MODUL 6. JAVASCRIPT & JQUERY

Tujuan Praktikum
1. Mahasiswa mampu memahami konsep dan implementasi Javascript pada *web.* 2. Mahasiswa mampu memahami sintaks, elemen, dan fungsi pada Javascript. 3. Mahasiswa mampu memahami konsep dan implementasi jQuery pada *web.* 4. Mahasiswa mampu memahami sintaks jQuery.

6.1. Pengenalan Javascript

6.1.1. Sejarah Singkat Javascript

Javascript, seperti namanya, merupakan bahasa pemrograman *scripting*. Dan seperti bahasa *scripting* lainnya, Javascript umumnya digunakan hanya untuk program yang tidak terlalu besar, biasanya hanya beberapa ratus baris. Javascript pada umumnya mengontrol program yang berbasis Java. Jadi memang pada dasarnya Javascript tidak dirancang untuk digunakan dalam aplikasi skala besar.

Meskipun dibuat dengan tujuan awal untuk mengendalikan program Java, komunitas Javascript menggunakan bahasa ini untuk tujuan lain, memanipulasi gambar dan isi dari dokumen HTML. Singkatnya, pada akhirnya Javascript digunakan untuk satu tujuan utama, "menghidupkan" dokumen HTML dengan mengubah konten statis menjadi dinamis dan interaktif. Bersamaan dengan perkembangan Internet dan dunia *web* yang pesat, Javascript akhirnya menjadi bahasa utama dan satu-satunya untuk membuat HTML menjadi interaktif di dalam browser.

6.1.2. Prinsip Dasar Javascript

Prinsip dasar yang terdapat pada bahasa pemrograman javascript adalah sebagai berikut.

1. Javascript mendukung paradigma pemrograman imparatif (Javascript dapat menjalankan perintah program baris demi baris, dengan masing-masing baris berisi satu atau lebih perintah), fungsional (struktur dan elemen-elemen dalam program sebagai fungsi matematis yang tidak memiliki keadaan (*state*) dan data yang dapat berubah (*mutable data*)), dan orientasi objek (segala sesuatu yang terlibat dalam program dapat disebut sebagai "objek").
2. Javascript memiliki model pemrograman fungsional yang sangat ekspresif.
3. Pemrograman berorientasi objek (PBO) pada Javascript memiliki perbedaan dari PBO pada umumnya.
4. Program kompleks pada Javascript umumnya dipandang sebagai program-program kecil yang saling berinteraksi.

6.2 Sintaks Umum pada Javascript

6.2.1. Tipe data dasar

Seperti kebanyakan bahasa pemrograman lainnya, Javascript memiliki beberapa tipe data untuk dimanipulasi. Seluruh nilai yang ada dalam Javascript selalu memiliki tipe data. Tipe data yang dimiliki oleh Javascript adalah sebagai berikut:

- *Number* (bilangan)
- *String* (serangkaian karakter)
- *Boolean* (benar / salah)

- *Object*
- Function (fungsi)
- Array
- Date
- *RegExp* (*regular expression*)
- Null (tidak berlaku / kosong)
- Undefined (tidak didefinisikan)

Kebanyakan dari tipe data yang disebutkan di atas sama seperti tipe data sejenis pada bahasa pemrograman lainnya. Misalnya, sebuah boolean terdiri dari dua nilai saja, yaitu true dan false.

