JavaScript

MSc. Trần Thị Bích Hạnh

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Variable

- Variable name
  - the first character must be a letter, or an underscore (_), or a dollar sign ($)
  - **case sensitive**

```
var carName;
```

```
carName = "Volvo";
```

```
var carName = "Volvo";
```

```
var person = "John Doe", carName = "Volvo", price = 200;
```

Scope

```
var globalVar = "Global Variable";

function foo() {
  var localVar = "Local Variable";
  newGlobalVar = "New Global Variable";
  ...
}
```

let

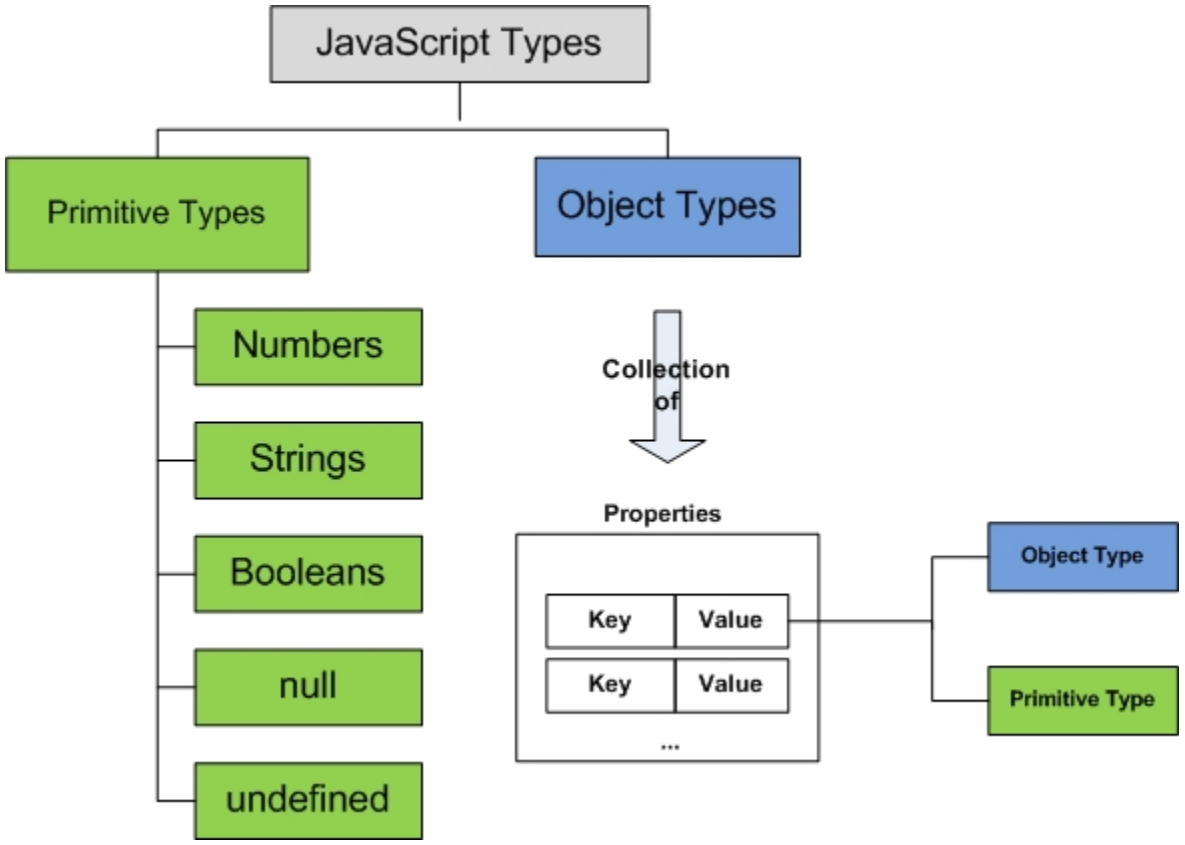
- Declare a variable with a block scope

```
var x = 10;
// Here x is 10
{
  let x = 2;
  // Here x is 2
}
// Here x is 10
```

const

```
const PI = 3.141592653589793;
PI = 3.14;      // This will give an error
PI = PI + 10;   // This will also give an error
```

Data Types


JavaScript Types
Primitive Types
Object Types
Numbers
Strings
Booleans
null
undefined
Collection of
Properties
Key
Value
Key
Value
...
Object Type
Primitive Type

