MODUL 6 JAVASCRIPT LANJUTAN

6.1 Deskripsi Singkat

Selain HTML dan CSS, JavaScript adalah salah satu dari 3 bahasa yang harus dipelajari semua pengembang web. HTML digunakan untuk mendefinisikan konten halaman web. CSS digunakan untuk mengatur tampilan dan menentukan tata letak konten pada halaman web. Adapun Javascript digunakan untuk memprogram perilaku halaman web. Pada praktikum kali ini, anda akan mencoba menggunakan Javascript untuk membuat konten yang dinamis pada suatu dokumen HTML.

6.2 Tujuan Praktikum

Setelah menyelesaikan praktikum pada modul ini, mahasiswa diharapkan dapat membuat beberapa dokumen HTML dan menambahkan JavaScript untuk membuat konten yang dinamis.

6.3 Material Praktikum

Praktikum ini memerlukan aplikasi text editor seperti Notepad (bawaan jika anda menggunakan Microsoft Windows). Anda juga bisa mengunduh aplikasi text editor lainnya seperti Notepad++ atau Visual Studio Code. Jangan menggunakan rich-text editor seperti Microsoft Word atau sejenisnya. Selain itu, praktikum ini juga memerlukan web browser seperti Internet Explorer atau Google Chrome. Saat ini, anda belum memerlukan web server untuk membuat halaman web.

6.4 Kegiatan Praktikum

6.4.1 Fungsi

1. Mari kita mulai dengan latihan tentang fungsi dalam JavaScript.
   a. Buka text editor lalu masukkan markup HTML dan kode JavaScript berikut ini.

```
<!DOCTYPE html>
<html lang="en-GB">
<head>
  <title>JavaScript 08A: Functions</title>
  <script id="s1">
    function loop(max) {
      function applyAndWrite(f, y) {
        document.writeln(f.name, '(', y, ') = ', f(y),
"<br>")
      }
      function times2(x) { return 2 * x }
      // Code for Exercise 1e here
      for (var i = 0; i <= max; i++) {
```

```
        applyAndWrite(times2, i)
      }
    }
  </script>
</head>
<body>
  <script id="s2">
    loop(2)
     // Code for Exercise 1d here
  </script>
  <noscript>JavaScript not supported or
enabled</noscript>
</body>
</html>
```

b. Simpan markup HTML dan kode JavaScript di atas dalam file bernama **jsDemo08A.html**.

c. Buka file jsDemo08A.html dengan web browser. Jika semua berjalan dengan benar, anda akan melihat output seperti ini:

```
times2(0) = 0
times2(1) = 2
times2(2) = 4
```

Jika berbeda, buka jendela console, lalu lacak kesalahannya dan betulkan.

d. Perhatikan bahwa fungsi applyAndWrite dan times2 didefinisikan di dalam (nested) fungsi loop. Bagaimana jika anda mencoba untuk memanggil fungsi times2 di luar fungsi loop? Tambahkan kode berikut.

```
    times2(6);
```

di akhir elemen script dengan id s2.

Simpan file, lalu muat ulang/buka kembali di web browser. Apa yang terjadi? Hapus kembali perintah panggil fungsi times2(6) dari kode progran.

e. Berikutnya, tambahkan kode berikut tepat di bawah definisi fungsi2, di tempat yang telah ditandai dalam komentar.

```
    function scope(x) { return i * x; }
```

Lalu, di dalam for-loop, tambahkan perintah panggil fungsi applyAndWrite(scope, i). Simpan file, kemudian muat ulang ke dalam web browser. Jika semua berjalan dengan benar, output akan seperti ini..

```
times2(0) = 0
scope(0) = 0
times2(1) = 2
scope(1) = 1
times2(2) = 4
scope(2) = 4
```

Mengapa outputnya seperti ini? Variabel i yang mana fungsi scope merujuk dalam kode programnya?

f. Kembangkan kode program ini dengan menambahkan definisi fungsi power2, yang mengambil satu argumen, yaitu x, dan mengembalikan hasil perhitungan x2. Kembangkan juga kode program dalam blok for-loop, dengan menambahkan perintah untuk memanggil fungsi power2, seperti halnya untuk fungsi times2 dan scope. Simpan dan amati hasilnya.

