Kapitel 5

Arrays

5.1 Deklaration och initiering av arrays

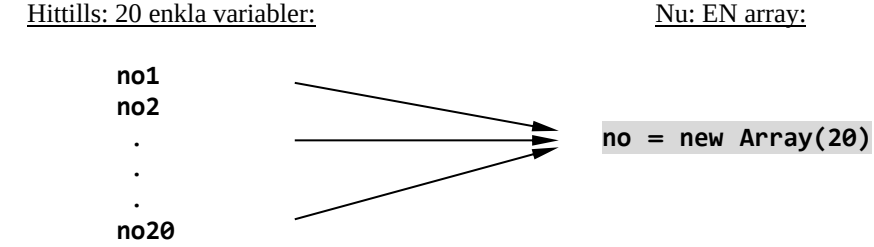
Datorn har några egenskaper som är helt överlägsna motsvarande egenskaper hos människan: snabbheten, noggrannheten och förmågan att effektivt lagra och hantera stora datamängder samt förmågan att aldrig bli trött.

Vi ska i detta avsnitt introducera ett verktyg som utnyttjar en av dessa överlägsna egenskaper, nämligen att kunna effektivt lagra och hantera *stora datamängder*. Detta verktyg heter *array* och betyder *ordnad uppställning* (*battle array* = stridsordning), en ordnad skara av data. Ibland används i litteraturen begreppet *fält* som är identiskt med *array*.

En *array* är en ordnad mängd av variabler grupperade under ETT namn.

Arrayens delar kallas för *element*. Elementens position kallas för *index*.

Vi kan gruppera t.ex. 20 variabler i en array med 20 element:



Hittills behövde vi skriva *20* satser för att skapa 20 variabler. Men nu har vi möjligheten att göra samma sak med endast *en* sats, genom att skapa *en enda* variabel – visserligen inte längre en vanlig variabel utan en *arrayvariabel* – och lägga till informationen om antalet element i den. På så sätt har vi skapat en *arrayvariabel* **no**.

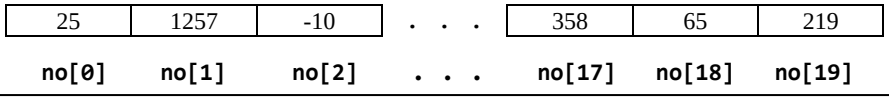
Arrayvariabeln **no** ersätter de 20 vanliga variablerna **no1**, **no2**, …, **no20** och består nu i sin tur av 20 *element*. Varje element är en variabel som kan lagra ett värde. Enda skillnaden är *sättet* dvs *koden* att komma åt dessa värden. Indexet är ett nummer som specificerar varje elements position i arrayen. Varje element i en array kan betraktas som en *indexerad* dvs *numrerad variabel*.

En array är inte längre en enkel utan en s.k. *sammansatt* datatyp. En *enkel datatyp* representerar ETT värde åt gången, t.ex. ett heltal, ett deci-maltal, ett tecken, ett sanningsvärde osv. En *sammansatt datatyp* representerar fler än ett värde åt gången, t.ex. flera heltal, flera flyttal, flera tecken, flera sanningsvärden osv. Man kan gruppera enkla datatyper till den sammansatta datatypen array.

Åtkomst till arrayens element

Följande sats definierar arrayen **no**: `no = new Array(20)`
Den allokerar (reserverar) 20 minnesceller för lagring av 20 värden. Låt oss anta att t.ex. vissa värden tilldelats arrayen **no**:s element, som man ser på bilden nedan. Eftersom elementen i en array alltid lagras i ett sammanhängande minnesområde, uppstår följande minnesbild:

Minnesbild av arrayen no:



Bilden visar hur indexeringen av element i en array organiseras. I raden under minnescellerna står hur JavaScript-kod kommer åt varje element i en array. Det är anmärkningsvärt är att indexnumreringen börjar med **0**, medan vi människor är vana vid att påbörja numreringen av ett antal objekt med **1**. Följande indexregel gäller:

> Indexregeln: I arrays börjar numreringen av index alltid med 0.
>
> Därför gäller: elementets position = index + 1

Med *position* menas numret som människan använder för att räkna elementen, medan kodens numrering – det som står inom hakparenteserna **[ ]** – kallas för *index*.

