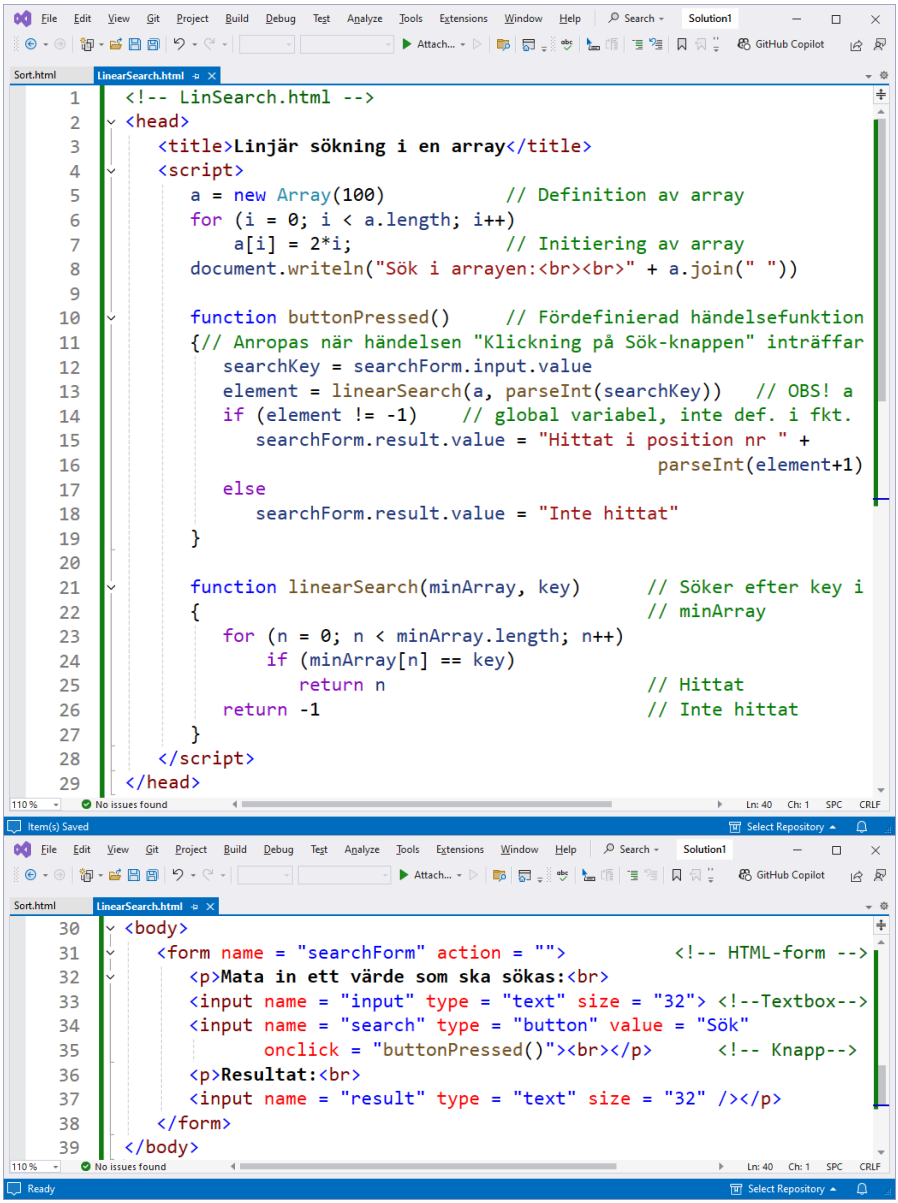
5.7 Sökning i arrays



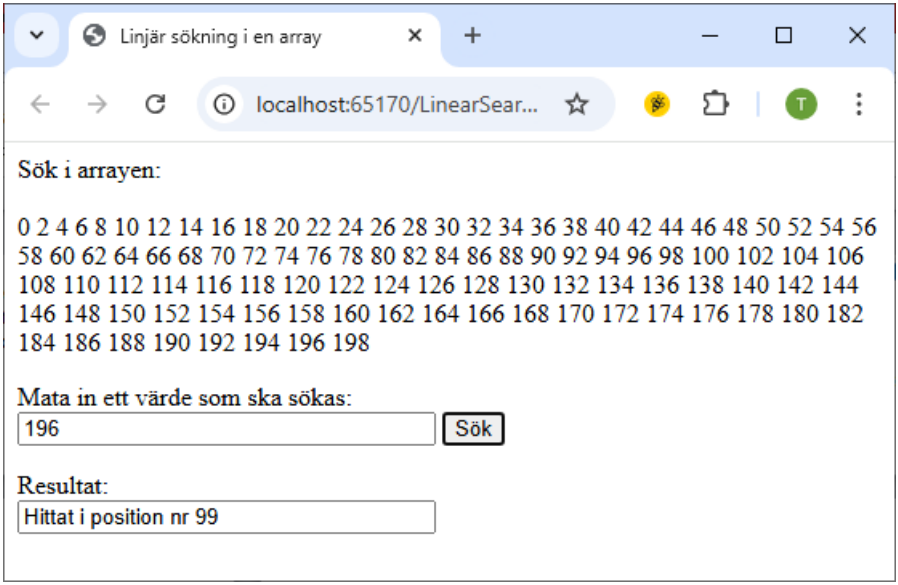
```
<!-- LinSearch.html -->
<head>
   <title>Linjär sökning i en array</title>
   <script>
      a = new Array(100)           // Definition av array
      for (i = 0; i < a.length; i++)
          a[i] = 2*i;              // Initiering av array
      document.writeln("Sök i arrayen:<br><br>" + a.join(" "))

      function buttonPressed()     // Fördefinierad händelsefunktion
      {// Anropas när händelsen "Klickning på Sök-knappen" inträffar
         searchKey = searchForm.input.value
         element = linearSearch(a, parseInt(searchKey))   // OBS! a
         if (element != -1)    // global variabel, inte def. i fkt.
            searchForm.result.value = "Hittat i position nr " +
                                                 parseInt(element+1)
         else
            searchForm.result.value = "Inte hittat"
      }

      function linearSearch(minArray, key)      // Söker efter key i
      {                                         // minArray
         for (n = 0; n < minArray.length; n++)
             if (minArray[n] == key)
                return n                        // Hittat
         return -1                              // Inte hittat
      }
   </script>
</head>
```

```
<body>
   <form name = "searchForm" action = "">          <!-- HTML-form -->
      <p>Mata in ett värde som ska sökas:<br>
      <input name = "input" type = "text" size = "32"> <!--Textbox-->
      <input name = "search" type = "button" value = "Sök"
             onclick = "buttonPressed()"><br></p>      <!-- Knapp-->
      <p>Resultat:<br>
      <input name = "result" type = "text" size = "32" /></p>
   </form>
</body>
```

Linjär sökning

I scriptet **`LinSearch`** ovan har vi implementerat en algoritm för sökning av data i en

array vars körresultat kan t.ex. se ut så här:



Vi letar efter värdet 196 i arrayen ovan som har 100 element. Svaret är att 196 finns på den 99:e positionen, där med ”position” menas platsen när man börjar räkna från 1 – så som vi brukar göra (inte från 0 som datorn brukar göra). Hade vi matat in värdet 0 hade vi fått position nr 1 osv. Detta har implementerats i koden på rad **16**. Matar man in ett värde som inte finns i arrayen får man svaret ”Inte hittat”.

Fördefinierad händelsefunktion

Intressant och ny för oss är en funktion som definieras på raderna **10-19** med huvudet:

```
function buttonPressed()
```

