akin to early printers' marks: they are not readily apparent, but they can be read if we know where to look, in the process of flipping back and forth between page and source code. But sometimes, the concealing and revealing is the result of the operation of the server itself. Because server-side commands execute before the HTML is written, they are hidden from the browser not by the programmer but by the server itself. PHP, currently one of the most popular server-side scripting languages, serves as a useful focal point for my discussion of hidden code: it is written but never viewed by the end user; it is in itself a kind of writer of code, passing instructions to server and

From A to : Keywords of Markup. Minneapolis, US: University of Minnesota Press, 2010. ProQuest ebrary. Web. 12 March 2017.
Copyright © 2010. University of Minnesota Press. All rights reserved.

a3ab206056a0b6c68f0789337945512d
ebrary
thence to a database; it is never seen but only experienced as the end result on the page. Thus, I believe, we are making another transition in the history of (electronic) texts: from visible markup to invisible code.

Working in the Copy Shop

The prehistory of electronic markup is intimately tied in with the history of copying. Texts and markup have been entwined since at least the monastic period of book production through the systematic processes developed to reliably copy religious (and later secular) texts. Writing, as Walter Ong notes, is a codifying of orality into a text that can be repeated, what Ong calls an "exactly repeatable visual statement" (124–25). Any such repetition will necessarily involve a whole invisible apparatus: the infrastructure necessary to carry out the reproduction of texts, including the hidden labor of copyists, who communicated to each other during the production of each text using an abbreviated and highly specialized series of codes. These codes were the language that ensured a new kind of faith—not in the mysteries of religious life but in the fidelity of the text.

The history of organized production line copying in the West goes back to the scriptorium, a kind of workshop for the copying of texts, in which copyists worked side by side with illustrators and illuminators for several hours every day, painstakingly reproducing religious texts—the texts of mysteries. However, the ramping up of the production and copying of books begins with the growth of medieval universities and the rising demand for scholarly books (Febvre and Martin 20). The growth of the secular market for books soon developed into a rationalized system of production: the pecia system, wherein copy texts, known as exemplars, were loaned out in parts (piecemeal, or "pecia") to be copied and then returned. The pecia system was most important in its capacity as a preserver of the fidelity of the text. The loaning of an exemplar meant the limiting of degrees of freedom from the original—a way of ensuring that errors did not propagate over multiple copies.

The pecia system was important in the way it rationalized and specialized the book production system into a kind of laboring human machine. Febvre and Martin note that the pecia system encouraged separation of skills and a division of labor, such that "it became more and more common for separate workshops to be set up, with copyists in one shop, rubricators perhaps in another, and illuminators in another" (26). Along with specialization comes the problem of communication: how does one preserve the

a3ab206056a0b6c68f0789337945512d
ebrary

From A to : Keywords of Markup. Minneapolis, US: University of Minnesota Press, 2010. ProQuest ebrary. Web. 12 March 2017.
Copyright © 2010. University of Minnesota Press. All rights reserved.

a3ab206056a0b6c68f0789337945512d
ebrary
fidelity of the text when it must pass through multiple hands for copying, illustrating, and binding? The pecia system achieved this by the use of pecia marks: marks added by the copyist to communicate where each pecia was to be placed. These codes, like the XHTML markup we use today, were specifically meant for the structuring and formatting of text, marking the beginning and ending of a section, for example. Books were broken into sections and handed out for copying. Because this piecemeal approach often resulted in the copied piece starting and/or finishing in the middle of the sentence, the end of a section was marked by the copyist with a pecia mark. For example, "p4" might mean "end of pecia 4." This operated similarly to the "quire signature," which marked the end of a quire (that is, a section of a book made from four sheets of parchment or vellum folded and then stitched, usually resulting in eight or sixteen pages) and ordered it for binding.

Most importantly, from our perspective, pecia marks were an expression of a new way of thinking about texts: as information to be structured and processed. Febvre and Martin note that the medieval copyist often placed instructions on the manuscript to tell the illuminator what to put in (a lion, a garden, a snake). As a placeholder, a guide letter was put in the space where the illuminator (who often did not read the text) was to illustrate the letter (26). But pecia marks were unique in terms of their function: they did not mark the insertion of content, but the beginnings of an understanding of logical structuring—that is, the assembly of the book as a number of pieces that must be stitched together in physical (not semantic) order. The pecia mark is a coding system for the reassembly of a text: a communication of repeatable formatting from one writer to another. This, I would argue, is the beginning of markup: a language specifically developed for the logical structuring and later formatting of a document.