Det 1:a elementet i arrayen **no** ovan har index **0** och värdet 25, medan positionen är 1. JavaScript kodar elementet med **no[0]**. Det 2:a elementet har index **1** och värdet 1257 medan koden är **no[1]**. Det 3:e elementet: index **2**, värdet –10 och koden **no[2]** osv. Det **n**:e elementet har alltid index **n-1**. Därför har också det 20:e elementet index **19** och värdet 219. Det gäller att hålla isär det mänskliga sättet att numrera som börjar med 1 från JavaScript-kodens sätt att indesera som börjar med **0**. Vi har definierat 20 variabler **no[0], ..., no[19]**. Antalet element är 20. Indexen går från **0** till **19**.

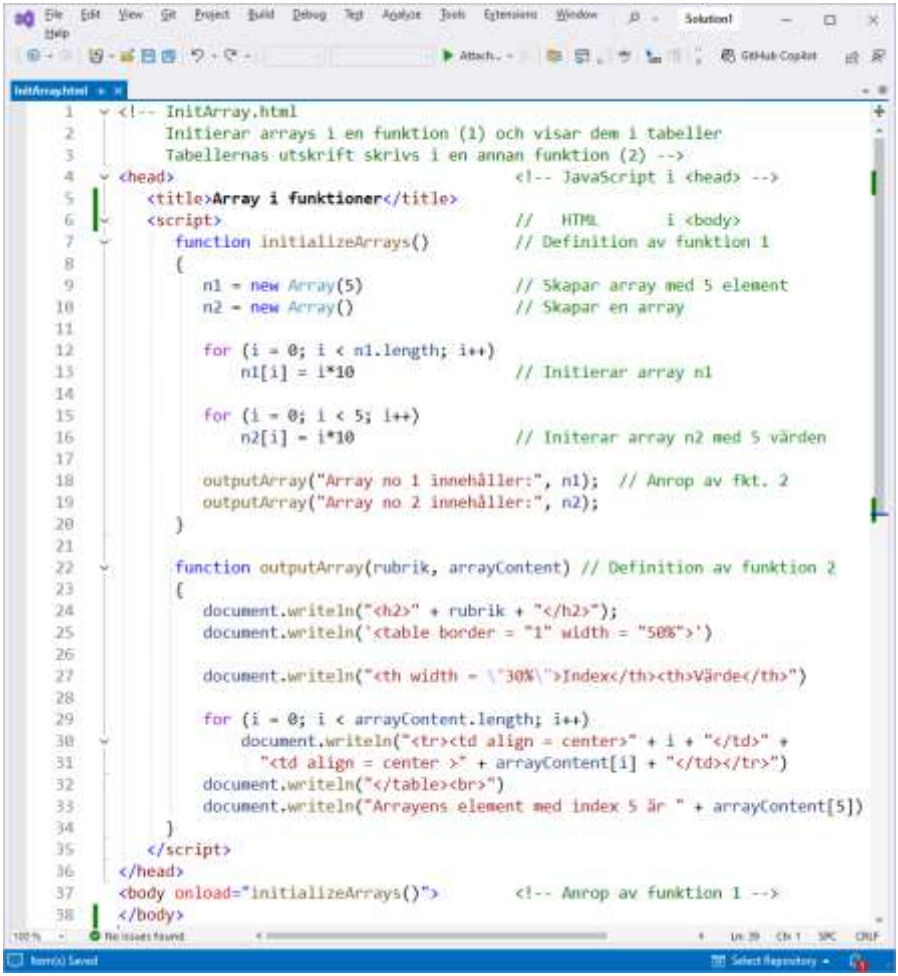
Av indexregeln följer dessutom att negativa index generellt inte är tillåtna.

