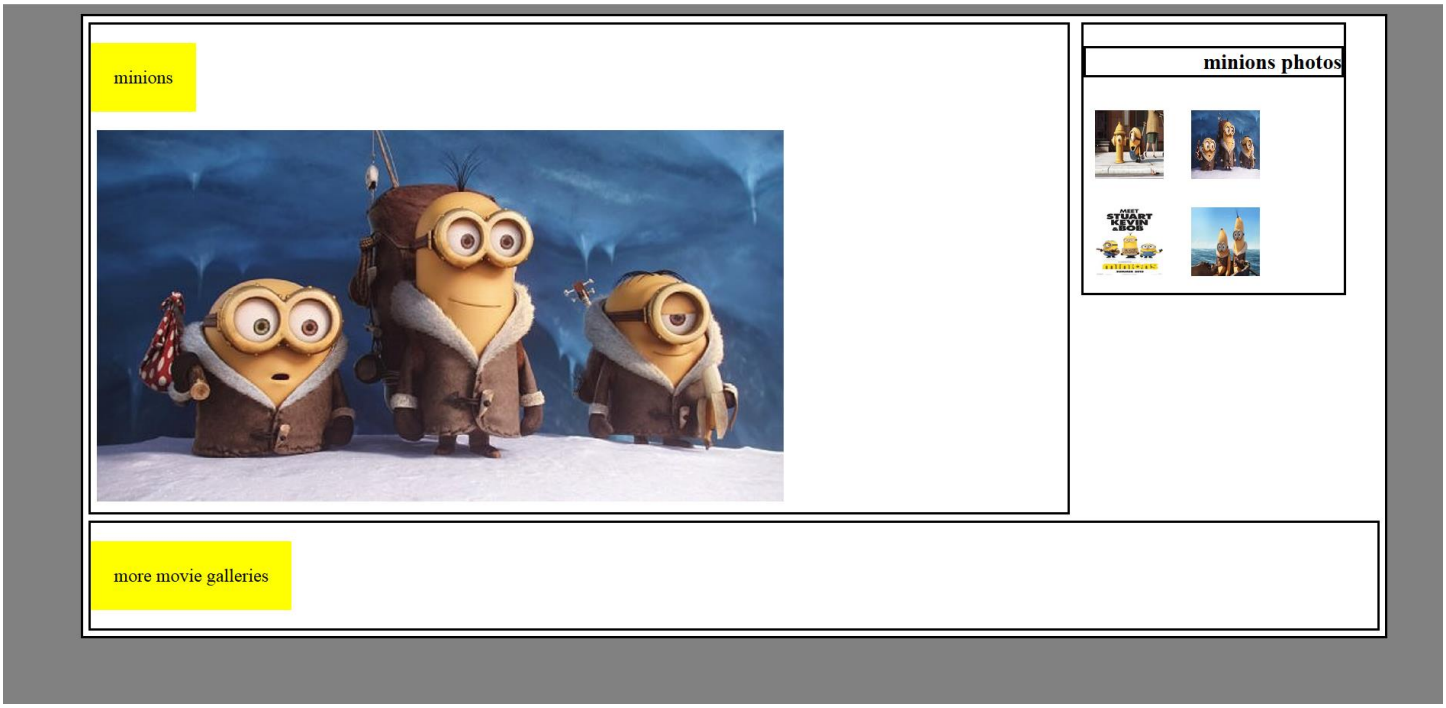
Practice Final 2

1. CSS

Write the **CSS code** necessary to recreate the following appearance on-screen, exactly as shown.
The page uses the HTML code below. You are **not allowed to modify the HTML**.




```
<body>
  <div id="contentArea">
    <div id="bigPic">
      <p>minions</p>
      <div>
        <img src="minions.jpg" alt="big" />
      </div>
    </div>
    <div id="gallery">
      <h3>minions photos</h3>
      <img src="bydrant.jpg" alt="hydrant" />
      <img src="snow.jpg" alt="snow" />
      <img src="meet.jpg" alt="meet" />
      <img src="banana.jpg" alt="banana" />
    </div>
    <div id="more">
      <p>more movie galleries</p>
    </div>
  </div>
</body>
```

All borders shown are 2px thick, solid and black in color.

The elements that contain “minions” and “more movie galleries” have yellow backgrounds and 20px of space between the words and the edge of the background color.

There is 5px empty space around the outside of each div.

The small pictures are 60px wide and 60px tall and have 10px space around them. They should appear on the same line unless the line is full.

The div with the big picture and “minions” title is 75% of the width. The one with the small photos is 20%.

The background color of the page is gray but the background color of the area containing everything is white. The white area is 90% wide and centered.

2. NodeJS

Write the NodeJS code for a web service `filter_service.js` that filters lines of text from a file. The service should take a word as a POST parameter. Your code should examine the contents of `text.txt` and remove any lines from the file that contain the passed in word, case-insensitively. Write the changes to the file so that any future viewings of the file will see the changes. Send back the new contents of `text.txt` as a string.