Going to the Print Shop

The movement from the pecia workshops to the print shop is a complicated tale, even from a mechanical standpoint: the trial-and-failure of metal castings, poor paper manufacture, and ink viscosity. The history of the printing press begins with a secret. In a lawsuit in 1434, Johannes Gutenberg was accused of working with partners on "secret processes" (Febvre and Martin 51), including the possible manufacture of press pieces (probably die-cast molds). He was not the only person working in this area—many metalworkers were attempting to improve the wooden block

From A to : Keywords of Markup. Minneapolis, US: University of Minnesota Press, 2010. ProQuest ebrary. Web. 12 March 2017.
Copyright © 2010. University of Minnesota Press. All rights reserved.

a3ab206056a0b6c68f0789337945512d
ebrary

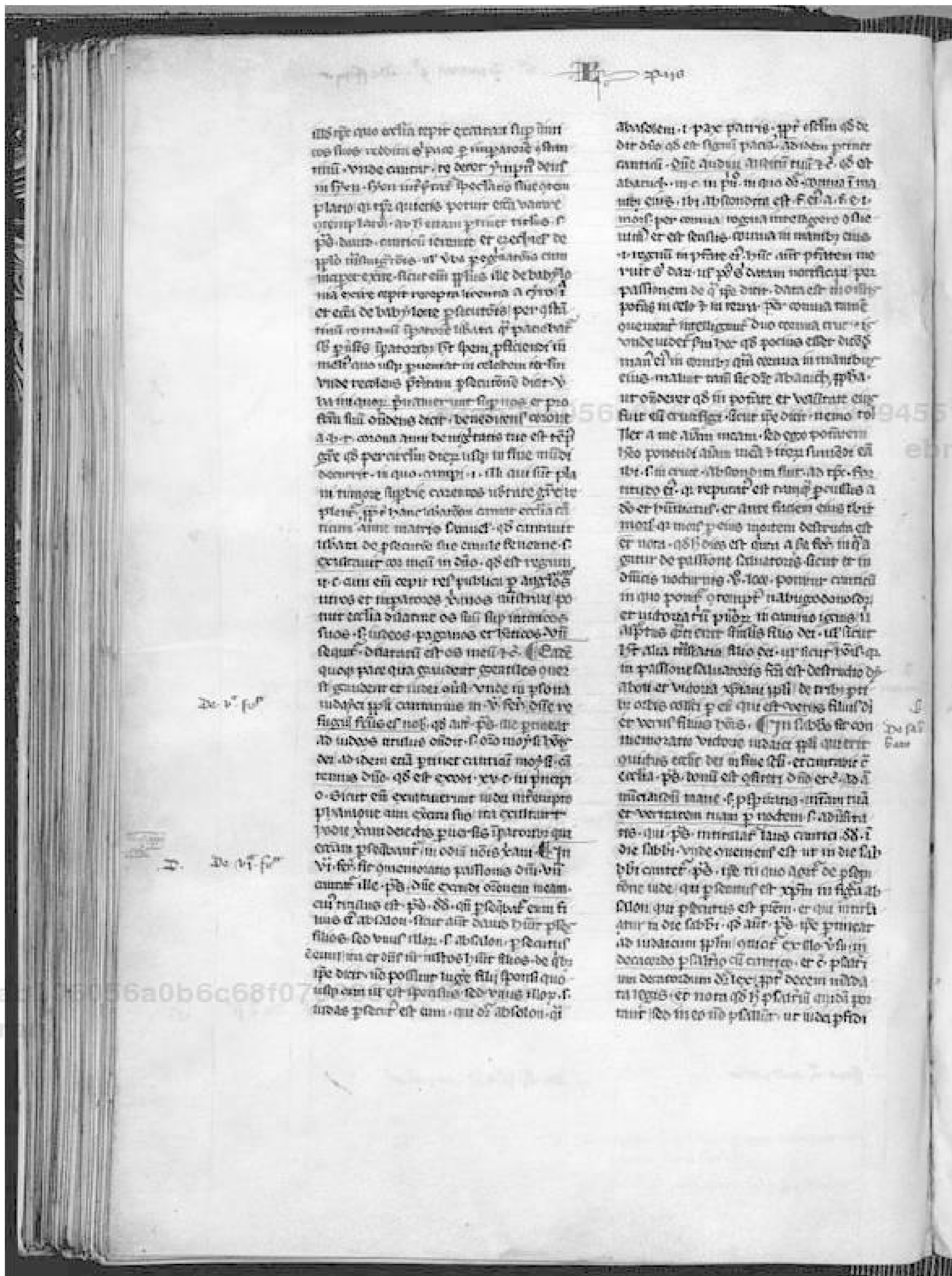

Figure 11.1. Manuscript showing pecia mark in outer margin of verso, "Finis xxx pet." From Guillelmus Durantis, Rationale Divinorum Officiorum. Italy (?), s. XIVin. call no. HM 26298. Reproduced by permission of The Huntington Library, San Marino, California.

a3ab206056a0b6c68f0789337945512d
ebrary

From A to : Keywords of Markup. Minneapolis, US: University of Minnesota Press, 2010. ProQuest ebrary. Web. 12 March 2017.
Copyright © 2010. University of Minnesota Press. All rights reserved.

a3ab206056a0b6c68f0789337945512d
ebrary
printing process by creating a system of interchangeable letters or moveable type. But Gutenberg is significant as a precursor of a dot-com millionaire: he was the holder of the technological "secret of the art of printing, . . . the making up of a page of print from separate, moveable types" (50). Already the value of the secret was evident: Gutenberg's angel investor, Fust, found him difficult to work with, so he waited until he had trained his more pliable assistant, Peter Schoeffer, and then called in the loan. Gutenberg's secret was thus revealed, and he died bankrupt (Febvre and Martin 55).

During this period, another series of codes was developed for marking up the text. The register—a table recording the first and last word on every page—was developed so that the printer could ensure pages were in order without having to read the entire text. Letters and numbers appeared on the bottom of leaves, not for pagination (as a service to readers) but as a way of ensuring the pages were folded and bound in the proper order.