JavaScript Types are Dynamic

```
var x;               // Now x is undefined
x = 5;               // Now x is a Number
x = "John";          // Now x is a String
```

```
var x = 5 + 5; // x = 10
var y = "5" + 5; // y = "55"
var z = "Hello" + 5; // z = "Hello5"
```

- Use parseInt(…), parseFloat(…), toString()

String Templates

```
let name = 'John';
`Hi ${name},
 Did you know that 5 * 3 = ${5*3}?`

/*
"Hi John,
 Did you know that 5 * 3 = 15?"
 */
```

Null vs Undefined

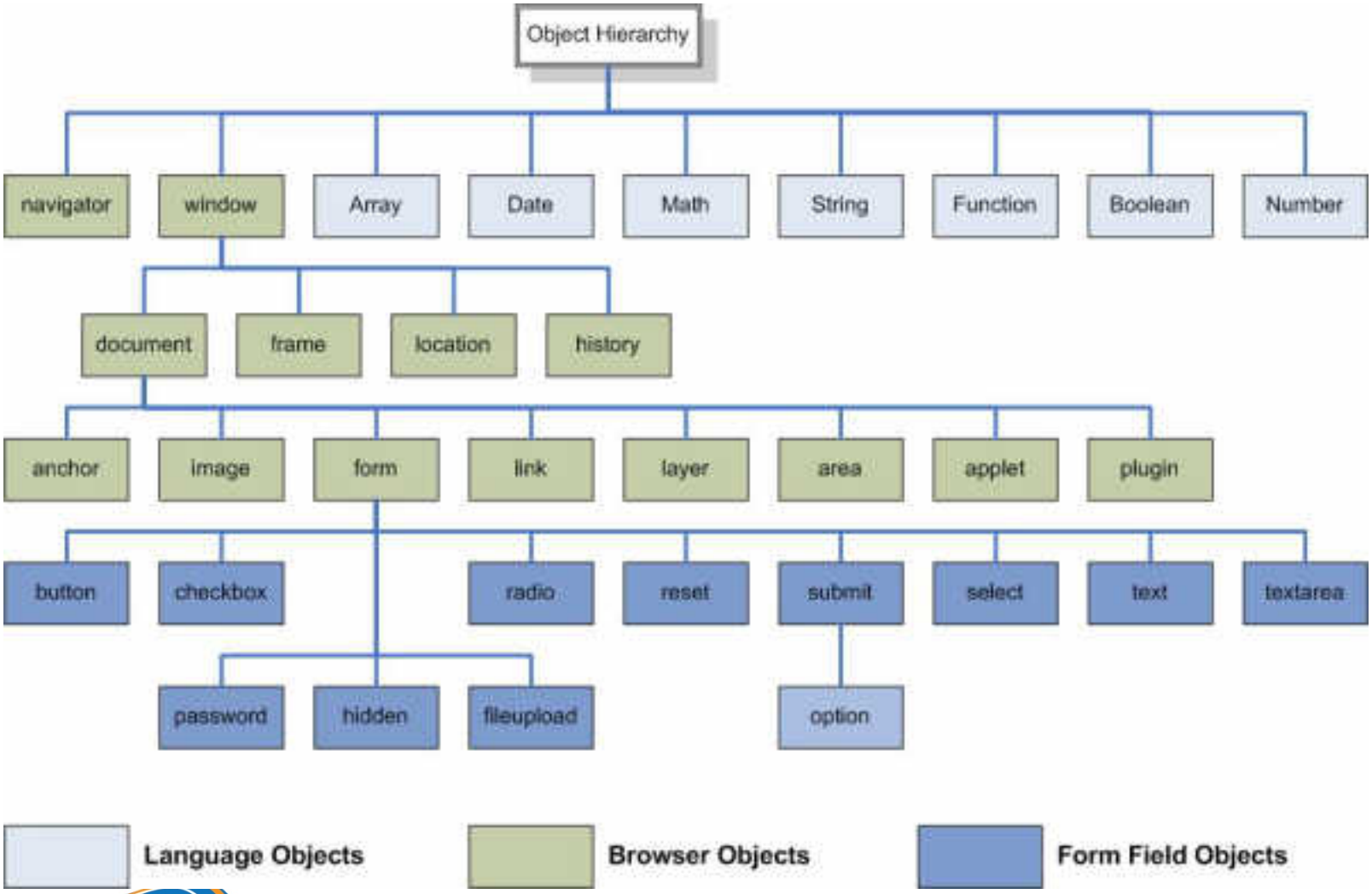
```
var car;                    // Value is undefined, type is undefined
```

```
var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
person = null;              // Now value is null, but type is still an object
```

```
typeof undefined            // undefined
typeof null                 // object

null === undefined          // false
null == undefined           // true
```

