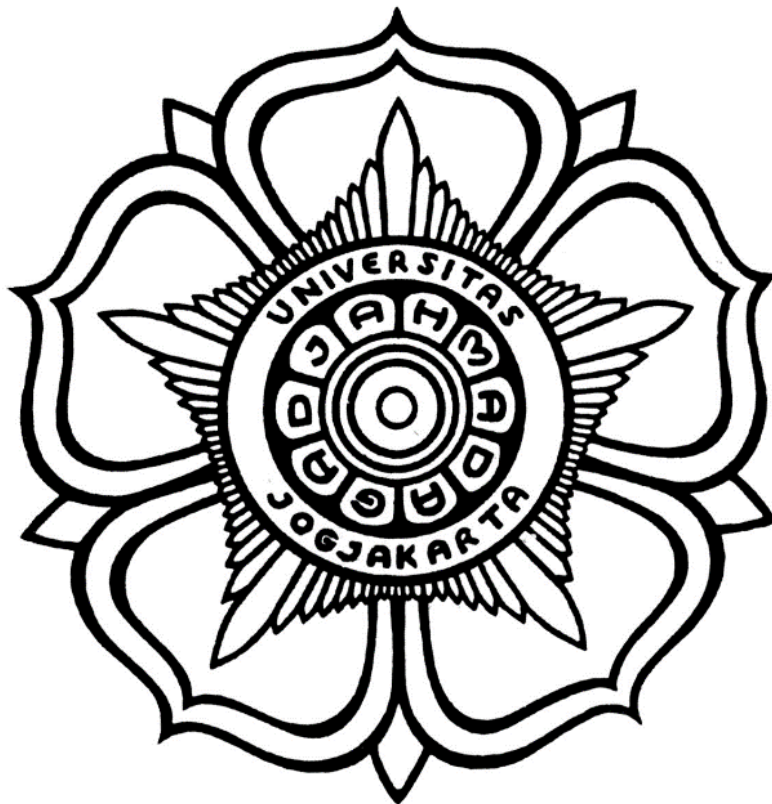


**LAPORAN PRAKTIKUM  
PEMROGRAMAN WEB III  
PERTEMUAN KE-3**



Oleh:  
Maulana Adam Sahid (18/431735/SV/15706)

**D3-KOMPUTER DAN SISTEM INFORMASI  
DEPARTEMEN TEKNIK ELEKTRO DAN INFORMATIKA  
SEKOLAH VOKASI – UNIVERSITAS GADJAH MADA  
2021**

## Laporan Praktikum PWEB3A 2021

### A. Membedakan Var dan Let

i.

```
1 <!DOCTYPE html>
2 <html lang="en" dir="ltr">
3   <head>
4     <meta charset="utf-8">
5     <title></title>
6   </head>
7   <body>
8     <script>
9       var a=1;
10      console.log("hasil awal: "+a);
11      var a=4;
12      console.log("hasil akhir: "+a);
13    </script>
14  </body>
15 </html>
```

hasil awal: 1	Awal.html:10
hasil akhir: 4	Awal.html:12
>	

Pada kasus di atas value dari variabel a berubah menjadi value yang sesuai dengan deklarasi yang terakhir, yaitu 4.

ii.

```
<script>
  let a=1;
  console.log("hasil awal: "+a);
  let a=4;
  console.log("hasil akhir: "+a);
</script>
```

✖ Uncaught SyntaxError: Identifier 'a' has already been declared	Awal.html:11
>	

Pada kasus ini terjadi error. Hal ini karena adanya 2 kali pendeklarasian suatu variabel yang sama dengan menggunakan let pada satu scope yang masih sama.

iii.

```
<script>
  var a=1;
  console.log("hasil awal: "+a);
  a=4;
  console.log("hasil akhir: "+a);
</script>
```

hasil awal: 1	Awal.html:10
hasil akhir: 4	Awal.html:12
>	

Skrup dan hasil pengujian di atas menunjukkan bahwa value dari a yang dideklarasikan dengan var dapat diubah valuenya tanpa meredeclarasi variabel tsb.

iv.

```
<script>
  a=1;
  console.log("hasil awal: "+a);
  var a=4;
  console.log("hasil akhir: "+a);
</script>
```

hasil awal: 1	Awal.html:10
hasil akhir: 4	Awal.html:12
>	

Value dari "a" dapat berubah karena pada dasarnya sintaks "a=1;" merupakan hoisting. Yang mana berarti "a" sebelumnya secara default sudah dideklarasikan terlebih dahulu dengan var barulah kemudian "a" diberikan nilai 1.