If the pecia system enabled the reproduction of the text to a reasonable standard of fidelity (by using an exemplar), the printing press was able to capitalize on its mechanical ability to fix the page many times over from the same tray. Although errors could propagate this way (if the tray was laid out incorrectly, every single copy would also be incorrect), and in fact made for a corruption of the text that could be more damaging precisely because of its consistency (hence the introduction of printers' editions), the printing press allowed for a fundamental shift in the culture of writing through its emphasis on mechanical reproduction. The printers' marks on a printed copy were evidence not of the faith of the copyist in a religious sense—or even faith in the sense of accurate fidelity to an original manuscript—but rather faith in the ability of the machine to automate the material process of reproduction.

The Archaeology of Electronic Markup

The movement from religious to secular to mechanical faith has taken another turn with the implementation of document processing for online environments. Although computerized print software is essentially an analog of workshop procedures and relies to a great extent on precisely the same printers' conventions and vocabulary, HTML, descended from the more general schema of SGML, is designed as a way to facilitate the formatting of electronic information in a logical fashion explicitly for display online, in a browser. Like the regulated procedures of the scriptorium,

a3ab206056a0b6c68f0789337945512d

ebrary

From A to : Keywords of Markup. Minneapolis, US: University of Minnesota Press, 2010. ProQuest ebrary. Web. 12 March 2017.
Copyright © 2010. University of Minnesota Press. All rights reserved.

a3ab206056a0b6c68f0789337945512d
ebrary
HTML is designed with reproducibility in mind: the faithful rendering (in this case, electronic display) of the same file over and over. HTML tags are, in this case, similar to the guide letters and notations written into the manuscript: instructions to the browser for formatting the file on the screen. But at the same time, the file is not expected to be literally copied: no second file is produced. This is a move to a kind of virtual faith—a faith in the consistent display of a document that will nevertheless disappear when the window is closed.

Electronic markup is a fairly straightforward, mostly human-readable system in which each piece of content is tagged. For example, I am working up a simple Web page in HTML to display classroom grades. Here's what my code looks like:

a3ab206056a0b6c68f0789337945512d
ebrary

```
<table>
<tr> <td>James</td> <td>Jones</td> <td>B</td> </tr>
<tr> <td>Pavithra</td> <td>Jones</td> <td>B</td> </tr>
</table>
```

Tables such as the one above are an interesting case study in the way HTML has changed over time. Originally, HTML tags were designed wholly for logical data display: the heading `<h1>` is larger than `<h2>` to signify relative importance and nesting, not to give the letters a nice fat 16-point Times New Roman look. Tables, in their original formulation, were designed to be just that: a method of tabulating data accessibly. As higher bandwidth speeds and more efficient image compression algorithms enabled the routine use of images, semantic markup started to get tangled up with visual formatting. The graphic designer's desire for flashy pages and fixed-size layout complicated the split between content and logical structure by using structural elements (notably tables) to visually format pages. Ironically, this visual formatting relied on invisible markup and image elements. Rather than being used as a logical grid to organize data, HTML tables were used as a grid for laying out sliced-up images intended to produce pixel-perfect visual layouts. The one-pixel transparent GIF (an invisible square) was routinely used to force page layouts.

a3ab206056a0b6c68f0789337945512d
ebrary

Groups such as the World Wide Web Consortium (W3C), dedicated to the formalizing of Web standards, and a few designers (notably Jeffrey Zeldman and Eric Meyer), who wanted code to be more compliant with browsers and easier to reformat, were unimpressed by the hybridization of logical structure and visual design. Their dissatisfaction led to the standardizing of XHTML and CSS, a rigorous approach aimed at again separating

a3ab206056a0b6c68f0789337945512d
ebrary

From A to : Keywords of Markup. Minneapolis, US: University of Minnesota Press, 2010. ProQuest ebrary. Web. 12 March 2017.
Copyright © 2010. University of Minnesota Press. All rights reserved.

a3ab206056a0b6c68f0789337945512d
ebrary
form and content. Such a form/content division is logical and readable: "View Source" once again becomes a human HTML reader's best friend. We can read the markup and style sheet and make a fairly accurate guess as to what the document would look like on the page. And yet even at this stage, we already see the beginnings of a deferral of content: the link to an external style sheet means we must find and read that file to get a sense of the look and feel of the page. Nevertheless, the style sheet is readable by the end user; all the information is there. Reading and interpreting markup is thus relatively simple while we are talking about an electronic document in terms of one XHTML "text" to which is applied a structural markup and a presentational markup.

Three or four years ago, I would have been happy with my online grade listing. But mere display is really not all that useful. I've decided I want to put all my grades into a database and rerender the page in a scripting language, so that I can make changes to grades and add students without having to rewrite all that code. Virtual fidelity is no longer my goal: I want to change the document from reload to reload. In this instance, HTML is inadequate. I need a place I can store my grades, a way to update them, and a way to display them through my browser (computer scientists call this CRUD—create, read, update, delete). In short, I need a scripting language that can interact with a database.

PHP 101: What Is It? How Does It Work? And Why Should I Care?

