Modul praktikum - Minggu 07 - *Objects*

Dosen pengampu: **Henokh Lugo Hariyanto**
Asisten mata kuliah: **Jein Ananda - (10221031)**; **Muhammad Aulia Rahman - (10221055)**

Tujuan:

- Mampu memahami terminologi dasar terkait *object type*.
- Mampu menggunakan sintaks *object literals*.
- Mampu melakukan proses pembacaan, penulisan, penghapusan, enumerasi, dan eksistensi *properties* dari suatu *object*.
- Mampu memahami proses penggadaan *object*.

> Tips belajar bahasa pemrograman adalah mengetik ulang perintah yang kita temukan di buku atau di internet, lalu kita ubah-ubah untuk menguji pemahaman kita sudah tepat atau belum. Faktor bermain-main dan eksplorasi sangat diperlukan untuk memahami setiap perintah bahasa pemrograman yang kita pelajari. Setiap potongan kode di bawah dapat ditulis dalam berkas `.js` lalu dapat di-*running* dengan Node.js.

Object adalah suatu tipe data dalam JavaScript yang merupakan kumpulan dari elemen yang disebut *property*. *Property* ini memiliki nama (disebut *property name*) dan nilai (disebut *property value*). Perlu kita tekankan bahwa *object* disini berbeda dengan istilah *object* dalam *object-oriented programming*, meskipun dalam beberapa hal tipe data *object* ini memiliki istilah-istilah yang serupa yang dimiliki oleh *object-oriented programming*, seperti *inheritance*, *prototype*, dsb.

Beberapa sifat JavaScript *object* adalah sebagai berikut:

- Prototypical inheritance: merupakan salah satu *key feature* dari JavaScript yaitu kemampuan *object* untuk mendapatkan warisan (*inherit*) *properties* dari *object* yang lebih umum. Object yang lebih umum ini disebut *prototype*.
- Dynamics: *properties* dapat ditambahkan ke dalam *object* atau dapat juga dihapus
- Mutable: nilai suatu *property* dapat kita ubah-ubah, sifat *dynamics* juga menjamin sifat *mutable* ini.
- Manipulated by reference: dua buah variable yang mengacu ke object yang sama akan saling mempengaruhi apabila salah satu property di salah satu variable tersebut kita ubah-ubah. Penggandaan object perlu cara khusus agar tidak mempengaruhi satu sama lain ketika dilakukan modifikasi.
- No duplicate property names

Berikutnya akan kita bahas, teknik-teknik standar yang dapat dilakukan dengan *object*.

Membuat object

Ada beberapa cara untuk membuat *object*:

- Menggunakan *object literals*:

object-literal-examples.js

```
let empty = {};             // Suatu object tanpa property
console.log("empty: ", empty);

let point = {x: 0, y: 0};   // Suatu object dengan dua properties
console.log("point: ", point);

let p2 = {                  // Contoh object dengan nilai property
x: point.x,               // yang lebih rumit
y: point.y + 1};
console.log("p2: ", p2);

let student = {
"name": "Anastashia",         // Ketika di property name terdapat spasi
"student-id": 10111047,       // atau hypens gunakan string literal
interest: {                   // object dapat berisi object juga
    technical: ["C++", "Rust", "JavaScript"],
    non_technical: ["arts", "investing"]
}
};
console.log("student: ", student);
```

- Menggunakan `new Object_name`:

with-new-object.js

```
let o = new Object();   // Memuat suatu objek tanpa property: sama seperti
{}
console.log(o);

let a = new Array();    // Membuat suatu array tanpa elemen: sama seperti []
console.log(a);

let d = new Date();     // Membuat objek Date yang memiliki nilai
                        // waktu saat ini
console.log(d);

let r = new Map();      // Membuat objek Map yang tersusun atas
                        // key dan value
console.log(r);
```