Match the exact word, not other words that contain it as a substring. For example, if the user submits the word "me" you would filter out lines containing the word "me", but not lines that just contain a word such as "men" or "game".

If the user makes a POST but somehow does not submit the query parameter for the word, or if the word they submit does not consist entirely of upper/lowercase letters, issue an HTTP 400 error and send back an empty string.

3. NodeJS/JSON

Write the code for a web service `courses_service.js` that outputs `JSON` data about available courses at a school. This service should take two `GET` parameters named `start` and `end`, and search a `classes` collection in the `school` database for all courses that match those start/end times exactly and have open seats available.

The database stores the following information about each class:

code startTime endTime seatsAvailable seatsTotal name

The `JSON` that is output should contain a number labeled "`count`". This should represent a count of all courses at the given start and end times. It should also contain a list called "`courses`" that contains a list for each course exactly matching the start and end times that has open seats. The list for each course should contain the code labeled as "`code`", the number of seats left (the total seats minus the seats available) labeled as "`seats`", and the name labeled as "`name`",

You can assume both parameters will be passed. If no courses match the start/end and have seats you should send back the same data as usual, just leave the course list empty.

An example database contents:

```
{code: CSE154, startTime: 130, endTime: 230, seatsAvailable: 250, seatsTotal: 250, name: Web Programming}
{code: CSE143, startTime: 130, endTime: 230, seatsAvailable: 700, seatsTotal: 800, name: Programming II}
{code: ANTH300, startTime: 130, endTime: 230, seatsAvailable: 13, seatsTotal: 14, name: Anthropology}
{code: DANCE250, startTime: 130, endTime: 3, seatsAvailable: 40, seatsTotal: 50, name: World Dance History}
```

Output using the above database and `http://localhost:3000?start=130&end=230`

```
{ "count":3,
  "courses":[ {"code" : "CSE143",
               "seats" : 100,
               "name" : "Computer Programming"
              },
              {"code" : "ANTH300",
               "seats" : 1,
               "name" : "Anthropology"
              }
            ]
}
```

4. Ajax/JSON

Write Javascript code in a file called `courses.js` that contacts the `courses_service.js` code that you wrote for question 3. Remember, `courses_service.js` takes `start` and `end` times as parameters and outputs data in the form of the data below:

```
{ "count":3,
  "courses":[ {"code" : "CSE143",
               "seats" : 100,
               "name" : "Computer Programming"
              },
              {"code" : "ANTH300",
               "seats" : 1,
               "name" : "Anthropology"
              }
            ]
}
```

Your Javascript code will be attached to the below `HTML` page:

```
<!DOCTYPE html>
<html>
   <head><script type="text/javascript" src="courses.js"></script></head>
   <body>
         <h1>Search for open courses</h1>
         <label>start time:<input id="start" type="text" /></label>
         <label>end time:<input id="end" type="text" /></label>
         <button id="search">search</button>
         <ul id="results"></ul>
         <p id="count"></p>
   </body>
</html>
```

When the `search` button is clicked, clear the previous results on the page and request the `JSON` data for the input times with `Ajax`. Add each returned course to the list in the following format:

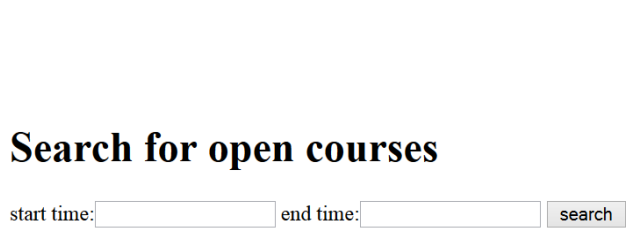
name - ***seats*** *seats*

Add the count to the `count` paragraph in the following format:

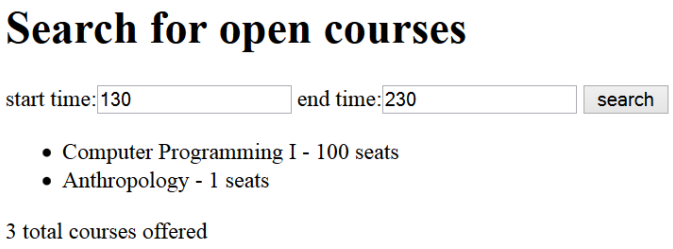
count *total courses offered*

You may assume that the JSON data is valid and in the format described previously, the data typed into the text boxes is valid, and that the service is reachable. You do not need to do anything special if there are no matching courses.

You may not use any Javascript libraries such as Prototype and JQuery.