PHP (in a typical piece of GNU-recursive jokery, this stands for "PHP hypertext preprocessor") is a server-side scripting language that, among other things, can be made to write code. In HTML, we are accustomed to writing the following:

```
<p>Pavithra Jones</p>
```

The browser reads these instructions, and behaves accordingly, printing to the browser

Pavithra Jones

In PHP, however, we have another kind of operation going on. Usually, you upload your HTML files to a server. Browsers all over the Web send a request for the file. The server gives it to them, and they read the HTML and display it accordingly.

From A to : Keywords of Markup. Minneapolis, US: University of Minnesota Press, 2010. ProQuest ebrary. Web. 12 March 2017.
Copyright © 2010. University of Minnesota Press. All rights reserved.

a3ab206056a0b6c68f0789337945512d
ebrary
PHP adds in an extra layer:

- The browser asks for the PHP file.
- The server looks for the file, executes the PHP, and gives the result to the browser as HTML.
- The browser interprets and displays the HTML.
- We see the words "Pavithra Jones."

Think of a restaurant: When we go to the restaurant and ask the waiter for an egg sandwich, he doesn't go back to the kitchen and dig up an egg and some bread. He goes back to the kitchen, asks the chef to cook the egg and toast, and then brings them back to us prepared. In this case, the Web server is the chef, cooking up the PHP code and sending it back as HTML.

The PHP code, in fact, looks like this:

a3ab206056a0b6c68f0789337945512d
ebrary

```
<?php echo "Pavithra Jones"; ?>
```

Again, we will see

```
Pavithra Jones
```

But if we look at the HTML source, the PHP will have magically disappeared, leaving plain vanilla HTML identical to our first example. The difference is that the server is reading and executing the PHP portions of the code, and writing "Pavithra Jones" when it has been told, right into that dynamically scripted Web page.

This simple echo command might be interesting in that it represents the movement from simple tagging to server interaction, but thus far it merely replicates the functionality of the HTML page: to display a hard-coded message. Why bother, when hard-coded HTML achieves the same purpose, and without having to compile PHP on your server? What gets interesting is the magic that happens when PHP interacts with a database language such as MySQL to retrieve information and display it on the page. I've decided I only want to display grades for students with the surname Jones. Let's take a sample snippet:

a3ab206056a0b6c68f0789337945512d
ebrary

```
<?php
$result = mysql_query("SELECT * FROM DTC355 WHERE sur-
name='Jones'");
?>
```

This is a hybrid piece of code. The **SELECT** part inside the quotes is a MySQL query: a piece of code asking the MySQL server to select all the

a3ab206056a0b6c68f0789337945512d
ebrary

From A to : Keywords of Markup. Minneapolis, US: University of Minnesota Press, 2010. ProQuest ebrary. Web. 12 March 2017.
Copyright © 2010. University of Minnesota Press. All rights reserved.

a3ab206056a0b6c68f0789337945512d
ebrary
records in the database table named "DTC355" with a "surname" value equal to the string "Jones." **mysql_query()** tells the server to execute the query inside the quotes. And the record data gets fed into a variable called **$result**. From there, it's a simple matter to get it to print on the page:

```
<?php
// Get records from the "DTC355" table
$result = mysql_query("SELECT * FROM DTC355 WHERE surname
='Jones'");
// keeps getting the next row from the database until there are
no more to get
while($row = mysql_fetch_array( $result )) {
// Print out the contents of each row and concatenate into lines
echo $row['name'] . $row['surname'] . $row['grade'];
}
?>
```

a3ab206056a0b6c68f0789337945512d
ebrary

All the information contained in **$result** gets fed into an array called **$row**. Then **$row** prints its values over until there are no more records in the array. This will produce the output:

JamesJonesBPavithraJonesBSammyJonesCChandraJonesA

Clearly we have all the information, but we want it to be marked up usefully. This is where PHP comes into its own as a metamarkup tool:

```
<?php
// Set up some repeating HTML tags to save some space later on
$startrow="<tr> <td>";
$cell="</td> <td>";
$endrow=</td> </tr>";
// start the table tag in HTML and put in a new line string feed
echo "<table>" . "\n";
// Get records from the "engl499" table
$result = mysql_query("SELECT * FROM engl499 WHERE sur-
name='Jones'");
// keeps getting the next row until there are no more to get
while($row = mysql_fetch_array( $result )) {
// Print out the contents of each row and concatenate with HTML
tags
// use the new line string feed to break up the HTML when viewing
source
```

a3ab206056a0b6c68f0789337945512d
ebrary
a3ab206056a0b6c68f0789337945512d
ebrary

From A to : Keywords of Markup. Minneapolis, US: University of Minnesota Press, 2010. ProQuest ebrary. Web. 12 March 2017.
Copyright © 2010. University of Minnesota Press. All rights reserved.

a3ab206056a0b6c68f0789337945512d
ebrary

```
echo $startrow . $row['name'] . $cell . $row['surname'] . $cell .
$row['grade'] . $endrow . "\n";
}
// close the table HTML tag
echo "</table>";
?>
```

The PHP script has now taken the information from the database, added in HTML for formatting, and output it. We've added in some invisible new line characters (\n) so that when we view the HTML, it won't all be in one long string. When we view the source, we see this:

```
<table>
<tr> <td>James</td> <td>Jones</td> <td>B</td> </tr>
<tr> <td>Pavithra</td> <td>Jones</td> <td>B</td> </tr>
<tr> <td>Chandra</td> <td>Jones</td> <td>A</td> </tr>
</table>
```

a3ab206056a0b6c68f0789337945512d
ebrary

Not pretty, but useful as an example: the PHP scripting has disappeared, to be replaced with a machine-written, static page.

