Please answer the following questions about the section “Navigation Quick Keys for HTML”

1. What are core tasks for this domain?
The domain is navigation of HTML webpage elements. The Quick Keys are navigation only. Tasks are more comprehensive than just commands. Tasks contain hierarchical goals and end with specific actions. For a command language, the actions are the actual commands. To understand the core tasks, you need to think about how you read/explore a web page. You usually start with an overall glance around the page, an “overview”: looking at the top headings, the links and any text and/or images that are large and prominent. “Looking” usually means a combination of reading and visual search or even text search “find”. These actions verify that this is indeed the page you expected and are interested in. After you confirm that this is a page of interest, you will usually go in depth to a particular section and read the text and look at pictures, or activate the elements of the page (links, forms objects, etc.)

Does Jaws map well to the functionality of the task domain? Explain.
I believe that blind users will approach these tasks in much the same way as sighted users. Unfortunately, JAWS is not based on the types of tasks described above. This compromises the usability of this command language. For blind users, there are extra tasks that are needed such as stopping and restarting the reading of an element or possibly an undo to return to the previous webpage object examined.

The Quick Keys navigate to HTML objects of a web page. If you don’t know these objects, it will be difficult to use the commands. For example, “A” is the command to move to “Anchor”. What is an anchor to a person unfamiliar with HTML? (From Wikipedia: “HTML uses the <a> (anchor) tag to create a link to another document. An anchor can point to any resource on the Web: an HTML page, an image, a sound file, a movie, etc.”) To the user, an anchor is just another link. And would anyone care about navigating to a “Block Quote” in and of itself? I can’t imagine that blind users (or even sighted users like me) choosing to look at a webpage with the full array of HTML objects. Many Web pages do not contain all these objects. What is important are the following objects: Headings (by level), Text, Links. These form the basis of being able to glance over a page and then explore in depth. Note: the Quick Keys do not contain the important overview or “glance” commands List Links (INSERT+F7) and List Headings (INSERT+F6). These are both mentioned in Reading #1 as the first, and most common commands used by blind users when exploring a web page. Why not make these quick keys if they are used very frequently? My sense is that the JAWS language does not approach the support of a “glance” task. That kind of command would read multiple HTML types in a particular manner: Headings at the top level first, then major text and graphics, then detailed text such as paragraph or non-link text. One would hopefully want to explore a page as a hierarchical structure. That is the way webpages are usually structured.
Without a total redesign of JAWS, and limiting ourselves to the existing language, you need to figure out what are necessary, useful and frequently performed functions and then make them Quick Keys since you want to speed up performance. These Quick Keys could even be composed functions (more than one basic command). We would need to observe blind users and find out what those commands might be. My first guess is that List Links and List Headings should be quick keys. Then Next Link (TAB) Open Link (ENTER), Next Heading (H) seem frequently used along with the heading level numbers. The “Next (same or different) element” commands and “Prior Element” commands seem very useful. The “Place Marker” command seems very useful as well as the “Move to PlaceMarker” command that is not included in the Quick Keys! The JAWS Find command seems also useful. The Forms commands (button, check box, edit field) seem much less used and should be handled separately. The Table commands are probably more useful than the Forms commands but less used than basic web page navigation.

2. Is the language consistent in lexicon and syntax (Note, this is just the Quick Keys)?

LEXICON
The lexicon is single letters (except for Y and W), the numbers 1-6, PERIOD, COMMA, SHIFT, CTRL, TAB and “Windows key”.
Each command has a description or command “name” that consists of from one to three words. Most command names are HTML objects, but a few describe actions. There are 28 commands that are semantically distinct forward navigation commands. These can be combined with the SHIFT to move backwards, giving 56 distinct navigation commands for elements in an HTML page.

Consistent:
Abbreviation Rule 1: Take first letter of the command name. Example is “A” for “Anchor”. (Applies to 19 elements of the Quick Keys table. There are 10 elements that do not follow this rule.)
Abbreviation Rule 2: Take last consonant before the end of the last word of the command name. Examples are “K” for “Place Marker” and “M” for “Frame” and “X” for “Check Box”. (Applies to 3 elements of the quick keys table. There are 7 elements that do not follow either this rule or rule 1.)

