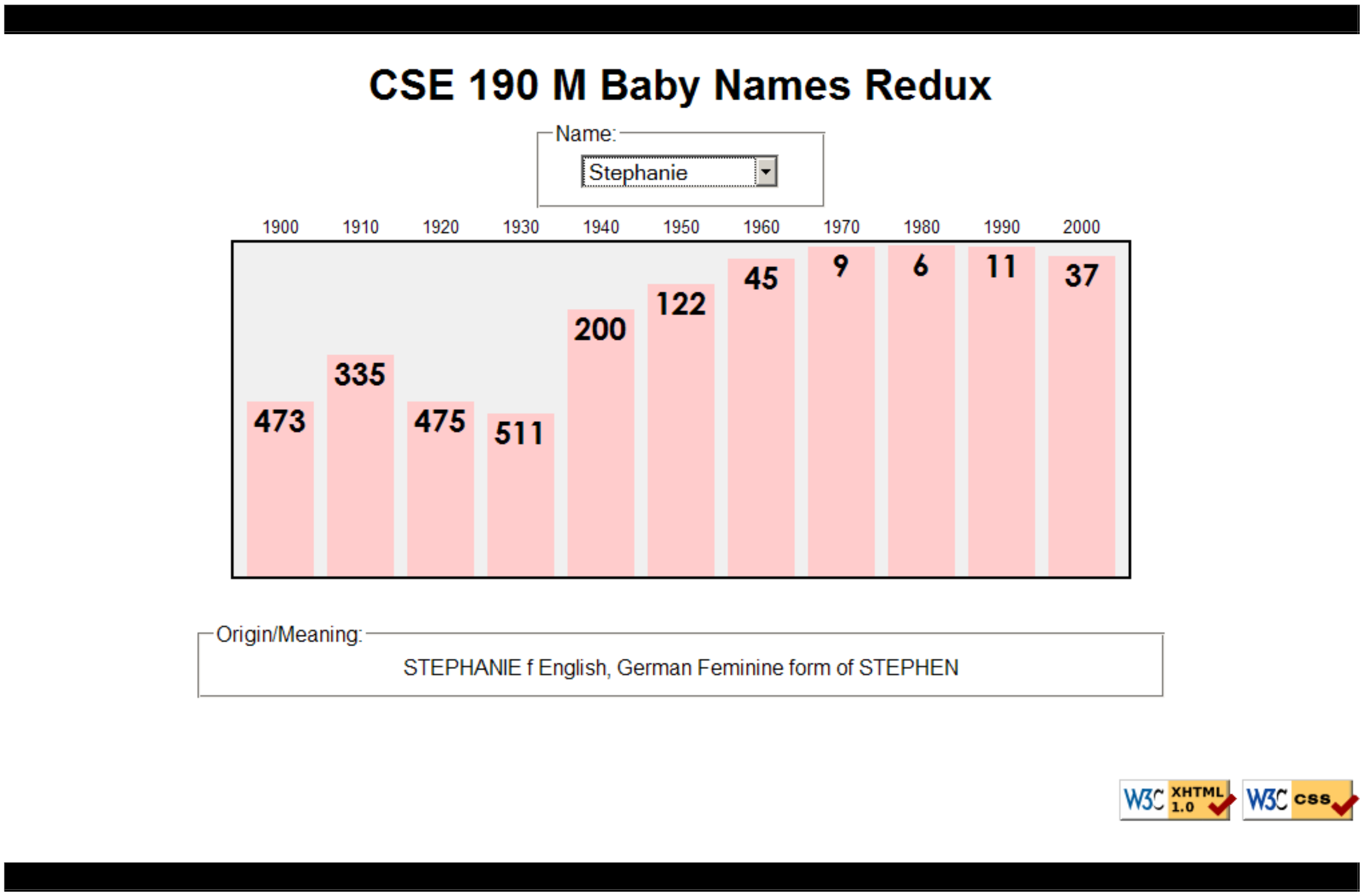
University of Washington, CSE 190 M, Spring 2007
Homework Assignment 5: Baby Names Redux

Part A (no graph) due Tuesday, May 1, 2007, 11:59pm electronically
Part B (complete) due Tuesday, May 8, 2007, 11:59pm electronically

Special thanks to Nick Parlante for the idea of this nifty assignment!
also see: http://www.peggyorenstein.com/articles/2003_baby_names.html

This assignment tests your understanding of fetching data from files and web applications using Ajax (Asynchronous Javascript and XML). You must match in appearance and behavior the following web page:



Every 10 years, the Social Security Administration provides data about the 1000 most popular boy and girl names for children born in the US, at http://www.ssa.gov/OACT/babynames/. Your task for this assignment is to write the Javascript code for a web page to display the baby names, popularity rankings, and meanings.