6.2.2. Variabel

Seperti pada bahasa pemrograman lainnya, variabel dalam Javascript merupakan sebuah tempat untuk menyimpan data sementara. Variabel dibuat dengan kata kunci **`var`** pada Javascript.

```
// Deklarasi berbagai tipe data dasar
var angka = 42;               // Number
var teks = "Ahmad Ruslan";    // String
var benar = true;             // Boolean
var salah = false;            // Boolean
var kosong = null;            // Null
var tidakDidefinisikan;       // Undefined
var daftar = ["apel", "pisang", "mangga"]; // Array
var tanggal = new Date();      // Date (objek tanggal)
var pola = /[A-Z]/;    // RegExp (regular expression)
var orang = { nama: "Ruslan", umur: 21 }; // Object

// Gabungkan hasil ke satu variabel string
var hasil = "";
hasil += "<b>Number:</b> " + angka + "<br>";
hasil += "<b>String:</b> " + teks + "<br>";
hasil += "<b>Boolean 1:</b> " + benar + "<br>";
hasil += "<b>Boolean 0:</b> " + salah + "<br>";
hasil += "<b>Null:</b> " + kosong + "<br>";
hasil += "<b>Undefined:</b> " + tidakDidefinisikan + "<br>";
hasil += "<b>Array:</b> " + daftar.join(", ") + "<br>";
hasil += "<b>Date:</b> " + tanggal + "<br>";
hasil += "<b>RegExp:</b> " + pola + ", Ini Sirsak = " + pola.test("Sirsak") +
"<br>";
hasil += "<b>Object:</b> nama = " + orang.nama + ", umur = " + orang.umur +
"<br>";

// Tampilkan hasil di halaman
document.getElementById("outputTipeData").innerHTML = hasil;
```

Atau tampilkan dengan cara lainnya seperti:

```
var nama = "Dewi";
var umur = 21;
var isMahasiswa = true;

document.getElementById("test").innerHTML =
            "Nama: " + nama + "<br>" +
            "Umur: " + umur + "<br>" +
            "Mahasiswa: " + isMahasiswa;
```

Nilai yang ada di dalam variabel dapat diganti dengan mengisikan nilai baru, dan bahkan dapat diganti tipe datanya juga.

```
nama = "Anton"; // nama berisi string "Anton"
nama = 1;       // nama sekarang berisi integer 1
```

Walaupun kemampuan untuk menggantikan tipe data ini sangat memudahkan kita dalam mengembangkan aplikasi, fitur ini harus digunakan dengan sangat hati-hati. Perubahan tipe data yang tidak diperkirakan dengan baik dapat menyebabkan berbagai kesalahan (*error*) pada program, misalnya jika kita mencoba mengakses method **charAt()** (fungsi yang mengembalikan nilai *char* pada indeks sebuah *string*) setelah mengubah tipe pada contoh di atas.

6.2.3. Array

Array merupakan sebuah tipe data yang digunakan untuk menampung banyak tipe data lainnya. Berbeda dengan tipe data *object*, *array* pada Javascript merupakan sebuah tipe khusus. Walaupun memiliki *method* dan properti, *array* bukanlah objek, melainkan sebuah tipe yang "mirip objek". Pembuatan *array* dalam Javascript dilakukan dengan menggunakan kurung siku ([]):

```
var data = ["satu", 2, true];
```

Elemen *array* pada Javascript tidak harus memiliki tipe data yang sama seperti contoh diatas. Selain itu Javascript juga mendukung untuk membuat *array* di dalam array yang biasa dikenal dengan *array* dua dimensi seperti contoh dibawah ini.

```
var arr2 = [["satu", "dua"], ["tiga", "empat"]];
```

Pengaksesan elemen dalam *array* dilakukan dengan menggunakan kurung siku. Nilai yang kita berikan dalam kurung siku adalah urutan elemen penulisan *array* (indeks), yang dimulai dari nilai 0. Jika indeks yang diakses tidak ada, maka kita akan mendapatkan nilai *undefined*.

```
data[2];     // mengembalikan true
arr2[0][1]; // mengembalikan "dua"
data[10];    // mengembalikan undefined
```

Copy code ini:

```
var data = ["Satu", 2, true, "Tiga", 4]; // Array dengan tipe data campuran
var arr2 = [["satu", "dua"], ["tiga", "empat"]]; // Array 2D

document.getElementById("array").innerHTML =
      "Elemen pertama: " + data[0] + "<br>" +
      "Elemen ketiga: " + data[2] + "<br>" +
      "Isi arr2: " + arr2 + "<br>" +
      "Elemen arr2[0][1]: " + arr2[0][1] + "<br>" + //elemen pada baris 0, kolom 1
      data[10];     // mengembalikan undefined
```

Sebagai sebuah objek khusus, *array* juga memiliki *method* dan properti. Beberapa *method* dan properti yang populer misalnya *length*, pop(), dan push().