2. Mari kita lanjutkan latihan kita tentang fungsi dalam JavaScript.

   a. Buka text editor lalu masukkan markup HTML dan kode JavaScript berikut ini.

```
<!DOCTYPE html>
<html lang="en-GB">

<head>
  <title>JavaScript 08B</title>
  <script>
    // Function sumAll has no minimum number of arguments,
    // but accepts an arbitrary number of arguments
    function sumAll() {
      sum = 0
      for (var i = 0; i < arguments.length; i++)
        sum = sum + arguments[i]
      return sum
    }
  </script>
</head>

<body>
  <script>
    document.writeln("sumAll() = " + sumAll() + "<br>")
    document.writeln("sumAll(2) = " + sumAll(2) + "<br>")
    document.writeln("sumAll(2,3) = " + sumAll(2, 3) +
"<br>")
    document.writeln("sumAll(2,3,4) = " + sumAll(2, 3, 4)
+ "<br>")
  </script>
  <noscript>JavaScript not supported or
enabled</noscript>
</body>

</html>
```

   Simpan dalam file **jsDemo08B.html**.

b. Buka file jsDemo08B.html dengan web browser. Jika semua berjalan dengan benar, anda akan melihat output sepert ini.

```
sumAll() = 0
sumAll(2) = 2
sumAll(2,3) = 5
sumAll(2,3,4) = 9
```

Jika berbeda, telusuri kode programnya dengan debugger/console dan betulkan kesalahannya.

c. Dalam definisi fungsi sumAll, kita menggunakan variabel sum untuk menyimpan hasil penjumlahan. Apakah variabel sum merupakan variabel lokal yang cakupannya hanya dalam blok fungsi; atau global?

Mari kita coba periksa dengan menambahkan kode berikut sebelum tag penutup elemen script </script>.

```
    document.writeln("sum = " + sum + "<br>");
```

Simpan file dan muat kembali ke dalam web browser. Dari outputnya, kesimpulan apa yang anda dapatkan?

d. Jika kesimpulan anda bahwa variabel sum adalah global, coba ubah kode program definisi dari fungsi sumAll agar variabel sum menjadi lokal. Simpan perubahannya, lalu tes bagaimana hasilnya.

3. Mari kita perdalam tentang fungsi yang lebih komples dalam JavaScript. Kita akan mengulas kembali fungsi bubble_sort yang telah dibahas pada pertemuan kelas.

a. Buka text editor lalu masukkan markup HTML dan kode JavaScript berikut ini.

```
<!DOCTYPE html>
<html lang="en-GB">
<head>
  <title>JavaScript 08C</title>
  <style>
    table {
      border: 1px;
      padding: 5px;
    }
  </style>
  <script>
    function bubble_sort(array) {
      function swap(array, i, j) {
        // swap can change array because array is
        // a local variable of the outer function bubble_sort
        var tmp = array[i];
        array[i] = array[j];
        array[j] = tmp;
      }
```

```
      if (!(array && array.constructor == Array))
        throw ("Argument not an array");
      for (var i = 0; i < array.length; i++)
        for (var j = 0; j < array.length - i; j++)
          if (array[j + 1] < array[j]) swap(array, j, j + 1)
      return array;
    }
  </script>
</head>
<body>
  <h1>Sorting Arrays</h1>
  <script>
    numbers = [20, 4, 3, 9, 6, 8, 5, 10];
    document.writeln("<h2>bubble_sort function</h2>");
    document.writeln("<table>\n<tbody>");
    document.writeln("<tr><td>array before sorting" +
      "</td><td>" + numbers.join(", ") + "</td></tr>");
    // Sort a copy of array
    sorted = bubble_sort(numbers.slice());
    document.writeln("<tr><td>array after sorting of copy" +
      "</td><td>" + numbers.join(", ") + "</td></tr>");
    sorted = bubble_sort(numbers);
    document.writeln("<tr><td>array after sorting of itself"
+
      "</td><td>" + numbers.join(", ") + "<\/td></tr>");
    document.writeln("<tr><td>sorted array</td><td>" +
      sorted.join(", ") + "</td></tr>");
    document.writeln("</tbody>\n</table>");
  </script>
</body>
</html>
```

b. Simpan markup HTML dan kode JavaScript di atas dalam file bernama **jsDemo08C.html**.

c. Buka file jsDemo08C.html dengan web browser, lalu bandingkan hasilnya dengan yang ada pada bahan paparan materi kelas. Pastikan anda paham perbedaan antara penggunaan fungsi bubble_sort(numbers) dengan bubble_sort(numbers.slice()).

d. Kita telah pelajari tentang property prototype untuk mengubah method/fungsi yang diasosiasikan dengan suatu objek. Fitur ini tidak hanya untuk user-defined objects, namun juga untuk objek yang disediakan oleh JavaScript, misalnya Array. Kita akan mengeksplor lebih lanjut melalui fungsi bubble_sort ini.