För det första är det inte vi som definierar funktionen utan JavaScript som redan har definierat den. Men vi *definierar om* den. Dvs vi bibehåller *namnet* som JavaScript har föreskrivit, men skriver *kroppen* själva. Det får vi göra eftersom JavaScript tillåter det – ett programmeringstekniskt koncept som används i många språk och kallas för *överlagring* (eng. *Overloading*). Vi överlagrar funktionen `buttonPressed()` genom att definiera en egen variant av den. I så fall får vi inte ändra namnet på den, inte ens vad gäller stora/små bokstäver. Överlagring är ett ofta använt koncept i programmeringen.

För det andra är funktionen `buttonPressed()` inte en vanlig funktion, dvs den anropas inte genom att kalla på dess namn, utan genom att initiera en händelse. Med ”händelse” menas en musklickning, en tangenttryckning eller en systemåtgärd. I det här fallet anropas funktionen genom att vi vid körning klickar på en knapp. Det är

anledningen varför sådana funktioner kallas för *händelsefunktioner*. Närmare bestämt anropas funktionen `buttonPressed()` i scriptet **`LinSearch`** när vi klickar på knappen Sök. Global variabel som parameter

Global variabel som parameter

En annan programmeringsteknisk nyhet är att variabeln `a` används i funktionen `buttonPressed()`, men inte är definierad som en lokal variabel i funktionen. Detta sker i anropet av funktionen `linearSearch()` på rad **`13`**, där `a` är den första parametern:

```
element = linearSearch(a, parseInt(searchKey))
```

`a` är i själva verket en *global* variabel, därför att den är definierad på rad **`5`** dvs *utanför* funktionen `buttonPressed()` och dessutom utanför *alla* funktioner. Detta är möjligt eftersom globala variabler är giltiga i hela programmet och därmed även i alla funktioner som definieras där. `a` är ju arrayen som vi initierar med värden på raderna **`6-7`** och skickar sedan till funktionen `linearSearch()` för sökning.

Det ”vanliga” sättet att överföra arrayen `a` till funktionen `buttonPressed()` hade varit att definiera `buttonPressed()` med parametern `a`, typ `buttonPressed(a)`, och sedan använda den i kroppen. Men vi får inte ändra namnet på funktionen `buttonPressed()` eftersom den är fördefinierad i JavaScript och vi får endast definiera om kroppen, som vi sade tidigare. `buttonPressed(a)` fungerar faktiskt inte i koden.

Det verkar som om användningen av `a` som global variabel är den enklaste lösningen i det här fallet. Annars rekommenderas generellt en restriktiv policy för användningen av globala variabler eftersom de motverkar modulariseringen. Funktioner med globala variabler som prametrar kan inte användas som fristående moduler i andra program. Löser man dem från sitt sammanhang förlorar den globala variabeln sin giltighet.