You do not need to submit any `.html` or `.css` file. Instead, we will provide you with the HTML (`names.html`) and CSS (`names_style.css`) code to use. Turn in the following file:

- `names_script.js`, the Javascript code for your web page

1 point of your grade will come from successfully posting your program solution to your UW web space. To receive this point, we must be able to reach your password-protected page at *exactly* the following URL:

```
http://students.washington.edu/your_UWnetID/cse190m/homework/5/names.html
```

You'll need to upload `names.html` and `names_style.css` to the web space with your Javascript file.

Data:

Your program will read its data from a web application located at the following URL:

`http://students.washington.edu/mdoocy/babynames.php`

This web application accepts three different types of queries, specified using a query string with a parameter named `type`. Each type of query returns text or XML as its output. If you submit an invalid query, such as one missing a necessary parameter, your request will return a status of 400 rather than the default 200. If you submit a query for an unknown name, your request will return a status of 404.

The first type of query is `list`, which returns text of all baby names on file, with each on its own line. The following query would return the results below (shortened by ...):

`http://students.washington.edu/mdoocy/babynames.php`**`?type=list`**

```
Aaliyah
Aaron
Abagail
...
```

The second type of query is `rank`, which returns an XML document about that baby name's popularity rank in each decade. The `rank` query requires a second parameter named `name`.

The overall XML document tag is called `<baby>` that contains an attribute named `name` that represents the baby's first name. Inside each `<baby>` tag are many `<rank>` tags, one for each decade of survey data. Each `<rank>` tag contains an attribute named `year` that represents the relevant year of data. The text inside the `<rank>` tag is that baby name's popularity in that decade.

The data has 11 rankings per name, one per decade from 1900 to 2000, from 1 (popular) to 999 (unpopular). A rank of 0 means the name was not in the top 1000. The following query would return the results below:

`http://students.washington.edu/mdoocy/babynames.php`**`?type=rank&name=morgan`**

```
<baby name="Morgan">
    <rank year="1900">430</rank>
    <rank year="1910">477</rank>
    <rank year="1920">526</rank>
    ...
    <rank year="2000">25</rank>
</baby>
```

The third type of query is `meaning`, which returns text about that baby name's origin and meaning. The `meaning` query also requires a second parameter named `name`. For example, the following query would return the results below:

`http://students.washington.edu/mdoocy/babynames.php`**`?type=meaning&name=martin`**

```
MARTIN m English, French, German, Scandinavian, Russian, Romanian, Czech,
Slovak, Slovene, Hungarian, Bulgarian From the Roman name Martinus, which
was derived from Martis, the genitive case of the name of Roman god MARS.
```

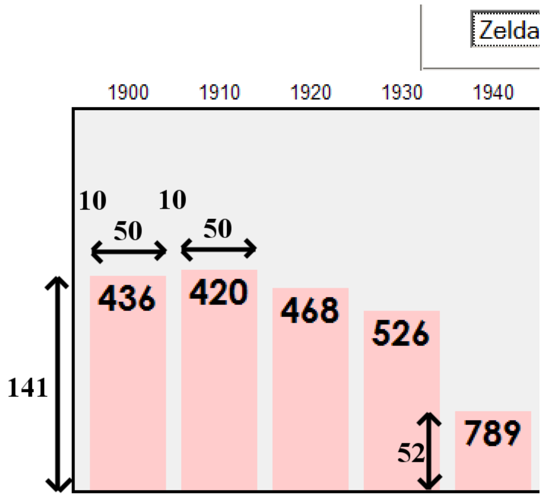
All names returned by the `list` query have ranking data, but not all will have meaning data. In such a case, do not display any text in the Origin/meaning area.

You may assume that all XML and text data sent to your program is valid and does not contain any errors. You may also assume that the web app is reachable at the time your code runs.

Appearance and Behavior:

The HTML page given to you shows a main heading followed by a selection box. When the page first loads, the box is empty and disabled. But in the background the page should load the name data as described later in this document, and once this has finished, the selection box should fill itself with all the names and should enable itself. Inject the names into the selection box with CSS ID `babyselect`.

There is also a large shaded bar graph section in the center of the page that is initially blank. The shaded section is 670 pixels wide and 250 pixels tall. If a name is selected from the selection box, this shaded section is filled with bars representing that name's popularity for each year in the data. Any data from a previous name should be cleared from the graph.