   Sebagai persiapan, tambahkan kode JavaScript berikut di dalam elemen-head di dalam file jsDemo08C.html.

```
function bubble_sort2() {
  return this;
}
```

Kemudian, tambahkan kode JavaScript berikut di akhir skrip JavaScript yang ada di dalam elemen-body di dalam file jsDemo08C.html.

```
    Array.prototype.bubble_sort = bubble_sort2;
    numbers = [20, 4, 3, 9, 6, 8, 5, 10];
    document.writeln("<h2>bubble  sort  array  prototype
method</h2>");
    document.writeln("<table>\n<tbody>");
    document.writeln("<tr><td>array  before sorting" +
      "</td><td>" + numbers.join(", ") + "</td></tr>");
    // Sort a copy of array
    sorted = numbers.slice().bubble_sort();
   document.writeln("<tr><td>array  after  sorting of copy"
+
      "</td><td>" + numbers.join(", ") + "</td></tr>");
    sorted = numbers.bubble_sort();
    document.writeln("<tr><td>array   after   sorting  of
itself" +
      "</td><td>" + numbers.join(", ") + "</td></tr>");
    document.writeln("<tr><td>sorted array</td><td>" +
      sorted.join(", ") + "</td></tr>");
    document.writeln("</tbody>\n</table>");
```

Simpan file. Bersihkan output jendela console jika ada, lalu muat ulang/buka kembali di web browser.

Karena skrip tambahan menuliskan numbers.bubble_sort(), saat ini, belum menghasilkan deret angka yang berurutan. Pastikan anda paham apa yang skrip berikut kerjakan:

```
Array.prototype.bubble_sort = bubble_sort2;
```

Kembangkan definisi fungsi bubble_sort2 dengan menggunakan kembali kode dari fungsi bubble_sort, sehingga untuk tiap array a0 yang diberikan, a0.bubble_sort() akan mengembalikan array yang telah diurutkan.

Kisi-kisi: Ambil kode dari fungsi bubble_sort lalu ganti bagian yang ditulis sebagai array dengan this.

e. Bubble sort adalah algoritma yang tidak efisien untuk mengurutkan suatu deret, dan siapa pun tahu itu. Ditambah, JavaScript juga sudah menyediakan pre-defined sort method untuk mengurutkan array menggunakan algoritma yang lebih efisien.

   Tambahkan kode program berikut di akhir skrip JavaScript di dalam elemen-body file jsDemo08C.html.

```
    numbers = [20, 4, 3, 9, 6, 8, 5, 10];
    document.writeln("<h2>pre-defined      array      sort
method</h2>");
    document.writeln("<table>\n<tbody>");
    document.writeln("<tr><td>array  before sorting" +
      "</td><td>" + numbers.join(", ") + "</td></tr>");
    // Sort a copy of array
    sorted = numbers.slice().sort();
    document.writeln("<tr><td>array   after   sorting  of
copy" +
      "</td><td>" + numbers.join(", ") + "</td></tr>");
    sorted = numbers.sort();
    document.writeln("<tr><td>array   after   sorting  of
itself" +
      "</td><td>" + numbers.join(", ") + "</td></tr>");
    document.writeln("<tr><td>sorted array</td><td>" +
      sorted.join(", ") + "</td></tr>");
    document.writeln("</tbody>\n</table>");
```

Simpan file. Bersihkan output jendela console jika ada, lalu muat ulang/buka kembali di web browser.

f. Dari hasil di atas, anda akan melihat hasil dari pre-defined sort method berbeda dengan hasil fungsi bubble_sort yang kita buat. Cari tahu dari sumber di web mengapa begitu.

g. Sekali lagi, dengan bantuan referensi dari web, cari tahu bagaimana caranya agar kode pada nomor 1e memberikan hasil yang diharapkan, yaitu urutan secara numerik. Pastikan anda mendokumentasikan sumber dari solusi yang anda ambil dalam komentar kode program.