Copy code ini:

```
var data2 = ["pisang", "apel", "mangga"];

        // Menampilkan hasil ke HTML
        document.getElementById("array2").innerHTML =
            "Isi awal array: " + data2 + "<br>" +
```

```
            "Panjang awal array: " + data2.length + "<br><br>" +

            // Menambahkan elemen baru
            "Menambahkan durian (push): " + data2.push("durian") + "<br>" +
//mengembalikan panjang baru array
            "Isi array setelah push: " + data2 + "<br>" +

            // Menghapus elemen terakhir
            "Menghapus elemen terakhir (pop): " + data2.pop() + "<br>" + // mengembalikan
elemen yang dihapus
            "Isi array setelah pop: " + data2 + "<br>";
```

6.2.4. Pengendalian Struktur

Javascript memiliki perintah-perintah pengendalian struktur (*control stucture*) yang sama dengan bahasa dalam keluarga C. Perintah **`if`** dan **`else`** digunakan untuk percabangan, sementara perintah **`for`**, **`for-in`**, **`while`**, dan **`do-while`** digunakan untuk perulangan.

Percabangan pada Javascript bisa dikatakan sama persis dengan C atau Java:

```
var pendidikan = "S2";
        var gelar;

        if (pendidikan === "S1") {
            gelar = "Sarjana";
        } else if (pendidikan === "S2") {
            gelar = "Master";
        } else if (pendidikan === "S3") {
            gelar = "Doktor";
        } else {
            gelar = "Tidak diketahui";
        }

        document.getElementById("outputPercabangan").innerHTML =
            "Sekarang kamu: " + pendidikan + ", gelar kamu: " + gelar;
```

Satu hal yang perlu diperhatikan, tiga buah sama dengan (===) digunakan pada operasi perbandingan di Javascript. Javascript mendukung dua operator perbandingan sama dengan, yaitu == dan ===. Perbedaan utamanya adalah == mengubah tipe data yang dicek menjadi nilai terdekat, sementara === memastikan tipe data dari dua nilai yang dibandingkan sama. Untuk mendapatkan nilai perbandingan paling akurat, selalu gunakan === ketika mengecek nilai.

Sama seperti **`if`**, perulangan **`do`**, **`do-while`**, dan **`for`** memiliki cara pemakaian yang dapat dikatakan sama persis dengan C atau Java:

```
var text1 = "";
        var i = 0;
        while (i < 3) {
            text1 += "Perulangan while ke-" + i + "<br>"; // Tambah teks dengan nomor
perulangan
            i++; // Increment i by 1
        }
        document.getElementById("whileLoop").innerHTML = text1;
```

6.3. Object Orientation pada Javascript

Javascript memiliki dua jenis tipe data utama, yaitu tipe data dasar dan objek. Tipe data dasar pada Javascript adalah angka (*numbers*), rentetan karakter (*strings*), boolean (*true* dan *false*), *null*, dan *undefined*. Nilai-nilai selain tipe data dasar secara otomatis dianggap sebagai objek. Objek dalam Javascript didefinisikan sebagai *mutable properties collection*, yang artinya adalah sekumpulan properti (ciri khas) yang dapat berubah nilainya. Karena nilai-nilai selain tipe data dasar merupakan objek, maka pada Javascript sebuah *Array* adalah objek. Fungsi adalah objek dan *Regular expression* juga merupakan objek.

6.3.1 Pembuatan Object pada Javascript

Notasi pembuatan objek pada Javascript sangat sederhana, yaitu sepasang kurung kurawal yang membungkus properti. Notasi pembuatan objek ini dikenal dengan nama *object literal*. *Object literal* dapat digunakan kapanpun pada ekspresi Javascript yang valid:

Format deklarasi objek literal:

```
var namaObject = {
  key1: value1,
  key2: value2,
  key3: value3
};
```

Contoh penggunaan:

```
 var objek_kosong = {};
 var mobil = {
      "warna-badan": "merah", // properti dengan nama
mengandung tanda hubung
      "nomor-polisi": "BK1234AB",
      merk: "Toyota", // properti dengan nama biasa
      tahun: 2020
      };
```

Nama properti dari sebuah objek harus berupa *string*, dan boleh berisi *string* kosong (""). Jika merupakan nama Javascript yang legal, kita tidak memerlukan petik ganda pada nama properti. Petik ganda seperti pada contoh ("warna-badan") hanya diperlukan untuk nama Javascript ilegal atau kata kunci seperti "`if`" atau "`var`". Misalnya, "nomor-polisi" memerlukan tanda petik, sementara nomor_polisi tidak. Contoh lain, variasi tidak memerlukan tanda petik, sementara "`var`" perlu.