This kind of query-based scripting, combined with PHP-generated markup, is interesting on several accounts. The first is that it is still displayed flat: all we see if we view the source from the browser is the HTML to display it. The second is that we can make as many calls as we want into the database: the page is suddenly as malleable as the database itself, a plastic space that can change with every impatient reload. The third is that it's writing within writing within writing: HTML holds the PHP, which holds the MySQL query.

a3ab206056a0b6c68f0789337945512d
ebrary

But the problem with PHP is that the end user can't read it. Now we are talking about a disappearing or invisible tagging system—one in which not only is the PHP operating invisibly on the page, but also the actual code of the document can no longer be read in the "View Source" command. Because PHP is executed on the server, not in the browser, any attempt to view the source locally reveals only the executed markup, not the PHP commands. We have moved to a tripartite markup system: content included (via script or database call) by PHP at run time, resulting information marked up logically in the XHTML, final look-and-feel text/image formatting in the CSS. This movement, from presentation to dynamic generation and retrieval, echoes the medieval transition from religious manuscript production to secular copy production: from the guidelines and sketches

a3ab206056a0b6c68f0789337945512d
ebrary

From A to : Keywords of Markup. Minneapolis, US: University of Minnesota Press, 2010. ProQuest ebrary. Web. 12 March 2017.
Copyright © 2010. University of Minnesota Press. All rights reserved.

a3ab206056a0b6c68f0789337945512d
ebrary
of the monastic scriptoria to the operational marks of the pecia workshop, and later the page registers of the print shop. An emphasis on the visual formatting of an existing long-form document gives way to instructions for the logical structuring of a document in pieces: the document object model. The shift from static formatting to run-time assembly also echoes the pecia and print shop's rationalization of production, in that pieces are farmed out to different agents—the style sheet, the looping algorithm, the database—just as pieces of a book were given to typesetter, binder, and cutter. To the end user, all this labor is invisible.

Ghosts in the Deep Web: The Database

In an electronic file, PHP scripts stand in for the information they will call; like pecia marks, they pinpoint the beginnings and endings for inclusions of text, whether quire, chapter, or blog entry. Like printer's marks, PHP snippets elicit an operation or execution: they are symbols requiring an action be performed. The key difference is that although pecia marks are meant to ensure repetition, PHP scripts are written to ensure flexibility; the goal of a dynamic Web site is to produce pages customized to each query. PHP does not contain the information itself (this is left to the database), but rather contains instructions on what to do with the data, and where to put it.

Database storage and retrieval (known as the deep Web) is obviously useful for corporate databases, shopping carts, online ordering, online catalogs, and so on. But it's interesting to us as Web writers because of the next-generation social Web. PHP scripting allows us to include a chunk of (often unformatted) content (notably, content presented by a content management system [CMS], blog entries, menu items) from a database, place it on the page, and then apply markup and styles. It allows for the switching of styles and templates according to whim, the easy addition of information via a text box and the reshuffling of that information or narrative at a moment's notice. A blog or a CMS can be changed with a click from a chronological sorting of entries, to one by category, to one by author.

The flexible chunking system of the deep Web means several things for a writer. First, content is broken into discrete sections and stored in a database, which has the effect of breaking the historical tradition of writing as a kind of complex narrative weaving. Lev Manovich has identified what he sees as a fundamental antagonism between narrative and database. Database logic, he argues, stores information in discrete pieces that are

From A to : Keywords of Markup. Minneapolis, US: University of Minnesota Press, 2010. ProQuest ebrary. Web. 12 March 2017.
Copyright © 2010. University of Minnesota Press. All rights reserved.

a3ab206056a0b6c68f0789337945512d
ebrary
only connected together at run time: the database "exists materially" (in the sense that all the available data are stored somewhere ready to be used) (231), while "the narrative is constructed by linking elements of this database in a particular order, that is by designing a trajectory leading from one element to another" and is thus "virtual" (231). Fundamentally, a database is about storage, while narrative is about ordering.

Narrative personal blogs, threaded discussion boards, and news services rely on just such a logic. On the surface, such sites appear to work chronologically (usually, although not always, backwards, with the newest entry at the top). Items might be linked in a kind of mininarrative. For example, Boingboing.net often posts a minichronology at the bottom of a post, pointing to recent posts on the same topic, which are also listed chronologically. But in fact, the flexibility of PHP to retrieve multiple records from a database and display them according to any number of requirements means that at any particular moment, the content might be reordered on the page by date, or category, or subject—or, in the case of multiauthored sites, the author. Under the hood, we don't have to move information from one category to another. We just change category flags in the database, or add a new keyword for multiple categories. Information retrieval can now be keyed to specific tags and sorted according to a different chronology, author, or numerical pattern. The SQL command "`ORDER by [asc, desc]`" renders this operation trivial. The server can take multiple database entries, recombine them into different groupings, and display them out of sequence, thereby having the effect of changing the fundamental nature of the page as progressive narrative. Narrative time (in terms of the chronology of the entry) is overturned in favor of recombinant time.