v.

```

<script>
  a=1;
  console.log("hasil awal: "+a);
  let a=4;
  console.log("hasil akhir: "+a);
</script>

```

Uncaught ReferenceError: Cannot access 'a' before initialization at Awal.html:9

Terjadi error karena pada kasus tersebut ingin menghoist “a” tetapi pada scope tersebut terdapat pendekalrasian “a” dengan let. Sehingga hoisting tidak bisa dilakukan karena let tidak mengizinkan hoisting.

vi.

```

<script>
  let a=1;
  console.log("hasil awal: "+a);
  a=4;
  console.log("hasil akhir: "+a);
</script>

```

hasil awal: 1  
hasil akhir: 4

Value “a” yang semula adalah 1 berubah menjadi 4. Hal tersebut dapat terjadi meskipun dengan let karena operasi pengubahan value “a=4;” terjadi setelah pendeklarasian variabel a dengan let.

vii.

```

<script>
  let a=1;
  console.log("hasil awal: "+a);
  var a=4;
  console.log("hasil akhir: "+a);
</script>

```

Uncaught SyntaxError: Identifier 'a' has already been declared at Awal.html:11

Variabel “a” sebelumnya dideklarasikan dengan let terlebih dahulu maka tidak bisa dideklarasikan ulang. Variabel tersebut hanya bisa diubah valuenya saja. Jika ingin meredeklarasikan, mesti membuat scope turunan terlebih dahulu dan barulah pada scope tsb. dapat dideklarasikan ulang namun hanya dapat dengan let dan itu juga hanya berlaku pada local scope tsb.

viii.

```

<script>
  var a=1;
  console.log("hasil awal: "+a);
  let a=4;
  console.log("hasil akhir: "+a);
</script>

```

Uncaught SyntaxError: Identifier 'a' has already been declared at Awal.html:11

“a” sudah dideklarasikan dengan var maka tidak bisa dideklarasikan dengan let. Karena pendeklarasian var dan let tsb. masih dalam satu tingkatan scope yang sama.

ix.

```

<script>
let x = 1;
for(let x=1; x<4; x++){
  console.log('dalam perulangan nilai x = '+x);
}
console.log('di luar perulangan nilai x = '+x);
// var a=1;
// console.log("hasil awal: "+a);
// let a=4;
// console.log("hasil akhir: "+a);
</script>

```

	Filter	Default levels ▾	⚙
dalam perulangan nilai x = 1		<a href="#">Awal.html:11</a>	
dalam perulangan nilai x = 2		<a href="#">Awal.html:11</a>	
dalam perulangan nilai x = 3		<a href="#">Awal.html:11</a>	
di luar perulangan nilai x = 1		<a href="#">Awal.html:13</a>	

Dengan menggunakan let, maka nilai suatu variabel dengan let tsb. tidak akan mempengaruhi variabel yg sama di luar scope let tersebut. Sehingga let yang lebih lokal tidak akan mengubah nilai let yang lebih global. Let pada dasarnya hanya berlaku pada tingkatan scope di mana ia dideklarasikan dan pada scope-scope turunannya apabila tidak didefinisikan kembali variabel let tsb. pada scope-scope turunan tersebut.

X.

```

<script>
var x = 1;
for(var x=1; x<4; x++){
  console.log('dalam perulangan nilai x = '+x);
}
console.log('di luar perulangan nilai x = '+x);
// var a=1;
// console.log("hasil awal: "+a);
// let a=4;
// console.log("hasil akhir: "+a);
</script>

```

	Filter	Default levels ▾	⚙
dalam perulangan nilai x = 1		<a href="#">Awal.html:11</a>	
dalam perulangan nilai x = 2		<a href="#">Awal.html:11</a>	
dalam perulangan nilai x = 3		<a href="#">Awal.html:11</a>	
di luar perulangan nilai x = 4		<a href="#">Awal.html:13</a>	

Menggunakan var pada suatu scope yang lebih lokal akan tetap mengubah nilai variabel yg sama pada tingkatan scope global. Karena var merupakan pendeklarasian variabel yang bersifat secara global. Sehingga meskipun dideklarasikan pada scope yang lebih lokal, maka var tersebut masih berlaku di luar local scope pendeklarasian var tersebut.