Sebuah objek dapat menyimpan banyak properti, dan setiap properti dipisahkan dengan tanda koma (,). Jika ada banyak properti, nilai dari properti pada setiap objek boleh berbeda-beda:

```
var jadwal = {
    platform: 34,
    telah_berangkat: false,
    tujuan: "Medan",
    asal: "Jakarta"
};
```

Karena dapat diisikan dengan nilai apapun (termasuk objek), maka kita dapat membuat objek yang mengandung objek lain (*nested object*; objek bersarang) seperti berikut:

```
var objectUtama = {
  properti1: nilai1,
  properti2: nilai2,
  propertiAnak: {
    subProperti1: nilaiA,
    subProperti2: nilaiB
  }
};
```

```
// Nested Object
        var jadwal = {
            platform: 34,
            telah_berangkat: false,
            asal: {
                kode_kota: "MDN",
                nama_kota: "Medan",
                waktu: "2023-12-29 14:00"
            },
            tujuan: {
                kode_kota: "JKT",
                nama_kota: "Jakarta",
                waktu: "2023-12-29 17:30"
            }
        };
```

6.3.2. Akses Nilai Property

Akses nilai properti dapat dilakukan dengan dua cara, yaitu.

1. Penggunaan kurung siku ([]) setelah nama objek. Kurung siku kemudian diisikan dengan nama properti, yang harus berupa *string*. Cara ini biasanya digunakan untuk nama properti yang adalah nama ilegal atau kata kunci Javascript.
2. Penggunaan tanda titik (.) setelah nama objek diikuti dengan nama properti. Notasi ini merupakan notasi yang umum digunakan pada bahasa pemrograman lainnya. Notasi ini tidak dapat digunakan untuk nama ilegal atau kata kunci Javascript.

Contoh penggunaan kedua cara pemanggilan di atas adalah sebagai berikut:

```
mobil["warna-badan"] // Hasil: "merah"

jadwal.platform     // Hasil: 34
```

Sebagai bahasa dinamis, Javascript tidak akan melemparkan pesan kesalahan jika kita mengakses properti yang tidak ada dalam objek. Kita akan menerima nilai *undefined* jika mengakses properti yang tidak ada:

```
jadwal.nomor_kursi  // Hasil: undefined mobil
["jumlah-roda"]     // Hasil: undefined
```

Code lengkap:

```
var mobil = {
    "warna-badan": "merah",
    "nomor-polisi": "BK1234AB"
    merk: "Toyota", // properti dengan nama biasa
    tahun: 2020
};
```

```
// Nested Object
var jadwal = {
    platform: 34,
    telah_berangkat: false,
    asal: {
        kode_kota: "MDN",
        nama_kota: "Medan",
        waktu: "2023-12-29 14:00"
    },
    tujuan: {
        kode_kota: "JKT",
        nama_kota: "Jakarta",
        waktu: "2023-12-29 17:30"
    }
};

document.getElementById("outputObject").innerHTML =
    "Warna Mobil: " + mobil["warna-badan"] + "<br>" +
    "Nomor Polisi: " + mobil["nomor-polisi"] + "<br>" +
    "Merk: " + mobil.merk + "<br>" +
    "Tahun: " + mobil.tahun + "<br>" +
    "Asal: " + jadwal.asal.nama_kota + "<br>" +
    "Tujuan: " + jadwal.tujuan.nama_kota;
```

Pengaksesan properti pada Javascript juga dapat digunakan secara dinamis untuk mengubah nilai dari properti tersebut. Perubahan nilai properti juga dapat dilakukan untuk properti yang bahkan tidak ada pada objek tersebut:

```
// Tambah properti baru
mobil.jumlahBan = 4;
mobil.bahanBakar = "Bensin";
mobil["warna-badan"] = "biru";
```