- Menggunakan `Object.create()`

with-object-create.js

```
// Objek pertama dibuat yaitu objek literal ({x: 1, y: 1})
// Lalu Object.create() membuat objek baru dari objek literal tersebut
// sehingg semua property dalam objek literal tersebut akan di wariskan
// ke o1
let o1 = Object.create({x: 1, y: 1});
console.log(o1);

// Sama seperti objek o1, namun tidak memiliki property atau method
let o2 = Object.create(null);
console.log(o2);

// Sama seperti menggunakan objek literal {} atau new Object()
let o3 = Object.create(Object.prototype);
console.log(o3);
```

Memanggil dan mengatur nilai property

- Pemanggilan dan pengaturan nilai property bisa menggunakan dua cara yaitu menggunakan titik (`.`) atau kurung siku (`[]`).

  query-and-set-object.js

```
let student = {
  "name": "Anastashia",          // Ketika di property name terdapat spasi
  "student-id": 10111047,        // atau hypens gunakan string literal
  interest: {                    // object dapat berisi object juga
      technical: ["C++", "Rust", "JavaScript"],
      non_technical: ["arts", "investing"]
  }
};

let name = student.name;
let student_id = student["student-id"];
let interest_technical = student.interest.technical;
console.log(name, student_id, interest_technical);

student.cumul_grade = 3.67;
student.interest.non_technical = ["leadership", "cooking"];
console.log(student);
```

Menghapus property

Dengan menggunakan operator `delete`, kita dapat menghapus property suatu object.

Beberapa sifat-sifat operator `delete` ketika dikenakan pada suatu object:

- berlaku untuk menghapus property tidak hanya nilai dari property
- proses penghapusan selalu memberikan output `true` (meskipun property ada atau tidak ada di suatu object)

delete-demo.js

```
let student = {
  "name": "Anastashia",          // Ketika di property name terdapat spasi
  "student-id": 10111047,        // atau hypens gunakan string literal
  interest: {                    // object dapat berisi object juga
      technical: ["C++", "Rust", "JavaScript"],
      non_technical: ["arts", "investing"]
  }
};

// objek student sekarang tidak memiliki property name
delete student.name;

// object student sekarang juga tidak memiliki property name student-id
delete student["student-id"];

console.log(student)

// menghapus property yang tidak ada akan memberikan output true
console.log(delete student.name);
```

Menguji keberadaan property

Keberadaan suatu property dalam suatu object dapat diuji dengan empat cara:

- menggunakan `in` operator
- menggunakan method: `hasOwnProperty()`
- menggunakan method: `propertyIsEnumerable()`
- memanggil (querying) property (menggunakan titik atau kurung siku)

testing-properties.js

```
let student = {name: "Anastashia"};
console.log("name" in student);   // => true: object student memiliki
                                  //    property name
console.log("nim" in student);    // => false: object student tidak memiliki
                                  //    property nim
console.log("toString" in student);   // true: object student mewarisi
                                      // property toString dari Object prototype


console.log();
console.log(student.hasOwnProperty("name"));
console.log(student.hasOwnProperty("nim"));
console.log(student.hasOwnProperty("toString"));

console.log();
```

```
console.log(student.propertyIsEnumerable("name"));
console.log(student.propertyIsEnumerable("nim"));
console.log(student.propertyIsEnumerable("toString"));

console.log();
console.log(student.name !== undefined);
console.log(student.nim !== undefined);
console.log(student.toString !== undefined);
```

Untuk ekspresi `hasOwnProperty("toString")` dan `propertyIsEnumerable("toString")` memberikan hasil yang berbeda karena `toString()` bukan property yang dimiliki secara pribadi oleh object student, tetapi hasil warisan dari object prototype.