JavaScript Object Hierarchy



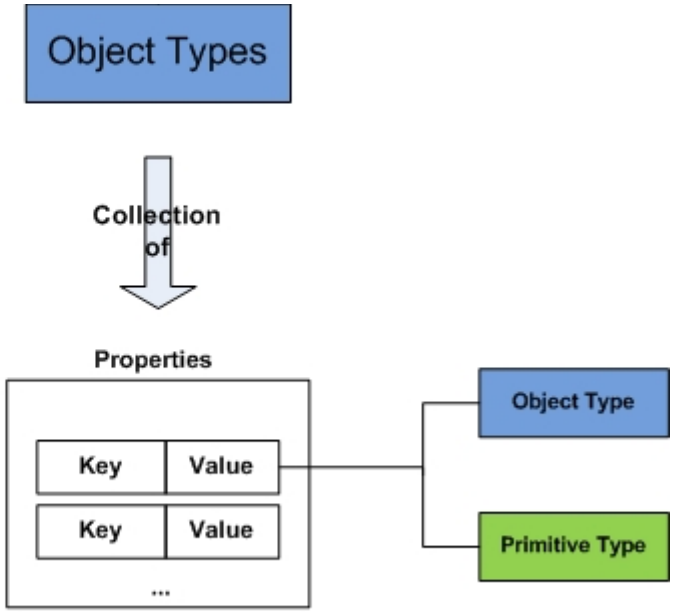
Array

```
var myArray = new Array();
// or
var myArray = [];
    myArray[0] = "Andrew";
    myArray[1] = "Monica";
    myArray[2] = "Catie";
    myArray[3] = "Jenna";
    myArray[4] = "Christopher";

// you can achieve the same result doing this:

var myArray = ["Andrew","Monica","Catie","Jenna","Christopher"];
```

Object




```
// Example 1
var myFirstObject = {};
myFirstObject.firstName = "Andrew";
myFirstObject.lastName = "Grant";
console.log(myFirstObject.firstName);
// Example 2
var mySecondObject = {
    firstName: "Andrew",
    lastName: "Grant"
};
console.log(mySecondObject.firstName);
// Example 3
var myThirdObject = new Object();
myThirdObject.firstName = "Andrew";
myThirdObject.lastName = "Grant";
console.log(myThirdObject.firstName);
```

Shorthand assignment

```
let x = 3;
let xo =   {x};
// xo = { x: 3 }
```

equivalent to:

```
let x = 3;
let xo =   {x:x};
// xo = { x: 3 }
```

JSON – JavaScript Object Notation

```
var jsonObject = {
      "firstName": "John",
      "lastName": "Smith",
      "address": {
          "streetAddress": "21 2nd Street",
          "city": "New York",
          "state": "NY",
          "postalCode": 10021
      },
      "phoneNumbers": [
          "212 555-1234",
          "646 555-4567"
      ]
};
```

function

```
function myFunction(x, y) {
    if (y === undefined) {
        y = 0;
    }
}
```

```
function myFunction(p1, p2) {
    return p1 * p2;
}
```

Default Parameter Values

```
function myFunction(x, y = 10) {
  // y is 10 if not passed or undefined
  return x + y;
}
myFunction(5); // will return 15
```

Arrow Functions

```
function inc(x) {
  return x + 1;
}
```

is equivalent to:

```
let inc = x => x + 1;
```

2 parameters:

```
let inc = (x, y) => x + y;
```

no parameters

```
let inc = () => 7;
```

Arrow Functions

more than 1 statement

```
let inc = (x) => {
   console.log(x);
   return 7;
}
```

arguments

```
x = sumAll(1, 123, 500, 115, 44, 88);

function sumAll() {
    var i, sum = 0;
    for (i = 0; i < arguments.length; i++) {
        sum += arguments[i];
    }
    return sum;
}
```

Anonymous self-invoking function

```
(function () {
    var x = "Hello!!";      // I will invoke myself
})();
```