## B. Menampilkan Nilai Variabel

Pada Javascript (JS) untuk menandai sesuatu yang bertipe string dapat dengan mengapit dengan menggunakan petik tunggal ( ' '), petik ganda ( " " ), dan tick ( ` ` ). Sebagai contoh dapat dilihat pada kode Javascript berikut beserta hasil testingnya.

```

20 //B
21 var text1 = 'Praktikum';
22 var text2 = "Pemrograman Web";
23 console.log('isi text1 adalah = '+text1);
24 console.log('isi text2 adalah = '+text2);
25 console.log(` ${text1} ${text2}`);

```

isi text1 adalah = Praktikum	<a href="#">Awal.html:23</a>
isi text2 adalah = Pemrograman Web	<a href="#">Awal.html:24</a>
Praktikum Pemrograman Web	<a href="#">Awal.html:25</a>

Adapun untuk melakukan interpolasi variabel dari dalam string harus menggunakan tick. Dan dalam pemanggilannya diawali dengan tanda dollar (\$) lalu diikuti nama variabel yang hendak dipanggil dalam apitan kurung kurawal ( {}) seperti pada contoh sebelumnya.

Apabila tidak menggunakan tanda *tick* dan menggunakan kutip tunggal ataupun kutip ganda, maka tanda \$ dan { } akan dibaca juga sebagai string dan dioutputkan sebagai string juga. Termasuk apa yang ada dalam kurung kurawal juga akan dibaca sebagai string yg harus dioutputkan. Sehingga luarannya menjadi “\${text1} \${text2}”. Karena petik tunggal dan petik ganda akan membaca semua hal yang ada di dalamnya sebagai string. Dan karena itu tidak dimungkinkan melakukan interpolasi dari dalam petik tunggal maupun petik ganda.

```
25 console.log('${text1} ${text2}');
```

Awal.html:25

```
25 console.log("${text1} ${text2}");
```

Awal.html:25

Dan dengan menggunakan tanda *tick* kita bisa juga menuliskan string dengan banyak baris tanpa harus menyisipkan karakter backslash (\) dan karakter 'n' untuk membuat baris baru dalam string tersebut.

Jika biasanya kita menggunakan dengan cara berikut:

```
27 var contoh1 = "Contoh string dengan\n"+
28               "baris-baris baru menggunakan\n"+
29               "backslash dan 'n'.";
30 console.log(contoh1);
```

Awal.html:30

Contoh string dengan  
baris-baris baru menggunakan  
backslash dan 'n'.

Maka dengan tanda *tick* kita dapat melakukannya dengan contoh berikut:

```
32 var contoh2 = `Contoh string dengan
33 baris-baris baru tanpa
34 backslash dan 'n',
35 menggunakan tick (\` \`).`;
36 console.log(contoh2);
```

Awal.html:36

Contoh string dengan  
baris-baris baru tanpa  
backslash dan 'n',  
menggunakan tick (` `).

Namun mesti diingat juga bahwa apapun yang ada diantara tanda *tick* tsb. juga akan tetap dianggap sebagai bagian dari rangkaian string tsb juga, termasuk karakter-karakter spasi di dalamnya.

## C. Array

Array adalah tipe data variabel yang dapat menampung satu atau lebih kelompok data. Pada Javascript terdapat dua array yang berbeda, yaitu array berindeks dan array asosiatif.

Array berindeks adalah array yang memiliki indeks seperti deret array pada umumnya yang berurutan/*sequential ordered* mulai dari indeks ke-0 hingga indeks ke-n yg terakhir. Indeks tersebut digunakan untuk pemanggilan sebuah data dari suatu array. Dalam Javascript array berindeks ini

dideklarasikan dengan tanda kurung siku [ ] dan tiap datanya satu dengan yang lainnya dipisahkan oleh tanda koma (,).

```
39 var deret = [11,22,33,44,55];
40 //no. indeks 0, 1, 2, 3, 4
41
42 console.log(deret[0]); //Luarannya 11
43 console.log(deret[2]); //Luarannya 33
44 console.log(deret); //Luarannya seluruh array
```