Enumerating Object

Merupakan proses untuk meng-iterasi property suatu object menggunakan control structure: `for/in`

enumerate-object.js

```
let student = {
  name: "Anastashia",
  nim: 10111047,
  age: 22
}

for (let p in student) {
  console.log(p, student[p]);
}

console.log();
for (let p in Object.keys(student)) {      // menggunakan Object.keys()
  console.log(p, student[p])
}
```

Extending objects

Untuk menggandakan property suatu objek ke property object lain dikenal dengan istilah *perluasan* (*extending*).

extend-manual.js

```
let target = {x: 1};
let source = {y: 2, z: 3};

for (let prop_name of Object.keys(source)) {
  target[prop_name] = source[prop_name];
}

console.log(target);
```

JavaScript menyediakan fungsi bawaan (*built-in function*) untuk melakukan proses *extending object* tanpa menggunakan `for/of`, yaitu menggunakan `Object.assign()`.

extend-object-assign.js

```
let target = {x: 1};
let source = {y: 2, z: 3};

Object.assign(target, source);
console.log(target);
```

Namun contoh di atas akan meng-*overwrite* (mengganti) nilai property `target` apabila property di `source` memiliki nama yang sama di `target`. Untuk itu diperlukan trik dengan cara membuat object kosong (`{}`) sebagai target dan menaruh object `target` sebaga argumen ke 3 dari `Object.assign()`

extend-object-no-overwrite.js

```
let target = {x: 1};
let source = {x: 5, y: 2, z: 3}

Object.assign(target, source);      // property x akan diganti nilainya dengan 5
console.log(target);

target = {x: 1};
source = {x: 5, y: 2, z: 3}
target = Object.assign({}, source, target);   // nilai x: 1 akan dipertahankan
console.log(target);
```

proses *extending object* sangat sering dilakukan yang biasanya digunakan untuk menggandakan *default property* ke object baru yang kita buat.

Serializing objects

Merupakan proses untuk mengubah suatu keadaan object (semua sisi kanan ketika kita mendeklarasikan nilai suatu object) menjadi suatu string panjang yang dapat kita simpan dan nantinya kita muat (*load*) ke dalam program.

Proses untuk mengubah menjadi string panjang menggunakan `JSON.stringify()` dan untuk membacanya kembali menjadi nilai suatu object menggunakan `JSON.parse()` ke dalam program.

serializing-object.js

```
let o = {
  x: 1,
  y: {
    z: [false, null, ""]
```

```
  }};
let string_result = JSON.stringify(o);
let parse_result = JSON.parse(string_result);

console.log("string_result: ", string_result);
console.log("parse_result: ", parse_result);
```

Built-in object methods

Pada bagian ini kita akan membahas *methods* bawaan dari suatu object (*built-in object methods*). `Methods` adalah istilah lain untuk fungsi namun digunakan untuk fungsi yang secara khusus melekat pada suatu object.

Di bagian ini hanya dijelaskan beberapa saja, lebih lengkap dapat dilihat di dokumentasi terkait object di MDN Web Docs.

- `toString()` method
  Merupakan *method* untuk memberi representasi dalam bentuk string suatu object. Umumnya *built-in method* `toString()` tidak memberikan hasil yang informatif, sehingga *programmer* sering mendefinisikan *method* `toString()` secara manual

  to-string-demo.js

```
let point_1 = {
  x: 1,
  y: 2,
}
console.log(point_1.toString());    // => [object Object]

let point_2 = {
  x: 1,
  y: 2,
  toString: function() { return `(${this.x}, ${this.y})`; }
}
console.log(point_2.toString());    // => (1, 2)
```

- `valueOf()` method
  Method ini digunakan untuk melakukan konversi ke tipe data selain tipe data string, misalnya ketika object dioperasikan melalui operasi penjumlahan dengan tipe data lain selain string. Secara *default*, *method* `valueOf` memberikan hasil yang tidak terlalu informatif, namun kita dapat mendefinisikan ulang *method* `valueOf` seperti yang kita mau.

  value-of-demo.js

```
let point_1 = {
  x: 1,
  y: 2,
}
console.log(point_1.valueOf());  // => { x: 1, y: 2}
```

```
let point_2 = {
  x: 1,
  y: 2,
  valueOf: function() { return [this.x, this.y]; }
}
console.log(point_2.valueOf());   // => [1, 2]
```