This reordering property fundamental to database logic is also spatial, as evidenced visually in the GUI to a MySQL database. One example, the Web-based phpMyAdmin panel, displays data spatially in text boxes and tables as a way of echoing its internal structure. This is only a visual representation, of course, but it resembles the spaces on an annotated manuscript page. The difference is that the spaces of the page can be changed at run time; clicking on a table heading or running a query will produce a new screen with a different combination of tables and text boxes. Unlike the old manuscript pecia marks, executed with repeatability in mind (the faithful reproduction of a text), PHP queries produce texts that can differ according to changeable input (and the data itself). Attempts to "browse" a database at the back end with a GUI such as phpMyAdmin lead to a misrecognition. We may think we are looking under the hood at the database,

From A to : Keywords of Markup. Minneapolis, US: University of Minnesota Press, 2010. ProQuest ebrary. Web. 12 March 2017.
Copyright © 2010. University of Minnesota Press. All rights reserved.

a3ab206056a0b6c68f07893379455 12d
ebrary

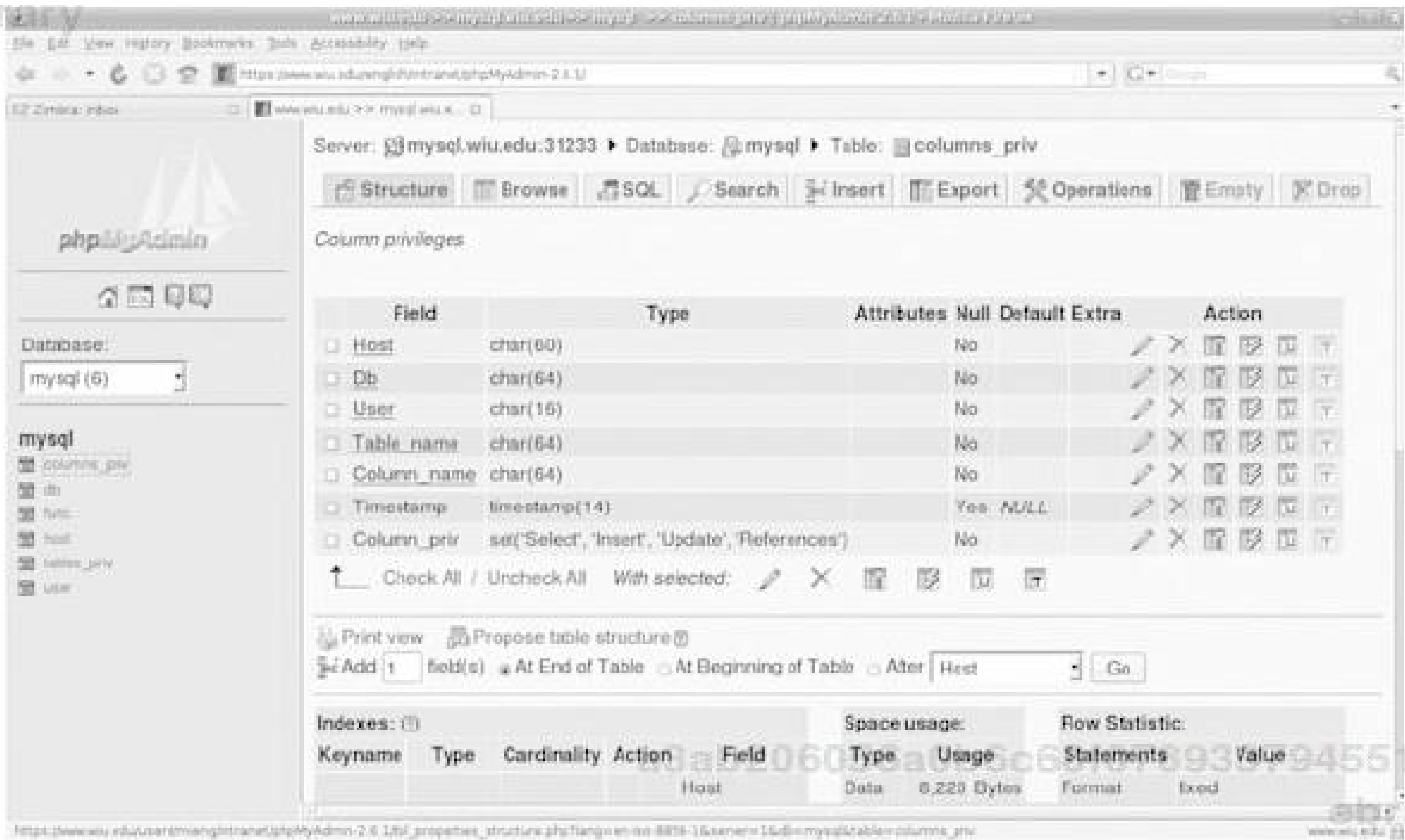

Figure 11.2. Tablelike layout of phpMyAdmin MySQL administration software.

but at any single moment, we are looking at a visual and tabular representation of a much more abstract field of data. The design of the interface itself determines what we see of the data structures and information.

At the same time, the phpMyAdmin interface offers us an interesting example of the meta-level similarity and difference between pecia mark and PHP query. Clicking on a link in phpMyAdmin displays the resulting data, but also helpfully displays the underlying query (the actual coded request) that was performed. On a manuscript, the pecia mark is already there, waiting to be executed. In the phpMyAdmin graphical interface, time is flipped: the query is performed, and then the code is displayed. This is because, ironically, the interface is itself scripted in PHP.