h. Kembangkan objek Array dengan menambahkan method sum: array.sum() yang mengembalikan hasil penjumlahan dari seluruh elemen yang ada dalam array. Jika jumlah elemen nol, method sum() akan mengembalikan 0.

```
    document.writeln("<h2>array       prototype       sum
function</h2>");
    // For array0 the expected result is 0
    array0 = [];
    document.writeln("<div>[" + array0 + "].sum() = " +
array0.sum() + "</div>");
    // For array1  the expected result is 6
    array1 = [3, 2, 1];
    document.writeln("<div>[" + array1 + "].sum() = " +
array1.sum() + "</div>");
    // For array2  the expected result is 6
    array2 = [3, "2M", [1, 0]];
    document.writeln("<div>[" + array2 + "].sum() = " +
array2.sum() + "</div>");
```

i. Kembangkan lagi objek Array dengan menambahkan method peek: array.peek() yang mengembalikan elemen pertama dalam array tanpa mengubah array. Jika jumlah elemen nol, method peek() akan mengembalikan undefined.

6.4.2Array

Berikutnya kita akan mempelajari array dan operasinya.

1. Buka text editor lalu buat file HTML jsDemo08D.html yang berisi kode JavaScript berikut ini.

```
    planets = ["earth"];
    planets.unshift("mercury", "venus");
    planets.push("mars", "jupiter", "saturn");
    document.writeln("planets\@1:  "  +  planets.join("  ")  +
"<br>");
    last = planets.pop();
    document.writeln("planets\@2:  "  +  planets.join("  ")  +
"<br>");
    first = planets.shift();
    document.writeln("planets\@3:  "  +  planets.join("  ")  +
"<br>");
    document.writeln(" \@4: " + first + " " + last + "<br>");
    home = ["mercury", "venus", "earth"].pop();
    document.writeln(" \@5: " + home + "<br>");
    number = ["earth"].push("mars");
    document.writeln(" \@6: " + number + "<br>");
```

   Jangan lupa tambahkan markup HTML yang diperlukan.

2. Buka file **jsDemo08D.html** dengan web browser. Pastikan hasilnya sesuai dengan yang diharapkan dari kode tersebut.
3. Kembangkan skrip dengan menambahkan kode berikut:

```
    planets.length = 2;
    document.writeln("planets\@7:  "  +  planets.join("  ")  +
"<br>");
    list = planets; // Modify this line
    document.writeln("list \@8: " + list.join(" ") + "<br>");
    planets[1] = "midgard";
    document.writeln("list \@9: " + list.join(" ") + "<br>");
```

4. Simpan, lalu buka/muat kembali di web browser. Baris terakhir dari output akan seperti ini:

```
list @9: venus midgard
```

   Bagaimana bisa seperti itu?

5. Kembangkan lagi skrip dengan menambahkan kode berikut:

```
    planets = ["Bellerophon", "Ymir", "Osiris"];
    document.writeln("list \@10: " + list.join(" ") + "<br>");
    document.writeln("planets\@10:  "  +  planets.join("  ")  +
"<br>");
```

6. Simpan, lalu buka/muat kembali di web browser. Dua baris terakhir dari output akan seperti ini:

```
list @10: venus midgard
planets@10: Bellerophon Ymir Osiris
```

Bagaimana bisa seperti itu?

7. Ubah kode yang ditambahkan pada no.3 di tempat yang ditunjukkan dengan '// Modify this line', agar perubahan pada variabel planets tidak berefek pada variabel list.

6.4.3 Objek dan Inheritence

1. Pada latihan ini, kita akan melihat bagaimana JavaScript mengimplementasikan objek dan pewarisan/inheritance tanpa konsep class.
   a. Buka text editor lalu masukkan markup HTML dan kode JavaScript berikut ini.

```
<!DOCTYPE html>
<html lang="en-GB">

<head>
  <title>JavaScript 09A: Inheritance</title>
  <script id="j1">
    function Rectangle(width, height) {
      this.width = width;
      this.height = height;
      this.type = 'Rectangle';
    }
    Rectangle.prototype.area =
      function () { return this.width * this.height; };
    function Square(length) {
      this.width = this.height = length;
      this.type = 'Square';
    }
    // Square inherits from Rectangle
    Square.prototype = new Rectangle();
  </script>
</head>

<body>
  <script id="j2">
```

```
    var rc1 = new Rectangle(2, 3);
    var sq1 = new Square(5);
    document.writeln("The area of rc1 is ", rc1.area(),
"<br>");
    document.writeln("The area of sq1 is ", sq1.area(),
"<br>");
    document.writeln("The area of ", rc1, " is ",
rc1.area(), "<br>");
    document.writeln("The area of ", sq1, " is ",
sq1.area(), "<br>");
  </script>
</body>

</html>
```