- `toJSON()` method
  Merupakan *method* yang digunakan oleh `JSON.stringify()` untuk melakukan konversi ke dalam bentuk string, nilai keadaan object. Dapat digunakan untuk melakukan modifikasi perintah `JSON.stringify` menjadi sesuai keinginan kita

  to-json-demo.js

```
let point_1 = {
  x: 1,
  y: 2
}
console.log(JSON.stringify(point_1));

let point_2 = {
  x: 1,
  y: 2,
  toString: function() { return `(${this.x}, ${this.y})`; },
  toJSON: function() { return this.toString(); }
};
console.log(JSON.stringify(point_2));
```

Sintaks tambahan untuk object literal

- Spread operator (`...`)
  Merupakan sintaks yang digunakan untuk menggantikan perintah `Object.assign()`

  spread-op.js

```
let target = {x: 1};
let source = {x: 5, y: 2, z: 3}

target = {...target, ...source};  // property x akan diganti nilainya dengan
5
console.log(target);

target = {x: 1};
source = {x: 5, y: 2, z: 3}
target = {...source, ...target};   // nilai x: 1 akan dipertahankan
console.log(target);
```

- Special methods (getter and setter)
  Methods ini disediakan agar ketika dilakukan proses *querying* (pemanggilan property) dan pengaturan nilai property dapat dimodifikasi sesuai keinginan kita.

  getter-and-setter.js

```
let student = {
  name: "Anastashia",
  student_id: 10111047,
  age: 22,

  get name_str() {return `Student's name: ${this.name}`},
  get age_str() { return `${this.name}'s age is ${this.age}`},
  get student_id_str() { return `${this.name}'s student id is
${this.student_id}`},

  set prop_val([new_name, new_student_id, new_age]) {
    this.name = new_name;
    this.student_id = new_student_id;
    this.age = new_age;
  }
}

console.log(student.name);
console.log(student.name_str);
console.log(student.student_id_str);
console.log(student.age_str);

console.log();
student.prop_val = ["Kyrlov", 10111051, 23];
console.log(student.name_str);
console.log(student.student_id_str);
console.log(student.age_str);
```

Tugas (Exercise - 04)

> Laporan harus ditulis dan dikumpulkan dalam bentuk berkas *markdown* atau berkas berekstensi `.md`.
> Apabila laporan memuat lebih dari satu berkas, misal memuat berkas gambar `.png` atau `.jpg`, maka berkas disatukan menjadi berkas `.zip`.
> **PASTIKAN** berkas `md` sudah dilakukan *preview*, sehingga kode *markdown* bisa di-*preview* dengan benar.
> Format penamaan file: `NIM_NAMA.md` atau `NIM_NAMA.zip` (boleh nama lengkap atau nama panggilan).
>
> **Contoh format laporan atau jawaban (`NIM_NAMA.md`)**
> Nama: [NAMA LENGKAP]
> NIM: [NIM]
>
> 1. (Jawaban nomor 1)
> 2. (Jawaban nomor 2)

1. [30 poin] Tuliskan pengalaman kalian selama seminggu eksplorasi terkait *object* type ini. Ceritakan dengan bahasa yang kalian pahami.

2. [70 poin]

    1. Buat suatu object kosong (tanpa property apapun) dengan nama `Person1`.
    2. Cetak object `Person1` tersebut di console
    3. Tambahkan properties berikut:

    ```
    name: "Anastashia",
    height: "178 cm",
    weight: "60 kg",
    skill: "photography, cosplayer"
    ```

    4. Tampilkan nilai nya saja dari properties yang telah ditambahkan
    5. Gandakan `Person1` ke `Person2` dan gunakan data berikut:

    ```
    name: "Kyrlov",
    height: "188 cm",
    weight: "78 kg",
    skill: "game developer, procedural animation"
    ```

    6. Tukarkan data-data `Person1` ke `Person2` demikian pula sebaliknya.
    Petunjuk: Gunakan *destructuring assignment*.
    7. Cetak object `Person1` dan `Person2` ke console.