6.3.3. Prototype pada Javascript

Pada Javascript yang mengimplementasikan PBO kita tidak lagi perlu menuliskan kelas, dan langsung melakukan penurunan terhadap objek. Misalkan kita memiliki objek mobil yang sederhana seperti berikut:

```
var mobil = { nama: "Mobil",
jumlahBan: 4 };
```

Kita dapat langsung menurunkan objek tersebut dengan menggunakan fungsi `Object.create` seperti berikut:

```
var truk = Object.create(mobil);
// truk.nama === "Mobil"
// truk.jumlahBan === 4
```

Contoh code lengkap:

```
var truk = Object.create(mobil);

        truk["warna-badan"] = "kuning";

        document.getElementById("outputTruk").innerHTML =
            "Warna Truk: " + truk["warna-badan"] + "<br>" +
            "Nomor Polisi Truk: " + truk["nomor-polisi"];
```

6.4. Function pada Javascript

Sebuah fungsi membungkus satu atau banyak perintah. Setiap kali fungsi dipanggil, maka perintahperintah yang ada di dalam fungsi tersebut dijalankan. Secara umum fungsi digunakan untuk penggunaan kembali kode (*code reuse*) dan penyimpanan informasi (*information hiding*). Implementasi fungsi kelas pertama juga memungkinkan penggunaan fungsi sebagai unit-unit yang dapat dikombinasikan, seperti layaknya sebuah lego. Dukungan terhadap pemrograman berorientasi objek juga berarti fungsi dapat digunakan untuk memberikan perilaku tertentu dari sebuah objek.

6.4.1 Pembuatan Fungsi pada Javascript

Sebuah fungsi pada Javascript dibuat dengan cara seperti berikut:

```
function tambah(a, b) {
            return a + b;
        }
```

Cara penulisan fungsi seperti diatas dikenal dengan nama ***function declaration***, atau deklarasi fungsi. Terdapat empat komponen yang membangun fungsi di atas, yaitu:

1. Kata kunci *function*, yang memberitahu Javascript bahwa akan dibuat sebuah fungsi.
2. Nama fungsi, dalam contoh di atas adalah tambah. Dengan memberikan sebuah fungsi nama maka pemanggilan fungsi dapat dirujuk dengan nama tersebut. Harus diingat bawa nama fungsi bersifat opsional, yang berarti **fungsi pada Javascript tidak harus diberi nama**.
3. Daftar parameter fungsi, yaitu a, b pada contoh di atas. Daftar parameter ini selalu dikelilingi oleh tanda kurung (()). Parameter boleh kosong, tetapi tanda kurung wajib tetap dituliskan. Parameter fungsi akan secara otomatis didefinisikan menjadi variabel yang hanya bisa dipakai di dalam fungsi. Variabel pada parameter ini diisi dengan nilai yang dikirimkan kepada fungsi secara otomatis.
4. Sekumpulan perintah yang ada di dalam kurung kurawal ({}). Perintah-perintah ini dikenal dengan nama badan fungsi. Badan fungsi dieksekusi secara berurut ketika fungsi dijalankan.

Penulisan deklarasi fungsi (*function declaration*) seperti di atas merupakan cara penulisan fungsi yang umumnya kita gunakan pada bahasa pemrograman imperatif dan berorientasi objek. Tetapi selain deklarasi fungsi Javascript juga mendukung cara penulisan fungsi lain, yaitu dengan memanfaatkan ekspresi fungsi (*function expression*). Ekspresi fungsi merupakan cara pembuatan fungsi yang memperbolehkan menuliskan fungsi tanpa nama. Fungsi yang dibuat tanpa nama dikenal dengan sebutan fungsi anonim atau fungsi lambda. Berikut adalah cara membuat **fungsi dengan ekspresi** fungsi:

```
var kali = function (a, b) {
            return a * b;
        };
```

Terdapat hanya sedikit perbedaan antara ekspresi fungsi dan deklarasi fungsi:

1. Penamaan fungsi. Pada deklarasi fungsi, nama fungsi langsung diberikan sesuai dengan sintaks yang disediakan Javascript. Penggunaan ekspresi fungsi pada dasarnya menyimpan sebuah fungsi anonim ke dalam variabel dan nama fungsi adalah nama variabel yang dibuat. Perlu diingat juga bahwa pada dasarnya ekspresi fungsi adalah fungsi anonim. Penyimpanan ke dalam variabel hanya diperlukan

karena kita akan memanggil fungsi nantinya.

