# Note

## Concept

- Javascript has **the backwards compatibility**.
- Simply use the latest Google Chrome during development.
- Use **Babel** to transpile and polyfill your code during production(converting back to ES5 to ensure browser compatibility for all users).
- ES5: Fully supported in all browsers (down to IE 9 from 2011).

## Variable Declaration

- **CamelCase** variable name. e.g. firstName
- **DO NOT** use keyword like **name**, **function**
- **$** and **_** are only accepted sign in naming of variables.
- **Const** is named with all **UPPERCASE**, editor will show it with different color.
- String can use both **"** and **'**
- If a variable declared without **let**, **var** or **const** it will be global variable.

| **let** | **var** |
| --- | --- |
| Block scope | function scoped |
| does not allow to redeclare variables | allows to redeclare variables |
| Hoisting does not occur in let | Hoisting occurs in var. |

## primitive types

1. Number

```
let age = 23;
```

2. String

```
let firstName = 'Marc';
```

3. Boolean

```
let fullAge = true;
```

4. Undefined

```
let child;
```

5. Null
6. Symbol (ES2015): Value is unique nad cannot be changed. ***Not useful for now***
7. BigInt (ES2020): Larger integer than Number.

# Operator

- JavaScript has power operator **double stars**
- == vs === (!= vs !==)
  - == **loose operator**, it will do type coercion
  - === **strict operator**, it won't do type coercion, it should be the same type(object) and the same value.

```
18 === '18'     // false    strict operator
18 == '18'      // true     loose operator
18 != '18'      // false    loose operator
18 !== '18'     // true     strict operator
```

# String template (since ES6)

```
const firstName = 'Max';
const job = 'Programmer';
const birthYear = 1997;
const year = 2037;

const maxence = "I'm " + firstName + ", a " + (year - birthYear) + " years old " + job + "!";

const maxenceNew = `I'm ${firstName}, a ${year - birthYear} years old ${job}!`;
```

- maxence use **+** for concatenating strings.
- maxenceNew use **`** **`** for creating **string template**.
- With string template, we can create cross-line string.

```
console.log(`Today is  \n
    thursday`);               //Success

console.log("Today is \n
thursday");                   //SyntaxError
```

# Type Conversion

```
const inputYear = '1992';

//Number() to convert a string to a number
console.log(Number(inputYear) + 8);             // 2000
console.log(inputYear + 8);                     // 19928
console.log(Number('Jack'));                    // NaN => invalid number
console.log(typeof Number('Jack'));             // Number

//String() to convert a number to a string
console.log(typeof String(25));                 // String

//Falsy values: 0, '', undefined, null, NaN
console.log(Boolean(100));                      // true
console.log(Boolean('Marc'));                   // true
console.log(Boolean({}));                       // true
console.log(Boolean('0'));                      // true
console.log(Boolean(0));                        // false
console.log(Boolean(undefined));                // false
console.log(Boolean(''));                       // false
console.log(Boolean(null));                     // false
console.log(Boolean(NaN));                      // false
```

# Type Coercion

```
console.log("I'm " + 23 + "years old.");    // I'm 23 years old
console.log("25" - '10' - 7);               // 8 => coerce "25" and '10'(String) to Number
console.log("25" + '10' + 7);               // 25107 => coerce 7(Number) to String
console.log("25" * '10');                   // 250
console.log(7 + 3 + "10");                  //1010
```