A second consequence of the PHP-enabled "chunking" of writing is that a site with a wide and changing range of dynamic content becomes one page; the database is the storage area for content. At its simplest, a site can now consist essentially of one XHTML page containing PHP scripts, one CSS page, and a database. Even CMSs, which seem hugely complicated in their file structure, adhere to the basic rule: the page you see is always the same index.php page. It calls various scripts and includes smaller formatted chunks (for example, a sidebar template or a footer) from other files, but the database is still responsible for all the content. At run time, the server executes the scripts in order, gathering first the "include" templates such as the sidebar, and then slotting in data from the database.

a3ab206056a0b6c68f07893379455 12d
ebrary

From A to : Keywords of Markup. Minneapolis, US: University of Minnesota Press, 2010. ProQuest ebrary. Web. 12 March 2017.
Copyright © 2010. University of Minnesota Press. All rights reserved.

a3ab206056a0b6c68f0789337945512d
ebrary
Within that simple structure, however, one must make a series of very careful choices. The retrieval of information is flexible, in the sense that databases change over time and can be called in multiple ways, yet at the same time rigid, in that the author has to decide in advance what choices can be made. This restriction affects both database design and PHP script design: what is likely to be most useful as a cluster of information? How should one piece of information be related to another in the database? My classroom example is simple enough—all I am likely to need to know is a link between name and grade. But complex databases can end up connecting one-to-many or many-to-many pieces of information. For example, my students might be in more than one class, or I might want to share the database with a colleague. I might suddenly discover that I have a student with a name that is too long for the field, or I might discover that one of my students has changed her name to Moon Unit, in which case I'll have to create a new record and somehow link it to the old one.

Holding all this structural information in my head long enough to determine what questions will be asked of it means conceptualizing the whole, even though I will only ever see it in parts. The database itself becomes, in some ways, a kind of material ghost that lurks at the back of every blog and CMS. As Manovich says, all the content (the paradigm) is there, but only a small amount will be syntagmatically displayed at any given page reload.

Finally, the intervention of PHP in the Web writing process means that agency is transferred from browser to server. If we think about it at all, it is the browser that is usually ascribed the agency in a Web session. The browser is, after all, responsible for sending and receiving coded messages; for interpreting the pecia marks embedded in the document. The browser is our faithful copyist, laboring within a series of markup rules to produce a text that is, depending on the browser type and settings, more or less the way the author imagined it.

With a PHP/SQL-driven site, however, the agency moves elsewhere. The secret technology is the server, which has now become more than a server: it does not just parcel out files, but also parses instructions inside them. The server is no longer a server but what Bruno Latour would call an actant—a semiautonomous personality or "quasi-object" (51) that is built in the nexus between different technologies of matter, machine, writing, and consciousness. Along with the end user author entering data and the originating author of the code, we have another hidden operator at work: the server, which faithfully executes the PHP code and writes the results to browser-readable HTML.

a3ab206056a0b6c68f0789337945512d
ebrary

From A to : Keywords of Markup. Minneapolis, US: University of Minnesota Press, 2010. ProQuest ebrary. Web. 12 March 2017.
Copyright © 2010. University of Minnesota Press. All rights reserved.

a3ab206056a0b6c68f0789337945512d
ebrary
In the age of the social Web—notably the advent of mass user-generated content sites such as YouTube—dynamic authoring has taken on a newly decentered role. Indeed, it is no longer clear who—if anyone—can claim to be the author of a site. Writers and video uploaders interface with databases and scripts in a network of relations, with agency shifting between writer, server, database, and browser. Manovich notes that the database is not merely the container upon which the algorithm operates:

> It may appear at first sight that data is passive and algorithms active—another example of the passive-active binary so loved by human cultures. A program reads in data, executes an algorithm, and writes out new data. . . . However, the passive/active distinction is not quite accurate because data does not just exist—it has to be generated. (224)

a3ab206056a0b6c68f0789337945512d
ebrary

For the casual user, the emphasis on content often serves to obscure the huge investment that has gone on behind the scenes to produce the infrastructure allowing all these mash-ups and content-sharing sites to exist. Blogging software, wiki packages, and CMSs are painstakingly scripted and debugged by large communities; the customizable Ajax interfaces are the product of intensive start-up development. Open source development means code is more accessible, but much of this code is aimed at producing seamless front ends for data input. At the click of "Upload" or "Refresh" or "Save Category," a kind of discretized and recombinant magic happens, with the server secretly working behind the scenes to produce the prestige. This last is the most interesting because it leads us right back to the beginning of writing: writing as mystique, the domain of the invisible expert who executes the marked-up page.

a3ab206056a0b6c68f0789337945512d
ebrary

Conclusion

Server-side scripting of database calls and markup are bringing into obsolescence the end-user "View Source" command. To be able to read HTML is no longer sufficient—one must be able to read the logic of the page and be able to tell what is going on underneath without being able to read it directly. Similarly, back-engineering a dynamically driven site is becoming a necessary skill for Web authors as they participate in the community of Web development. But marking up is not the "translation of the world into a problem of coding," as Donna Haraway would say (164). It is the act of making visible the invisible.