2. Ekspresi fungsi dapat dipandang sebagai sebuah ekspresi atau perintah standar bagi Javascript, sama seperti penuliskan kode var i = 0. Deklarasi fungsi merupakan konstruksi khusus untuk membuat fungsi. Hal ini berarti pada akhir dari ekspresi fungsi kita harus menambahkan, sementara pada deklarasi fungsi hal tersebut tidak penting.

6.4.2. Pemanggilan Fungsi

Sebuah fungsi dapat dipanggil untuk menjalankan seluruh kode yang ada di dalam fungsi tersebut, sesuai dengan parameter yang kita berikan. Pemanggilan fungsi dilakukan dengan cara menuliskan nama fungsi tersebut, kemudian mengisikan argumen yang ada di dalam tanda kurung.

Misalkan fungsi tambah yang kita buat pada bagian sebelumnya:

```
function tambah(a, b) {
            return a + b;
        }
```

dapat dipanggil seperti berikut:

```
tambah(3, 5);
```

Code lengkap:

```
document.getElementById("hasil").innerHTML =
             "Hasil dari tambah(3, 5) adalah " + tambah(3, 5) + "<br>" +
             "Hasil dari kali(4, 5) adalah " + kali(4, 5);
```

Yang terjadi pada kode di atas adalah nilai a dan b masing-masing digantikan dengan 3 dan 5. Seperti yang dapat dilihat, hal ini berarti pengisian argumen pada saat pemanggilan fungsi harus berurut sesuai dengan deklarasi fungsi.

Sama seperti sebuah variabel, fungsi juga mengembalikan nilai ketika dipanggil. Dalam kasus di atas, tambah(3, 5) akan mengembalikan nilai 8. Nilai ini tentunya dapat disimpan ke dalam variabel baru, atau bahkan dikirimkan sebagai sebuah argumen ke fungsi lain lagi:

```
var simpan = tambah(3, 5); // simpan === 8
tambah(simpan, 2);  // mengembalikan 10
tambah(tambah(3, 5), 2)    // juga mengembalikan 10
tambah(tambah(2, 3), 4)    // mengembalikan 9
```

Fungsi akan mengembalikan nilai ketika kata kunci *return* ditemukan. Pengembalian nilai fungsi dapat dilakukan kapanpun, dan fungsi akan segera berhenti ketika kata kunci *return* ditemukan. Berikut adalah contoh kode yang memberikan gambaran tentang pengembalian nilai fungsi:

```
 var naikkan = function (n) { var
hasil = n + 10; return hasil;
 // kode di bawah tidak dijalankan lagi hasil
 = hasil * 100;
 } naikkan(10);     // mengembalikan
 20 naikkan(25); // mengembalikan 35
```

Sebuah ekspresi dapat juga diberikan langsung kepada *keyword return*, dan ekspresi tersebut akan dijalankan sebelum nilai dikembalikan. Hal ini berarti fungsi tambah maupun naikkan yang sebelumnya bisa disederhanakan dengan tidak lagi menyimpan nilai di variabel hasil terlebih dahulu:

Code lengkap:

```
  // Pemanggilan fungsi dan penyimpanan hasilnya
  var simpan = tambah(3, 5);         // simpan = 8
  var hasil1 = tambah(simpan, 2);    // 8 + 2 = 10
  var hasil2 = tambah(tambah(3, 5), 2); // (8) + 2 = 10
  var hasil3 = tambah(tambah(2, 3), 4); // (5) + 4 = 9

  // Tampilkan hasil ke HTML
  document.getElementById("hasilTambah").innerHTML =
    "tambah(3,5) = " + simpan + "<br>" +
    "tambah(simpan,2) = " + hasil1 + "<br>" +
    "tambah(tambah(3,5),2) = " + hasil2 + "<br>" +
    "tambah(tambah(2,3),4) = " + hasil3;

  document.getElementById("hasilKali").innerHTML =
    "kali(4,5) = " + kali(4,5); // contoh fungsi ekspresi
```