11	Awal.html:42
33	Awal.html:43
► (5) [11, 22, 33, 44, 55]	Awal.html:44

Sedangkan array asosiatif adalah array yang tiap datannya memiliki memiliki suatu keyword yang digunakan untuk memanggil data tersebut. Array asosiatif dideklarasikan dengan *curly braces* {} dan tiap data yang satu dengan lainnya dipisahkan oleh tanda koma (,). Dan pada setiap data arraynya ditulis dengan format (keyword data dalam string):(value data).

```
47 //Associative Array
48 var larik = {'univ':'UGM', "prodi":"Komsi", "semester":6};
49
50 console.log(larik['univ']); //Luaran UGM
51 console.log(larik['semester']); //Luaran 6
52 console.log(larik['prodi']); //Luaran Komsi
53 console.log(larik); //Luaran semua data array
```

UGM	Awal.html:50
6	Awal.html:51
Komsi	Awal.html:52
► {univ: "UGM", prodi: "Komsi", semester: 6}	Awal.html:53

## D. Fungsi

Fungsi adalah sebuah blok kode yang tertata dan dapat digunakan ulang untuk melaksanakan suatu perintah/operasi tertentu. Fungsi pada umumnya memiliki nilai balik yang dituliskan pada bagian akhir fungsi dengan sintaks `return`. Dan nilai balik tersebut kemudian akan diberikan pada pemanggil fungsi tersebut.

Tetapi terdapat pula fungsi yang tidak memiliki nilai balik tersebut. Fungsi yang tidak memiliki nilai balik tersebut merupakan fungsi yang hanya berisikan perintah-perintah suatu operasi dan tidak memberikan nilai apapun pada pemanggilnya. Fungsi tanpa nilai balik ini biasanya disebut juga sebagai *method* dalam bahasa pemrograman lainnya.

```
55 //D Function
56 function cetakBalik(a,b) {
57     c = a*b;
58     return c;
59 }
60 hasil_1=cetakBalik(4,5);
61 console.log(hasil_1);
62
63 function cetakLog(a,b) {
64     for(let i=0;i<b;i++){
65         console.log(a);
66     }
67 }
68 hasil_2 = cetakLog(2,4);
69 console.log(hasil_2);
```

20	Awal.html:61
4 2	Awal.html:65
undefined	Awal.html:69

Pada contoh kasus di atas fungsi `cetakBalik()` memberikan sebuah value pada pemanggilnya, yaitu kepada `hasil_1`. Variabel `hasil_1` diberikan value dengan nilai balik dari fungsi `cetakBalik` yaitu sebuah int bernilai 20 yang di dapat dari nilai 'c' yang merupakan hasil dari perkalian 'a' dengan 'b' di mana 'a' bernilai 4 dan 'b' bernilai 5.

Sedangkan pada `hasil_2` yang memanggil fungsi `cetakLog()` tidak mendapatkan nilai balik sehingga ketika dicetak pada log konsol luarannya adalah `undefined`. Hal tersebut karena memang belum memiliki nilai dan tidak mendapat nilai balik dari fungsi `cetakLog()`. Namun fungsi `cetakLog()` tersebut ketika dipanggil oleh `hasil_2` ia melakukan suatu proses yaitu mencetak sebuah integer bernilai 2 sebanyak 4 kali sesuai dengan nilai yang dimasukkan pada parameter 'a' dan 'b' pada fungsi `cetakLog()` tersebut.

Dalam Javascript kita dapat menuliskan nilai default pada parameter sebuah fungsi. Nilai default tersebut akan digunakan sebagai nilai pengganti parameter yang dikosongi ketika sebuah fungsi dipanggil dengan sebagian ataupun seluruh parameternya belum diisi.

```
72 function mhs1(nama,umur,prodi) {
73     console.log(`${nama} adalah mahasiswa dari prodi ${prodi} dengan umur ${umur} tahun.`);
74 }
75
76 mhs1("Udin",18,"RPL");
77 mhs1("Idin");
78
79 function mhs2(nama="Jin",umur=19,prodi="TRPL") {
80     console.log(`${nama} adalah mahasiswa dari prodi ${prodi} dengan umur ${umur} tahun.`);
81 }
82
83 mhs2("Jun");
```

