Note TP1

- Test 1

```
typeof "John";
- 'string'
```

- Test 2

```
typeof 3.14;
- 'number';
```

- Test 3

```
typeof NaN;
- 'undefined';
```

- Test 4

```
 typeof false;
- 'boolean'
```

- Test 5

```
typeof null;
- 'object'
```

- Test 6

```
typeof undefined;
- 'undefined'
```

- Test 7

```
typeof un_truc_pas_declare;
- 'undefined'
```

- Test 8

```
   typeof {};
 - 'object'
```

- Test 9

```
typeof { name: "John", age: 34 };
        - 'object'
```

- Test 10

```
({}) instanceof Object;
- 'true'
```

- Test 11

```
typeof [1, 2, 3, 4];
- 'object'
```

- Test 12

```
[1, 2] instanceof Array;
- 'true'
```

- Test 13

```
[1, 2] instanceof Object;
- 'true'
```

- Test 14

```
typeof new Date();
- 'object'
```

- Test 15

```
typeof new Date("2020-02-02");
- 'object'
```

- Test 16

```
new Date() instanceof Date;
- 'true'
```

- Test 17

```
new Date() instanceof Object;
- 'true'
```

- Test 18

```
typeof function () {};
- 'function'
```

- Test 19

```
typeof (() => {});
- 'function'
```

- Test 20

```
(() => {}) instanceof Function;
- 'true'
```

- Test 21

```
(() => {}) instanceof Object;
- 'true'
```

Test 22

```
undefined == null; //return true
null === undefined; // return false

let example;
example == undefined; // true
example === undefined; // true
example == null; // true
example === null; // false

example = null;
example == undefined; // true
example === undefined // false
example == null; // true
example === null; // true

let sExample, iExample, bExample;
sExample = '1';
iExample = 1;
bExample = true;
sExample == iExample; // true
sExample === iExample; // true
iExample == bExample; // true
iExample === bExample; // true
sExample == bExample; // true
sExample === bExample; // true

let oExample0 = {a : 0};
let oExample1 = {a : 0};
oExample0 == oExample0; // true
oExample0 === oExample0; // true
oExample0 == oExample1; // false
```

```
let tExample0 = [0];
let tExample1 = [0];
tExample0 == tExample0; // true
tExample0 === tExample0; // true
tExample0 == tExample1; // false
tExample0[0] == tExample1[0]; // true
tExample0[0] === tExample1[0]; //true

"" == false; // true
"" === false; // false
"" == 0; // true
true == "true"; // false
true == "1"; // true
"10" + 5 == 15; // false
"10" + 5 == 105; // true
"10" + 5 === 105; //false
```

- **Exercice** À partir de l'exemple suivant, expliquez la différence de passage de paramètres en JavaScript entre les types primitifs et les objets.

  Réponse :

  - Les types primitifs sont passé par copie; La modification des types primitifs est valable uniquement dans la fonction; Les objets sont passé par référence; La modification par méthode est valable dans tout le projet; L'affectation d'un objet est valable uniquement dans la fonction;

```
let n = 42;
let b = false;
```

```
let s = "test";
let o = { name: "Romuald", age: 37 };
let a = [0, 1, 1, 2, 3, 5, 8];
let f = function () {
  return 42;
};

function test1(num, bool, str, obj, arr) {
  num++;
  bool = !bool;
  str += "foo";
  obj.name = "Olivier";
  arr.push(13);

  console.log(num, bool, str, obj, arr);
}

test1(n, b, s, o, a);
console.log(n, b, s, o, a);

function test2(obj, arr, fun) {
  obj = { foo: "bar" };
  arr = [42];
  fun = function () {
    return 0;
  };

  console.log(obj, arr, fun());
}
```

```
test2(o, a, f);
console.log(o, a, f);
```