Each ranking bar has a width of 50px; its height is one fourth as many px as the "inverse ranking" for that decade. The inverse ranking is 1000 minus the ranking. For example, the ranking of 880 leads to a bar with a height of 30 (one fourth of 120, which is 1000 - 880). The bars are positioned absolutely with respect to the overall shaded graph section. Their x-coordinates are such that the first bar's left edge is 10px from the left edge of the graph section; there are 10px horizontal space between adjacent bars. For example, the first bar's left corner is at x=10, the second is at x=70, the third at x=130, and so on. All ranking bars' y positions are such that their bottoms exactly touch the bottom of the shaded graph area.

Within each ranking bar appears the ranking number for that name in that decade. The numbers appear in an 18pt sans-serif font, top-aligned and horizontally centered within the bars. Use the `ranking` style in the provided CSS file to achieve this appearance. Some less popular rankings (around 900 and up) have numbers that drop below the graph region's black border; this is expected behavior.

Above each ranking bar at the top of the shaded region are labels representing the corresponding years. These are positioned vertically 1.5em from the top of the shaded region and centered horizontally within the ranking bar's width. Use the `year` CSS style in the provided CSS file to achieve this appearance.

Underneath the ranking bars appears a paragraph of text explaining the meaning of the name shown. When a new name is chosen, that name's meaning information is loaded asynchronously and appears underneath the graph. Inject the relevant text into the paragraph with CSS ID `meaning`.

All other style elements on the page are subject to the preference of the web browser. The screenshots in this document were taken on Windows XP in Firefox 2.0, which may differ from your system.

Parts A and B:

You will turn in this project in two phases: A partial implementation before the midterm exam, and a complete implementation after the midterm is over. The complete implementation contains all features described previously. **The partial implementation does not need to show the bar graph of the name ranking data.** It does, however, need to show the list of names in the selection box, and when a name is selected, the name's meaning must appear in the origin/meaning section below.

Extra Features / Creativity Aspect:

To encourage you to be creative on this assignment, we will award you **+1 free late day** (up to a maximum of +2) for each of the following possible additions to your Part B page. You are not required to do any of these.

- **better loading feedback**: The page as currently described does not give the user very good feedback while it is loading its data, both initially (loading the list of names) and subsequently (when a name is chosen and that name's XML data must be fetched). Consider adding code to display a "Loading" message, loading styles on the page (such as "graying out" items or putting placeholder text into them), or even an hourglass mouse cursor (achieved by using the CSS `cursor` property).
- **ability to compare multiple names**: By default the page displays data about one name at a time. Consider adding code so that if a second name is chosen, the previous data remains, with the new data superimposed in a different color. This provides a way to compare the relative popularity of the two names over time. You should add some sort of "Clear" button so that the user can erase previously shown names.
- **Other**: If you have an idea for a creative addition to this program, please ask us and we may approve it for late day credit. (Extra features that haven't been approved by us will not receive credit.)

Note that regardless of how many of these additions you choose to implement, your main program behavior and appearance should still work as specified. You also may not submit the assignment more than 3 days past its listed due date. If you modify the provided HTML page or CSS style sheet to implement these features, please make sure to submit your modified files.

Implementation and Grading:

Submit your assignment online from the turnin area of the course web site. You will not need to ZIP it; just turn in the `.js` file. For reference, our solution has 95 lines of Javascript.

Fetch the necessary data for the program by opening `XMLHttpRequest` object(s) to retrieve the files. You should process XML data by examining the request's XML tree, not by treating the XML as plain text.

Part of your grade in this assignment comes from choosing proper HTML tags to represent the semantics of the content. Represent the years and their rankings as a definition list, where each year is a term and its ranking is a definition. The provided `.css` file contains CSS classes called `year` and `rank` that do some of the work for you: they set up the bar colors, fonts, margins, and widths of each bar. You'll still need to write the code to set each ranking bar's x/y position and height.

Because the initial appearance of the HTML page shows no data, it will not pass the W3C XHTML validator. An error is given about an "end tag for "`dl`" which is not finished." This particular validation error is acceptable and will not affect your grade.

Your Javascript code should follow reasonable stylistic guidelines similar to those you would follow on a CSE 14x programming assignment. In particular, you should minimize the number of global variables, utilize parameters and return values properly, correctly utilize the HTML DOM objects, correctly use indentation and spacing, and place a comment header at the top of your Javascript file and atop every function explaining that function's behavior. You should only use material that has been discussed during the first five weeks of the course, unless given explicit permission from the instructor.

Two aspects of your Javascript style deserve particular emphasis. You should minimize redundant code. You should also exercise good procedural decomposition, breaking down lengthy operations into multiple functions, including use of parameters and returns over global variables whenever possible.

Please do not place a solution to this assignment online on a publicly accessible (un-passworded) web site.