When the page first opens



After a search for courses 130-230

(Write your solution on the next page)

5. Regular Expressions

a) Write a regular expression to match a **hexadecimal color code**. Remember, colors written in hexadecimal always start with a `#` and then contain 6 letters or numbers `0-9` and `A-F`. Letters can be lowercase or uppercase.

match:	**don't match:**
`#000000`	`#1234567`
`#D3D3D3`	`4#D3D3D3`
`#abCDeF`	`#abCDeG`

b) Write a regular expression to validate a **Mastercard number**. Mastercards have a 16 digit long number. The first number is always `5` and the second number is a `1`, `2`, `3`, `4` or `5`. The rest of the numbers can be anything. You should not look for or try to match dashes (-).

match:	**don't match:**
`5112345678901234`	`51123456789012`
`5555555555555555`	`55555a5555555b55`

c) Write a regular expression to match a **time** written like `11:04 AM`. Times consist of an hour `(1-12)` followed by a colon, followed by minutes `(00-59)`, followed by a space and then either `AM` or `PM`.

match:	**don't match:**
`12:00 AM`	`12:0 AM`
`1:11 PM`	`14:00 PM`
`4:59 PM`	`0:20 AM`
	`02:00 AM`
	`4:60 PM`

[abc]	A single character of: a, b, or c	.	Any single character
[^abc]	Any single character except: a, b, or c	\s	Any whitespace character
[a-z]	Any single character in the range a-z	\S	Any non-whitespace character
[a-zA-Z]	Any single character in the range a-z or A-Z	\d	Any digit
^	Start of line	\D	Any non-digit
$	End of line	\w	Any word character (letter, number, underscore)
\A	Start of string	\W	Any non-word character
\z	End of string	\b	Any word boundary
(...)	Capture everything enclosed	a+	One or more of a
(a|b)	a or b	a{3}	Exactly 3 of a
a?	Zero or one of a	a{3,}	3 or more of a
a*	Zero or more of a	a{3,6}	Between 3 and 6 of a

6. Javascript/DOM

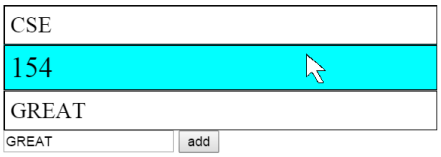
Write JavaScript code in a file called `simple.js` for manipulating a list. The page UI allows the user to type some text into a text box. The user can click an "`add`" button to make the text appear in the list (put it inside the `game` div).

When the mouse enters a list item the background color of that item should turn `cyan` and the `font size` of that list item should increase by 5px. When it leaves that item the background color should turn back to `white` and the `font size` back to what it originally was. **The original font size is included in a CSS file attached to the HTML**. Your code should work for any starting font size value. You can view the current size with your code but not by opening the file and looking at it.

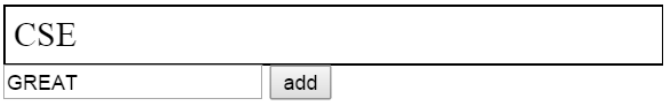
When the user clicks on an item it should be **deleted**.

You may not use any Javascript libraries such as Prototype and JQuery. You must write ALL Javascript to make the page work, none is provided. You can define any additional CSS you like as long as you label it in a clear way. You only need to write the appearance/CSS changes mentioned above. You can assume the page has a linked style sheet that provides the other styles visible in the screenshots.

The page when it first loads



The page after a couple adds with the mouse hovering



The page after "154" and "Great" have been clicked and removed

Your Javascript will be manipulating the following HTML code:

```
<!DOCTYPE html>
<html>
	<head>
		<script type="text/javascript" src="simple.js"></script>
		<link rel="stylesheet" type="text/css" href="simple.css" />
	</head>
	<body>
		<div id="game"></div>
		<input id="input" type="text" />
		<button id="add">add</button>
	</body>
</html>
```

6. SQL

Write an SQL query to search the `world` database for all languages that are spoken as the official language of at least two "newly growing" countries. We will define a "newly growing" country as a country that has both of the following qualities: became independent after the year 1900, and contains at least one city with a population of over one million. For example, Malay would be listed because it is the official language of Malaysia which contains Kuala Lumpur (population 1,297,526) and became independent in 1957, and Indonesia which contains Jakarta (population 9,604,900) and became independent in 1945. Each language should be listed alphabetically and only once**.**

Recall the `world` tables:

code	name	continent	independence_year	population	gnp	head_of_state	...
AFG	Afghanistan	Asia	1919	22720000	5976.0	Mohammad Omar	...
NLD	Netherlands	Europe	1581	15864000	371362.0	Beatrix	...
...							

countries

country_code	language	official	percentage
AFG	Pashto	T	52.4
NLD	Dutch	T	95.6
...			

languages

id	name	country_code	district	population
3793	New York	USA	New York	8008278
1	Los Angeles	USA	California	3694820
...				

cities

```
+----------+
| name     |
+----------+
| Arabic   |
| Chinese  |
| English  |
| French   |
| German   |
| Korean   |
| Malay    |
| Russian  |
| Spanish  |
+----------+
9 rows in set (69 ms)
```

When run on `world` database, your query produces the results at left.

If you join too many tables together that are not needed for the query, you will not receive full credit. You should solve this problem using only the SQL syntax shown in class and the textbook.