Array i funktioner

Programmet **InitArray** på nästa sida demonstrerar allt vi sagt om arrays speciellt indexregeln. Dessutom kan vi se, hur JavaScript hanterar överskridningen av de definierade indexgränserna.

En annan egenskap av **InitArray** är att den är modularierad. Scriptet kombinerar arrays med funktioner, genom att definiera deklarationen och initieringen av arrays i en funktion (1) och tabellutskriften av arrayens innehåll i en annan funktion (2).

Medan funktion 1 anropar funktion 2, anropas den själv i scriptets **body**-del. Annars är funktionernas definition placerade i scriptets **head**-del.



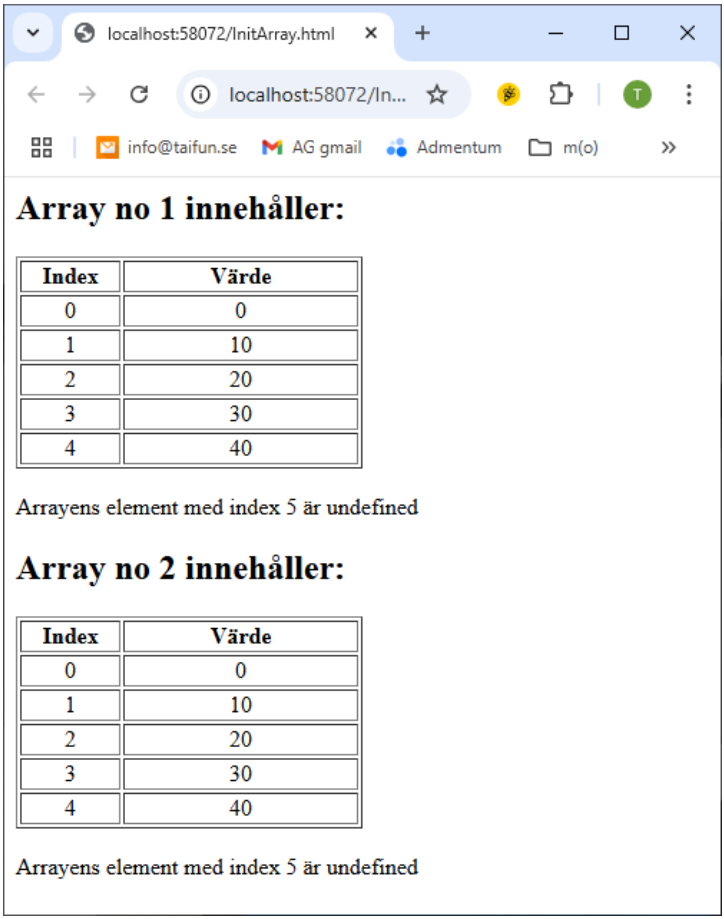
Odefinierade element i en array

Körningen visar att icke-definierade arrayelement inte leder till något fel. Index **5** överskrider de definierade indexgränserna. Arrayelementet **arrayContent[5]** på rad **33** är varken definierat eller tilldelat något värde. Ändå kan man skriva det i koden och köra programmet. Inte ens en varning påpekar att man använt kod som är odefinierad. Anledningen är följande:

I en array kontrollerar JavaScript endast arraynamnet, inte indexen.

Arrayelement som överskrider indexgränserna blir **”undefined”**.

En körning visar detta:



Ett annat namn än det definierade arraynamnet **`arrayContent`** leder till fel. Om vi däremot använder ett index som överskrider de definierade gränserna, kan vi fortfarande exekvera koden. Ansvaret för kontroll av indexgränserna ligger helt och hållet hos programmeraren. Skälet för denna liberala attityd är bl.a. strävan efter snabbhet, vilket förstås är på bekostnad av säkerheten.