Udin adalah mahasiswa dari prodi RPL dengan umur 18 tahun.	<a href="#">Awal.html:73</a>
Idin adalah mahasiswa dari prodi undefined dengan umur undefined tahun.	<a href="#">Awal.html:73</a>
Jun adalah mahasiswa dari prodi TRPL dengan umur 19 tahun.	<a href="#">Awal.html:80</a>

Pada kasus di atas bisa kita lihat bahwa fungsi `mhs1` tidak diberikan nilai default untuk parameter-parameternya. Sehingga ketika ada suatu panggilan yang hanya memberikan parameter pertama, yaitu nama, maka yang terjadi adalah luarannya hanya akan mengisikan nama “Idin” tsb. sedangkan parameter yang lain seperti umur dan prodi belum memiliki nilai (`undefined`) seperti pada hasil log konsol di atas.

Sedangkan pada fungsi `mhs2` sudah memiliki nilai default sehingga ketika ada suatu pemanggilan yang hanya memberikan parameter satu parameter maka nantinya nilai-nilai dari parameter yang dikosongi akan diberikan nilai default tersebut. Sebagai contoh adalah kasus pemanggilan `mhs2(“Jun”)` yang mana luarannya adalah dengan nama “Jun” itu sendiri karena sudah diisikan, sedangkan untuk umur dan prodinya mengikuti nilai default sehingga prodi dan umur yang keluar adalah TRPL dan 19.

## E. Fungsi Tanpa Nama

Dalam Javascript, sebuah fungsi dapat dibuat tanpa menggunakan nama tertentu yang biasa disebut *anonymus function*. Fungsi tersebut kemudian dapat dianggap menjadi sebuah nilai. Sebagai contoh fungsi tersebut dapat dimasukkan pada suatu variabel. Sehingga variabel tersebut memiliki nilai berupa fungsi. Dan ketika dioutputkan tanpa memberikan suatu parameter, maka yang akan dikeluarkan adalah fungsi yang ada pada variabel tersebut.

```

85     var tambah = function (a,b) {
86         return a+b;
87     }
88     console.log(tambah);
89     console.log(tambah(4,5));
90
91     var kurang = function (a,b) {
92         return a-b;
93     }
94     console.log(kurang);
95     console.log(kurang(4,5));

```

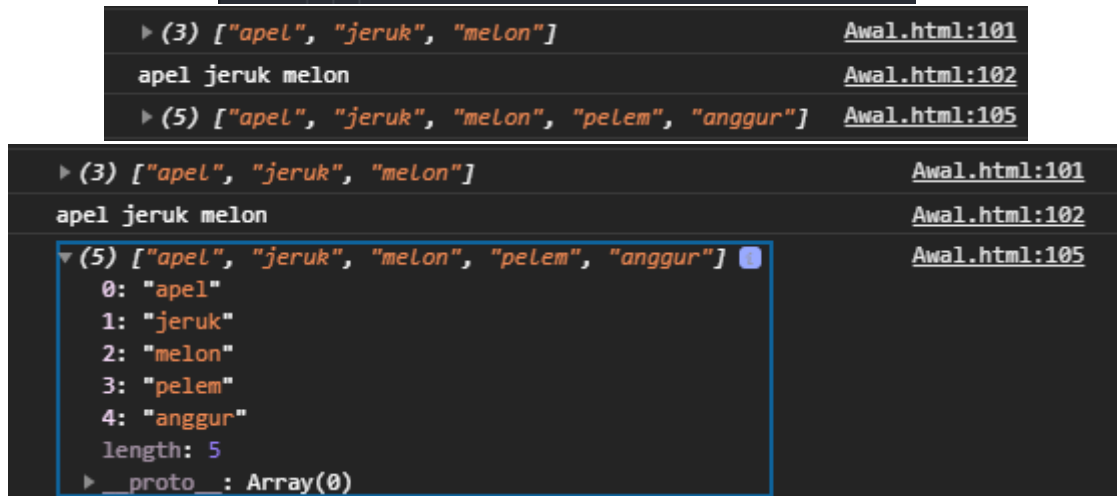
<code>f (a,b) {     return a+b; }</code>	<a href="#">Awal.html:88</a>
<code>9</code>	<a href="#">Awal.html:89</a>
<code>f (a,b) {     return a-b; }</code>	<a href="#">Awal.html:94</a>
<code>-1</code>	<a href="#">Awal.html:95</a>
<code>&gt;</code>	