6.5. Pengenalan jQuery

jQuery adalah sebuah library Javascript yang dibuat oleh John Resig pada tahun 2006. jQuery memungkinkan manipulasi dokumen HTML dilakukan hanya dalam beberapa baris *code*. Beberapa fitur utama yang terdapat pada jQuery adalah:

- **DOM *manipulation*** – jQuery memungkinkan untuk memodifikasi DOM (*Document Object Model*) menggunakan *source selector* yang disebut dengan ***Sizzle***.
- **Event Handling** – jQuery dapat menangani sebuah aksi pada dokumen HTML seperti saat pengguna melakukan *click* pada sebuah objek.
- **Ajax *Support*** – jQuery dapat memfasilitasi pembuatan *website* menggunakan teknologi AJAX.
- ***Animations*** – pada jQuery terdapat *build-in* animasi yang dapat digunakan pada halaman *web*.
- ***Lightweight*** – ukuran *file* jQuery sangat ringan yaitu sekitar 19KB.

jQuery dapat dengan mudah digunakan pada sebuah situs *web* dengan berbagai cara.

1. Instalasi Lokal
   - Kunjungi link https://jquery.com/download/ untuk mengunduh *library* jQuery.
   - Letakkan *library* yang sudah diunduh pada satu folder yang sama dengan *file* HTML dengan kode berikut.

```
<html>

<head>
    <title>The jQuery Example</title>
    <script type="text/javascript" src="jquery-3.7.0.min.js">
    </script>

    <script type="text/javascript">
          $(document).ready(function() {
     document.write("Hello, World!");
        });
    </script>

</head>

<body>
    <h1>Hello</h1>
</body>
```

Note: pastikan atribut `src` memiliki nilai yang sama dengan nama *file library* jQuery.

- Buka *file* HTML tersebut menggunakan *web browser* seperti Mozilla atau Chrome. Dan hasil yang didapatkan adalah sebuah teks "Hello World" seperti yang ditulis pada bagian `document.write().`

2. Menggunakan CDN (*Content Delivery Network*)
   - Buka file HTML tersebut menggunakan *web browser* seperti Mozilla atau Chrome. Dan hasil yang didapatkan adalah sebuah teks "Hello World" seperti yang ditulis pada bagian `document.write().`

     `https://code.jquery.com/jquery-3.7.1.min.js`

   - Buka *file* HTML menggunakan *web browser* dan hasil yang ditampilkan akan sama dengan cara instalasi lokal.

6.6. Fungsi Dasar jQuery

Fungsi dasar jQuery digunakan untuk mempermudah manipulasi elemen HTML, menangani event, dan mengubah tampilan halaman web tanpa harus menulis kode JavaScript yang panjang. Dengan jQuery, pengembang dapat memilih elemen HTML menggunakan selector, lalu melakukan berbagai aksi seperti menampilkan atau menyembunyikan elemen, mengubah teks, warna, hingga menambahkan animasi dengan sintaks yang sederhana.

Contoh code:

1. Text

```
<!DOCTYPE html>
<html lang="id">

<head>
    <meta charset="UTF-8">
    <script src="jquery.js"></script>
    <script>
        $(document).ready(function () {
            $("body").append("<p>Halo, Andi!</p>");
        });
    </script>
</head>

<body>
</body>

</html>
```

Penjelasan Kode:

Kode ini menampilkan tulisan ke halaman menggunakan jQuery. Saat halaman selesai dimuat ($(document).ready()), fungsi .append() digunakan untuk menambahkan teks baru ke elemen <body>. Tujuannya agar isi halaman tidak hilang seperti jika memakai document.write().

2. Array

```
<!DOCTYPE html>
<html lang="id">
<head>
  <meta charset="UTF-8">
  <script src="jquery.js"></script>
  <script>
    $(document).ready(function() {
      var hobi = ["Membaca", "Coding", "Musik"];
      $("#hasil").text("Hobi favorit: " + hobi[1]);
    });
  </script>
</head>
<body>
  <p id="hasil"></p>
</body>
</html>
```

Penjelasan Kode:

Pada bagian ini dibuat variabel hobi berisi tiga elemen string. Elemen kedua (hobi[1]) diambil dan ditampilkan ke dalam paragraf dengan id="hasil". Fungsi .text() dari jQuery digunakan untuk menampilkan teks hasil ke elemen HTML secara dinamis.

