В HTML создадим поле ввода **input**, и кнопку **button**. На кнопку «вешаем» обработчик события **onclick**, для запуска нашей программы. После того как программа получила толчок к действию, в права вступает функция **getPalindrome (),** и мы переходим не посредственно к JS.

Создадим переменную **count** - эта переменная будет отвечать за нумерацию списка палиндромов (если таковы будут), присвоим ей **2**, счетчик будет стартовать с номера **“2.”** в списке наших палиндромов, а цифра **“1.”** уйдет самому большому палиндрому из строки, которую ввел пользователь.

В переменной **coincidence**, будем записывать кол-во всех совпадений на палиндром, **count и coincidence** – глобальные переменные.

В функции **getPalindrome ()**, создадим переменную **bigPal_length**(сюда будет, приходит самый большой палиндром, пока присвоим ей **0**) и переменную **bigPal**(которой присвоим **“”** – пустую строку).

Далее создадим переменную **str** и присвоим ей строку, которую ввел пользователь, с помощью **getElementById**. Затем создадим цикл **for.**

Цикл выполняется таким образом:

Создадим переменную **i = 0**(первый символ строки); условие, пока **i < str.length**(длина строки); **i++.** Шаг: **i++** выполняется после тела на каждой итерации, но перед проверкой условия.
В теле цикла создаем переменную **subMain**(главная подстрока) присваиваем ей **str.substr(i, str.length)**, здесь мы берем подстроку, метод **substr(i, str.length)** возвращает подстроку с позиции **i** до, кол-ва символов **str.length**

В этом же цикле создаем второй цикл:

Создадим переменную **j** и присвоим ей **subMain.length**(грубо говоря, последний символ строки); **пока j >= 0**; **j--**

В теле второго цикла создаем переменную **subInside**(внутренняя подстрока) и присваиваем ей **subMain.substr(0, j)** – от **0** до последнего символа строки.

Если длина(**subInside.length**) станет меньше двух символов, директива **continue** прекращает выполнение текущей итерации цикла. Далее идет проверка на палиндром **isPalindrome(subInside)** - мы переходим к функции **isPalindrome(str)**, где собственно и будут, проверяться наши строки на палиндромность (проверку опишем в самом конце). А пока продолжим, если наша функция **isPalindrome(str)** вернула нам true, т.е. палиндром, все ОК идем дальше. Если **subInside.length > bigPal_length**, а

при первом цикле так и будет, так как **bigPal_length = 0**, то переменной **bigPal_length** присвоим длину **subInside.length** и соответственно переменной **bigPal** присваиваем **subInside**, таким образом, мы найдем самый большой палиндром. Конкатенируем **bigPal** с тегом **b** и номером **“1.”** и выведем его на экран нашей страницы через innerHTML, предварительно создавши **div** с классом **.boxLargest**, у нас получится самый большой палиндром с номером “**1.”** в списке и выделенным жирным шрифтом.

Продолжим, если **isPalindrome(subInside)** – **true**, то выполняется еще одна проверка - **else if.**

Если **subInside** не равен **bigPal**, т.е. равен самому большому палиндрому, смело выводим **subInside**(наш палиндром) на экран, опять таки через **innerHTML**, но уже в **div** с классом **.box**, каждый раз перезаписывая **.box** на новый палиндром с помощью **+=**, и конкатенируя **subInside** с **count**(наш счетчик нумерации), таким образом получим палиндром с номером **“2.”**, плюс увеличим счетчик на **1**, чтобы следующий палиндром получил уже номер **“3.”** и так далее.

Если наша функция **getPalindrome ()** не найдет ни одного совпадения на пилиндром, т.е. наша переменная **coincidence,** которая за это отвечает, будет меньше **1** (**coincidence < 1**), то вывести на экран **Palindrome is not found.**

Наконец заключительная часть, проверка на палиндром, создадим функцию **isPalindrome ()**. В этой функции создаем переменную **revPal** и присвоим ей перевернутую строку, делается это очень просто. С помощью метода **split**, превращаем строку в массив, разбив ее по разделителю **(“”)**, затем переворачиваем, с помощью метода **reverse** (этот метод работает только с массивами), и наконец, метод **join**, делает строго противоположное действие методу **split**. В итоге получается вот такая конструкция **str.split("").reverse().join("")**. Если пришедший параметр равен своей зеркально копии, значит, мы нашли палиндром, переменная **coincidence** увеличивается на **1**, и функции возвращается **true.** Как то так.