## F. Spread Operator

Spread operator adalah sebuah sintaks pada Javascript untuk memanggil sebuah array sehingga array yang dipanggil tsb. menjadi satu tingkatan dengan value yang ada pada si pemanggil tersebut. Untuk menggunakan spread operator yaitu dengan menggunakan triple dots (...) kemudian diikuti dengan nama variabel array yang hendak dipanggil. Sebagai contoh adalah seperti pada sintaks di bawah berikut.

```
99 //F Spread Operator
100 var buah = ['apel','jeruk','melon'];
101 console.log(buah);
102 console.log(...buah);
103
104 var buah2= [...buah, 'pelem', 'anggur'];
105 console.log(buah2);
```

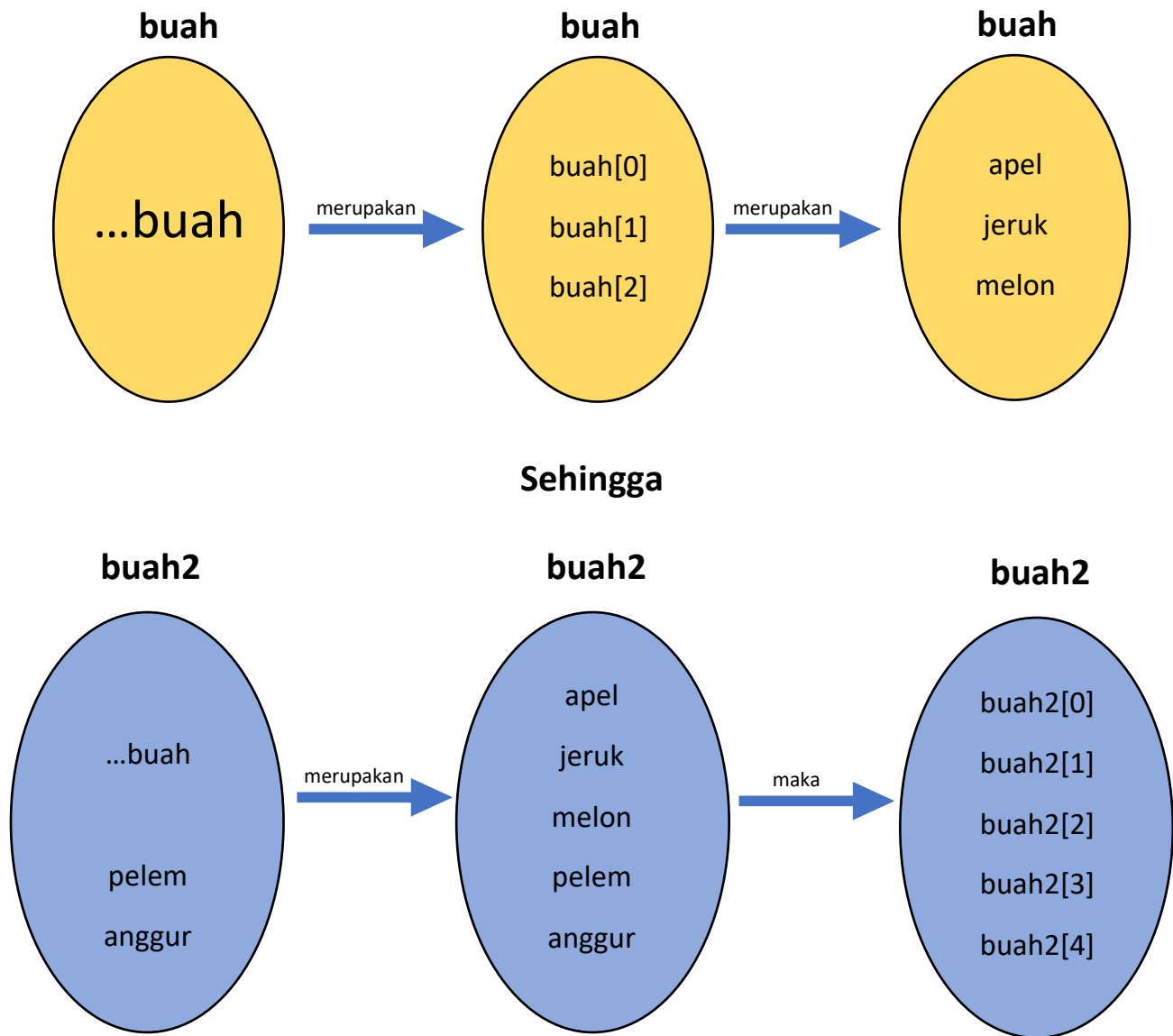


```
▶ (3) ["apel", "jeruk", "melon"] Awal.html:101
apel jeruk melon Awal.html:102
▶ (5) ["apel", "jeruk", "melon", "pelem", "anggur"] Awal.html:105
```

```
▶ (3) ["apel", "jeruk", "melon"] Awal.html:101
apel jeruk melon Awal.html:102
▼ (5) ["apel", "jeruk", "melon", "pelem", "anggur"] Awal.html:105
  0: "apel"
  1: "jeruk"
  2: "melon"
  3: "pelem"
  4: "anggur"
  length: 5
  __proto__: Array(0)
```