b. Simpan dalam file **jsDemo09A.html**.

c. Buka file jsDemo09A.html dalam web browser. Anda akan lihat hasilnya seperti ini:

```
The area of rc1 is 6
The area of sq1 is 25
The area of [object Object] is 6
The area of [object Object] is 25
```

Perhatikan bahwa ketika anda menampilkan suatu objek (misalnya dengan fungsi document.writeln atau console.log), JavaScript akan menggunakan method toString dari objek tersebut untuk mendapatkan representasi string dari objek tersebut. Jika method toString diwariskan dari Object tanpa diubah, maka representasi string adalah "[object *type*]", di mana *type* adalah tipe dari objek tersebut.

d. Ubah representasi dari objek Rectangle dengan menampilkan lebar/width *wd* dan tinggi/height *ht* menjadi "Rectangle[*wd*, *ht*]". Perhatikan bahwa property .type bisa digunakan untuk mendapatkan tipe dari objek, yaitu dalam contoh objek Rectangle adalah 'Rectangle'. Jika benar kodenya, hasil dari dua baris terakhir akan menjadi seperti ini:

```
The area of Rectangle[2, 3] is 6
The area of Rectangle[5, 5] is 25
```

e. Untuk baris terakhir, di mana seharusnya merepresentasikan objek Square, yaitu "Square[*wd*]" (cukup lebar/width saja karena bujur sangkar), perlu diubah lagi agar menjadi seperti itu.

Ubah method toString dari objek Square agar mendapatkan hasil yang diharapkan. Jika berhasil, akan menampilkan seperti ini:

```
The area of Square[5] is 25
```

f. Dengan meniru yang telah diimplementasikan pada Rectangle dan Square, tambahkan definisi objek Polygon dengan menambahkan kode JavaScript pada elemen denga id j1 pada file jsDemo09A.html. Constructor objek Polygon menerima dua parameter bilangan yang menentukan jumlah sisi *n* dan panjang tiap sisi *s*. Representasi string dari objek Polygon ini adalah "*n*-Polygon[*s*]". Luas area dari polygon dihitung dengan rumus seperti yang diberikan pada referensi [2], dibulatkan sampai dua digit di belakang koma.

   Kisi-kisi: Rumus di atas menggunakan fungsi tangen dengan argumen sudut dalam satuan derajat, sedangkan fungsi tangen dalam JavaScript menggunakan satuan radian. Lihat referensi [1] untuk konversi antar satuan sudut.

   Tes solusi anda dengan memberikan kasus poligon dengan 5 sisi dan panjang tiap sisi 6. Periksa hasilnya apakah seperti ini:

```
The area of 5-Polygon[6] is 61.94
```

2. Di kelas kita sudah melihat beberapa aspek pada objek dalam JavaScript, khususnya bagaimana variabel instance dan variabel 'class' dideklarasi dan bagaimana dengan aspek public dan privat. Dalam latihan ini kita akan melihat lebih mendalam.