Inconsistent:
“Q” for “Block Quote”;
“Z” for “Division”;
“SHIFT+PERIOD” for “Step Past element”.
“CTRL+J” for “Jump to Cell within table”
“1 through 6” for “Heading at Level”

SYNTAX
Consistent:
Single Quick key to more forward; SHIFT plus Quick Key moves backwards.
Inconsistent:
“SHIFT+PERIOD” for “Step Past element”.
“SHIFT+COMMA” for “Step Before Element”. This conflicts with “SHIFT+PERIOD” for “Step Past element”.

3. **Is this a simple command set, commands plus arguments/options, or hierarchical?**
It seems without deeper analysis that this is a simple command set, but it is a command plus option language with this syntax: option plus command where the option can be “null” or SHIFT key to move backwards. (There are also a few examples of CTRL.) In other words, this is about the simplest command plus option language you could have! (I note that a counter argument is that the SHIFT key merely capitalizes the letter key, so you really have only a simple language consisting of lower and upper case letters. I don’t think that is quite the case here and have opted for a command plus option organization.)

The use of a simple command set means that in a complex domain, there are many, many commands that must be created. For the Quick Keys there are 28. The power of using command plus argument is demonstrated in the use of “SHIFT” plus a command to move backwards that element. This essentially doubles the number of possible commands to 56. There is another structural feature of this language which is quite unusual and that is the use of a mode for the heading “H” command. Choosing one of the numbers 1 to 6 will set the “level” of the heading that will be navigated to in the “H” command.

4. **How easy is this language to recall without the sheet?**
Probably it is somewhat difficult since Quick Keys has 28 commands. Although the command names are meant to be mnemonic, you have to guess the right command name. Most are HTML objects, but some are actions such as “J” for jump to line. Most commands typically use Rule 1 for abbreviation. Many of the command names might be hard to remember because they are very general and not discriminating. For example, “Non-link text” or “Same type element”. On the other hand, some command names are infrequent and discriminating. For example, “anchor” and “combo box” should be easy to recall. Since some commands use multiple words, it is easy to be unsure about applying rule 2 above, which itself is a very strange rule and not recommended as an abbreviation strategy by the textbook. There are 7 out of the 28 commands that do not follow either rule 1 or 2 and must be uniquely memorized. Since rule 2 is almost impossible to know when to apply, one could say that 10 out of the 28 commands do not follow the abbreviation strategy. That means that 10/28 or 35% are exceptions: you must guess 1/3 of the commands. Finally, the consistent use of “SHIFT” when applied to a command doubles the number of possible commands with an easily remembered key.

5. **Predict common errors?**
1. Hit the wrong key using a key that is close to it: CONTROL instead of SHIFT or any of the single letter keys since almost are used except for W and Y.
2. Not pressing all the necessary control keys simultaneously: “Return to previous cell” is “CTRL SHIFT J”
3. Missing the control keys (SHIFT/CTRL/ALT/TAB etc.) tactiley since they are located on the perimeter of the keyboard and there are no physical landmarks.
4. Apply abbreviation rule #1 to one of the exceptions. “F” for “Frame”, “B” for “Block Quote”

5. Possible command name with same letter: “L” for “Link” or “Line” rather than “List”, “D” for “Division”, “J” for “Jump to cell”, “T” for “Non-link text”

6. Semantic ambiguity: “Jump to line” and “Jump to cell”

7. Mode errors when using the heading commands “H” and “1” to “6”. How do you inform the blind user of what the current level is?

6. How fast to perform?
Very fast. This is the advantage of JAWS in general and Quick Keys in particular. It takes 0.2 seconds to type a key not including the mental time to remember the command. (That takes, on average, 1.3 seconds, if you remember it right away.) Note that “perform” does not include learning time, and usually refers to practiced expert users.

7. How would you make this language more usable?
I would first do a thorough study of what the core functions should be. Observing blind people as was done in Reading #1 could do this. How do users use the screen reader in a real task environment? These commands have been chosen based on HTML objects. That is important in some ways, but many of the complexities of JAWS have been introduced by enumerating all possible elements in HTML. Like UNIX, there is a subset of these that are core. The developer needs to find out what those are and then optimize those commands so they are easy to learn and remember. A hierarchical structure might be better than the simple command structure that is used by JAWS.

Another possibility would be to design a special keyboard for blind users that really provided frequently performed commands in a spatial cluster that is fast to type. The control (modifier) keys should be located relative to the regular QWERTY letter keys. There should be tactile landmarks that can be felt to orient the person’s fingers to the keyboard.

Finally, I would provide learning materials that are task oriented rather than lists of alphabetically ordered commands. And those learning materials (or even the lists of commands already provided should be in Braille.)