3. Operasi Bilangan

```
<!DOCTYPE html>
<html lang="id">

<head>
    <meta charset="UTF-8">
    <script src="jquery.js"></script>
    <script>
        $(document).ready(function () {
            var a = 10;
            var b = 5;
            var hasil = a + b;

            $("#output").text("Hasil penjumlahan: " + hasil);
        });
    </script>
</head>

<body>
    <p id="output"></p>
</body>

</html>
```

Penjelasan Kode:

Bagian ini melakukan operasi aritmatika sederhana. Dua variabel a dan b dijumlahkan, hasilnya disimpan dalam variabel hasil. Kemudian nilai hasil ditampilkan ke elemen dengan id="output" menggunakan .text(). jQuery disini berperan untuk menampilkan hasil perhitungan ke halaman tanpa document.write().

4. If-Else

```
<!DOCTYPE html>
<html lang="id">

<head>
    <meta charset="UTF-8">
    <script src="jquery.js"></script>
    <script>
        $(document).ready(function () {
            var nilai = 75;

            if (nilai >= 70) {
                $("#hasil").text("Selamat, kamu lulus!");
                $("#hasil").css("color", "green");
            } else {
                $("#hasil").text("Maaf, kamu belum lulus.");
                $("#hasil").css("color", "red");
            }
        });
    </script>
</head>

<body>
    <p id="hasil"></p>
</body>

</html>
```

Penjelasan Kode:

Kode ini menunjukkan logika percabangan.Jika nilai >= 70, maka teks “Selamat, kamu lulus!” akan muncul dengan warna hijau. Jika tidak, akan muncul “Maaf, kamu belum lulus.” dengan warna merah. jQuery digunakan untuk menampilkan teks menggunakan .text() dan mengubah warnanya lewat .css() sesuai kondisi.

6.7. Kegunaan Lanjutan jQuery

6.6.1. Efek hide/show

Contoh *code*:

```
<!DOCTYPE html>
<html>

<head>
    <script src="https://code.jquery.com/jquery-3.7.1.min.js"></script>
    <script>
        $(document).ready(function() {
            $("#hide").click(function() {
                $("p").hide();
            });
            $("#show").click(function() {
                $("p").show();
            });
        });
    </script>
</head>

<body>

    <p>If you click on the "Hide" button, I will disappear.</p>
    <button id="hide">Hide</button>
    <button id="show">Show</button>

</body>
```

Penjelasan *code*:

- Pada baris 4-5, dilakukan pemanggilan *library* jQuery menggunakan CDN.
- Pada baris 6-15, fungsi jQuery *hide*/*show* diaplikasikan pada *tag* html <p>. Apabila *button* dengan id '`hide`' diklik, maka semua konten pada *tag* <p> akan disembunyikan. Selain itu, apabila *button* dengan id '`show`' diklik, maka semua konten pada *tag* <p> akan dimunculkan.

6.6.2. Efek animasi

Contoh *code*:

```
<!DOCTYPE html>
<html>

<head>
    <script src="https://code.jquery.com/jquery-3.7.1.min.js"></script>
    <script>
        $(document).ready(function() {
            $("button").click(function() {
                  $("div").anim
         ate({ height: 'toggle'
                });
            });
        });
    </script>
</head>

<body>

    <p>Click the button multiple times to toggle the animation.</p>
```

```
   <p>By default, all HTML elements have a static position, and cannot be moved.
To manipulate the position, remember to first set the CSS position property of
the element to relative, fixed, or absolute!</p>
     <div style="background:#98bf21; height:100px; width:100px;
  position:absolute;">
     </div>


 </body>


 </html>
```

Penjelasan *code*:

- Pada baris 6-15, fungsi jQuery menganimasikan "*toggle*" pada *tag* yang terdapat pada *body* HTML berdasarkan tinggi elemen