   a. Buka text editor lalu masukkan markup HTML dan kode JavaScript berikut ini.

```
<!DOCTYPE HTML>
<html lang="en-GB">

<head>
  <title>JavaScript 09B: Objects</title>
  <script>
  </script>
</head>

<body>
  <script>
    var e = [];
    e[0] = new Employee("Hal Smith", 30000);
    e[1] = new Employee("Tim Peck", 20000);
    e[2] = new Employee("Ari Bell", 18000);
    // For e[0].name we expect 'Hal Smith'
    document.writeln("e[0].name = "+e[0].name+"<br>");
    // For e[0].salary we expect 'undefined'
    document.writeln("e[0].salary = "+e[0].salary+"<br>");
    // For e[0].getSalary() we expect 30000
    document.writeln("e[0]'s           salary           =
"+e[0].getSalary()+"<br>");
```

```
    // For e[1].getName() we expect 'Tim Peck'
    document.writeln("e[1]'s             name             =
"+e[1].getName()+"<br>");
    // For e[1].getSalary() we expect 20000
    document.writeln("e[1]'s           salary             =
"+e[1].getSalary()+"<br>");
    // We make changes to e[1]
    document.writeln("Changing e[1]'s name to 'Tom Beck'
and " +
                     "salary to 25000<br>");
    e[1].name = "Tom Beck";
    e[1].setSalary(25000);
    // For e[1].getName() we now expect 'Tom Beck'
    document.writeln("e[1]'s             name             =
"+e[1].getName()+"<br>");
    // For e[1].getSalary() we now expect 25000
    document.writeln("e[1]'s           salary             =
"+e[1].getSalary()+"<br>");
    // For e[1].getEmployeeCount() we expect 3
    document.writeln("Employees:
"+e[1].getEmployeeCount()+"<br>");
  </script>
</body>

</html>
```

b. Simpan dalam file **jsDemo09B.html**.

c. Di sini, kita akan mendefinisikan objek employee dengan nama Employee. Agar tidak terlaku kompleks, atribut yang dimiliki oleh employee hanya name dan salary. Yang pertama bersifat publik, sedangkan yang kedua bersifat privat. Kita juga perlu satu method untuk mendapatkan informasi tentang salary employee, yang memungkinkan juga untuk diubah. Terakhir, kita juga perlu mengetahui berapa jumlah employee yang ada, dan jumlahnya perlu kita jaga secara privat. Jumlah employee didapat secara otomatis tiap kali ada employee baru.

   Buat constructor untuk objek Employee yang memenuhi kebutuhan di atas. Tambahkan kode JavaScript yang diperlukan di dalam elemen-script yang ada dalam elemen-head.

d. Setelah selesai, tes solusi anda dengan membuka file jsDemo09B.html di web browser. Amati hasilnya apakah sesuai dengan yang diharapkan dari kode JavaScript yang ada.

e. Karena jumlah employee sebetulnya bukan atribut dari employee tertentu, maka agak kurang pas kalau kita menggunakan ekspresi seperti e[1].employeeCount() untuk mendapatkan jumlah employee.

Apakah memungkinkan jika kita mengakses jumlah employee dengan misalnya mendefinisikan method employeeCount() dengan cara lain sehingga bisa diakses dengan Employee.employeeCount()? Jika memungkinkan, ubah kode anda, lalu tes dengan mengganti kode berikut:

```
document.writeln("Employees:
"+e[1].getEmployeeCount()+"<br>");
```

dengan

```
document.writeln("Employees:
"+Employee.getEmployeeCount()+"<br>");
```

dan periksa bahwa anda mendapatkan hasil yang diharapkan.

f. Kita sudah bisa 'membuat' objek employee. Itu saja sudah hebat. Namun akan lebih lengkap kalau kita juga bisa 'menghapus' employee yang ada.

Bisakah kita menambahkan method remove pada objek Employee yang berfungsi untuk menghapus employee tertentu, lalu sekaligus mengurangi jumlah employee yang ada? Jika bisa, ubah kode anda, lalu tambahkan kode berikut untuk mengetes solusi yang anda tambahkan.

```
e[2].remove();
// Expected to print the number 2 now
document.writeln("Employees:
“+Employee.getEmployeeCount()+"<br>");
```

Periksa apakah kode yang anda tambahkan berfungsi sesuai harapan.

Kisi-kisi: Ada diskusi di StackOverflow https://stackoverflow.com/a/15089643 .

g. Solusi yang paling optimal adalah dengan membuat objek Employees (dengan 's') yang menyimpan seluruh objek Employee, yang kemudian memungkinkan untuk menghapus employee dengan memberikan name. Namun itu akan menjadi sangat kompleks dan di luar dari cakupan latihan ini. Bagi yang tertarik, bisa dicoba untuk mengimplementasikan.

6.5 Penugasan

Kerjakan sesuai dengan yang dijelaskan pada bagian Kegiatan Praktikum. Hasil pekerjaan praktikum dikumpulkan dalam format zip dengan format nama <<nim>>_modul6, contoh: 192191234_modul6.