With this injunction, we have two options as readers and reproducers

a3ab206056a0b6c68f0789337945512d

ebrary

From A to : Keywords of Markup. Minneapolis, US: University of Minnesota Press, 2010. ProQuest ebrary. Web. 12 March 2017.
Copyright © 2010. University of Minnesota Press. All rights reserved.

a3ab206056a0b6c68f0789337945512d
ebrary
of electronic texts. The first, if we are unlucky, is to have to work through inference and perform the indirect act of reading the screen: looking at logical divisions of content in the page and working out the probable database calls. This requires a careful training in not merely visual (in the sense of images) but spatial and quantitative literacy—an understanding of how databases are organized, how they can be queried, and where and how the data can be best displayed on the page. In addition to the database design, the architecture of the site itself can often require a complex act of inference. Attempting to read a site can result in a series of structural deferrals as we look for a piece of code. In a CMS such as Joomla, for example, a desire to change the visible front page index.php will lead us several levels down the directory structure to another index.php inside the "Themes" folder, hacks of .htaccess files redirect pages while hiding the actual directory structure in the URL; server-side includes import of many snippets of HTML and PHP from other parts of the site. Thus, in addition to database logic, we must learn to read site design logic.

Our second option, if we are lucky, is to learn to read the code itself. In this endeavor, we have a notable helper: the comment. Comments are the pecia marks, registers, and guide letters of our time—in many ways, the literary exegesis of code. As creators of digital texts, we learn early on (if we are taught well) that commenting code well is absolutely essential for the transmission of an electronic text from one person to another, whether they be a fellow student, a fellow designer, or a client. The comment is in itself a literary form. The pinnacle in the genre of commenting is, ironically enough, a print text: *Lions' Commentary on Unix* (often called the *Lions' Book*). This book is a listing of the entire source code of the Unix 6 kernel, with an accompanying commentary. It bears a distinct resemblance to medieval text commentaries.

The comment is particularly vital today in the modern equivalent of the scriptoria: the large, virtual workshops of the open source software development community. Open source tasks in such communities as SourceForge are parceled out in pieces for coding by individuals; small software tools and plug-ins are developed by multiple authors using carefully arranged CMSs (such as Subversion and Bazaar) for versioning and communication. Chief among these communication tools are the comments embedded within the code itself: section markers and small reminders from one developer to another, explaining what a particular piece of code does or what interactions it might potentially have with other pieces of software.

a3ab206056a0b6c68f0789337945512d

ebrary

From A to : Keywords of Markup. Minneapolis, US: University of Minnesota Press, 2010. ProQuest ebrary. Web. 12 March 2017.
Copyright © 2010. University of Minnesota Press. All rights reserved.

a3ab206056a0b6c68f0789337945512d
ebrary
Comments are both like and unlike the original pecia marks they resemble. Structurally, they do indeed mark out the beginnings and endings of sections, and they mark out spaces for the inclusion of data. But just as PHP has changed the way we view markup—by becoming an active author in the process of generating a site from a database—so too comments go beyond the logical markers of the pecia system. Instead, comments seek to explain what will happen when the code executes; they look into the future and tell us what the invisible machine will do with all that data. Comments are thus a crucial companion to executable code, just as HTML was a companion to simple text and pecia marks were to manuscript fragments.

As I have shown, the social history of texts—a history of faith, from religious piety, to secular fidelity, to mechanical reproduction, to electronic display—that began in the medieval workshop has undergone a clear transition again in the age of the database and server-side script. But through all of these transitions, we have seen the consistent use of textual marks: guideline, pecia, tag, script, comment. In an electronic culture where so much press is devoted to front-end social tagging, user content development, and online jabber, we forget that there is always another kind of communication going on, an invisible social code: the communication between one developer and another, one server and another, one database and another. These marks may be hidden, but they are the underpinning of electronic writing.

Works Cited

Febvre, Lucien, and Henri-Jean Martin. *The Coming of the Book: The Impact of Printing, 1450–1800*. Translated by David Gerard. 1976. London: Verso, 1997.

Haraway, Donna. *Simians, Cyborgs, and Women: The Reinvention of Nature*. New York: Routledge, 1991.

Hayles, N. Katherine. *My Mother Was a Computer: Digital Subjects and Literary Texts*. Chicago: Chicago University Press, 2005.

Latour, Bruno. *We Have Never Been Modern*. Translated by Catherine Porter. Cambridge, Mass.: Harvard University Press, 1993.

Lions, John. *Lions' Commentary on UNIX 6th Edition with Source Code*. 1976. San Jose, Calif.: Peer-to-Peer Communications, 1996.

Manovich, Lev. *The Language of New Media*. Cambridge, Mass.: MIT Press, 2001.

McGann, Jerome. *The Textual Condition*. Princeton, N.J.: Princeton University Press, 1991.

McKenzie, D. F. *Bibliography and the Sociology of Texts*. 1986. Cambridge: Cambridge University Press, 1999.

Ong, Walter J. *Orality and Literacy: The Technologizing of the Word*. 1982. London: Routledge, 2002.

From A to : Keywords of Markup. Minneapolis, US: University of Minnesota Press, 2010. ProQuest ebrary. Web. 12 March 2017.
Copyright © 2010. University of Minnesota Press. All rights reserved.