Sekilas memang terlihat mirip antara luaran sintaks “buah” dengan luaran sintaks “...buah”. Tetapi kedua hal tersebut sebenarnya sedikit berbeda. Bukan hanya luaran “buah” yang terlihat benar-benar seperti array karena diapit oleh kurung siku dan tiap data dipisahkan oleh tanda koma. Juga bukan sekadar luaran “...buah” yang terlihat seperti sebagai rangkaian string. Tetapi yaitu berbeda pada pengartian “...buah” yang sebenarnya diartikan sebagai “**buah[0],buah[1],buah[2]**” yang mana **tanpa ada kurung siku yang mengapit ketiga data** tersebut sehingga hal itu membuatnya dianggap oleh si pemanggil sebagai **tiga data yang terpisah**. Sehingga pada `console.log(...buah)` itu diartikan sebagai `console.log(buah[0],buah[1],buah[2])`.

Sehingga pada **console.log(buah2)** yang pada dasarnya berarti **console.log(...buah, "pelem", "anggur")** diartikan menjadi **console.log(buah[0], buah[1], buah[2], "pelem", "anggur")** yang kemudian diartikan lagi menjadi **console.log("apel", "jeruk", "melon", "pelem", "anggur")**. Hal tersebut dapat divisualisasikan kurang-lebihnya sebagai berikut:



Sehingga dengan begitu tiap data dalam `"...buah"` menjadi nilai data tersendiri dalam array `"buah2"`.

Hal ini akan berbeda jika kita melakukan tanpa memberikan tiga tanda titik seperti pada skrip kode berikut:

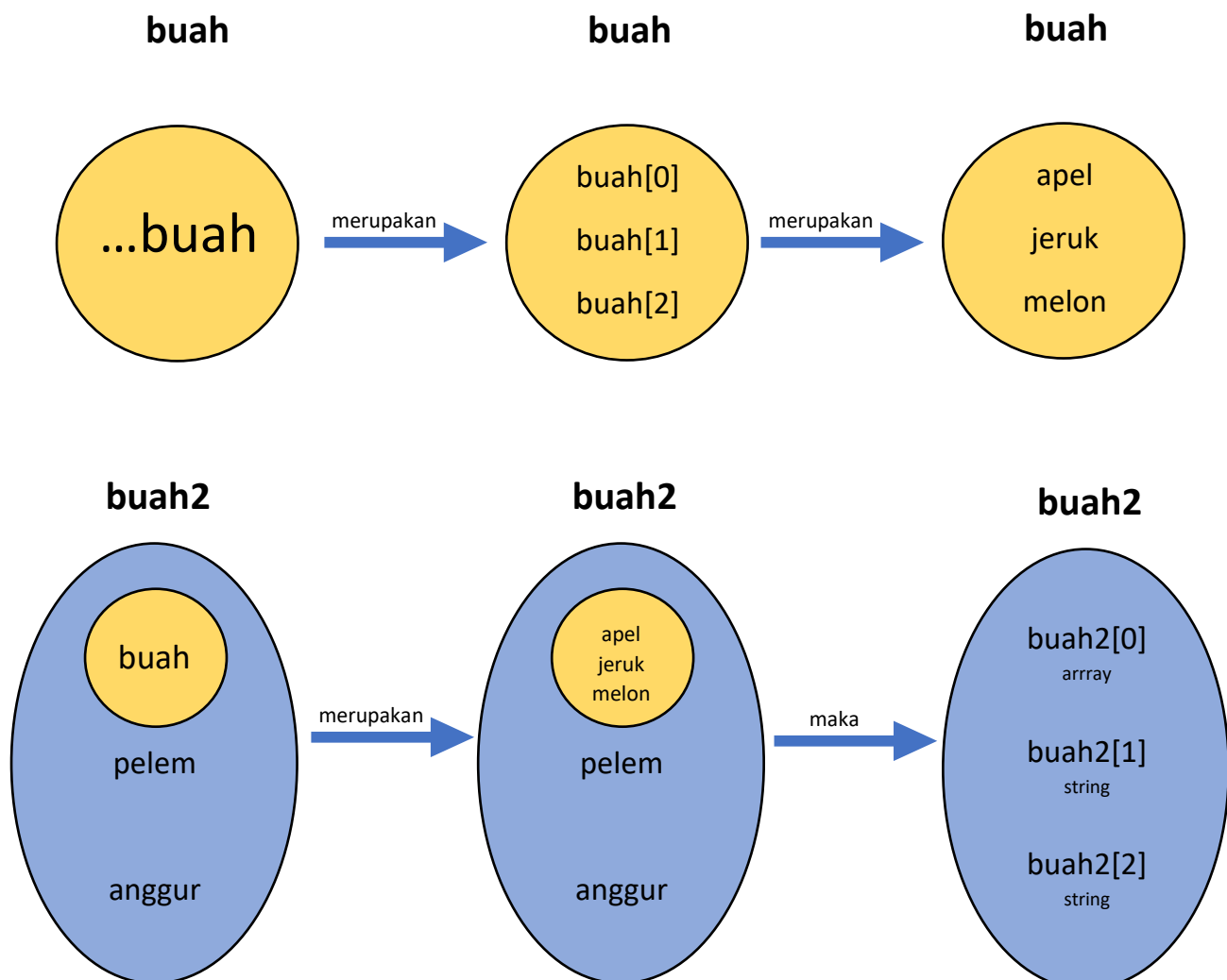
```
99 //F Spread Operator
100 var buah = ['apel','jeruk','melon'];
101 console.log(buah);
102 console.log(...buah);
103
104 var buah2= [buah, 'pelem', 'anggur'];
105 console.log(buah2);
```

Maka hasilnya akan menjadi berbeda seperti berikut:

```
▶ (3) ["apel", "jeruk", "melon"] Awal.html:101
apel jeruk melon Awal.html:102
▶ (3) [Array(3), "pelem", "anggur"] Awal.html:105
```

```
▶ (3) ["apel", "jeruk", "melon"] Awal.html:101
apel jeruk melon Awal.html:102
▼ (3) [Array(3), "pelem", "anggur"] Awal.html:105
  ▶ 0: (3) ["apel", "jeruk", "melon"]
    1: "pelem"
    2: "anggur"
    length: 3
  ▶ __proto__: Array(0)
```

Contoh kasus kedua tsb. kita bisa memvisualisasikan sebagai berikut:



Sehingga dari kasus kedua ini panjang array dari buah2[] hanya menjadi 3 saja. Hal tersebut dengan kasus pertama yang panjang array buah2[] nya adalah 5 karena apel, jeruk, dan melon masing-masing dianggap sebagai data-data yang terpisah dan bukan dalam satu-kesatuan sebuah data sehingga mereka berada pada satu tingkatan level data yang sama dengan pelem dan anggur dalam array buah2[].

Sedangkan pada kasus kedua apel, jeruk, dan melon dianggap satu paket data yang menjadi satu sehingga mereka dianggap pada tingkatan level yang berbeda dengan pelem dan anggur, yaitu menjadi data sub-array pada data buah[] atau menjadi data array yang ada di dalam buah2[0].

Kemudian dengan menggunakan spread operator tsb. kita juga dapat mengisi parameter sebuah fungsi dengan array memasukkan spread operator array yang diinginkan pada parameter sebuah pemanggilan fungsi seperti pada contoh skrip kode berikut.

```
107     var mahasiswa = ['Jono', 2019, 'RPL'];
108
109     mhs = (nama, angkatan, prodi)=>{
110         console.log(`${nama} merupakan mahasiswa ${prodi} dari angkatan ${angkatan}`);
111     }
112
113     mhs(...mahasiswa);
```

Jono merupakan mahasiswa RPL dari angkatan 2019

Awal.html:110

>