Callback

```
var actionsToTakeWhenServerHasResponded = function () {
  console.log('The server just responded!')
}
function communicateWithServer(callback) {
  callback();
}
communicateWithServer(actionsToTakeWhenServerHasResponded);
```

Define function as Object property

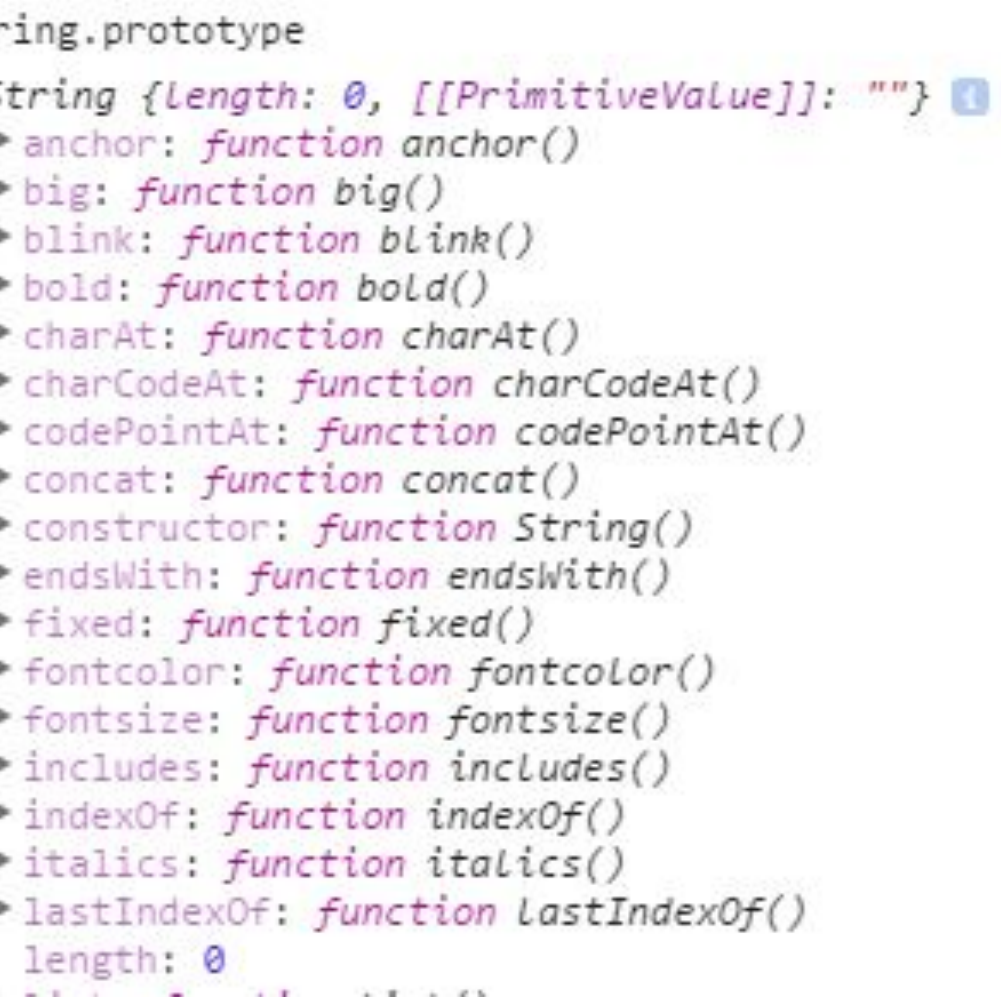
```
var myCleverObject = {
        firstName: "Andrew",
        age: 21,
        myInfo: function () {
            console.log("My name is " + this.firstName + ". ");
            console.log("My age is " + this.age + ".");
        }
};
    myCleverObject.myInfo();
```

Function as Object

```
function person(first, last) {
    this.firstName = first;
    this.lastName = last;
    this.fullname = function() {
      return this.firstName + " " + this.lastName;
    };
}
person.prototype.nationality = "English";

var myFather = new person("John", "Doe");
console.log(myFather.fullname());
```

prototype


> String.prototype
String {length: 0, [[PrimitiveValue]]: ""}
anchor: function anchor()
big: function big()
blink: function blink()
bold: function bold()
charAt: function charAt()
charCodeAt: function charCodeAt()
codePointAt: function codePointAt()
concat: function concat()
constructor: function String()
endsWith: function endsWith()
fixed: function fixed()
fontcolor: function fontcolor()
fontsize: function fontsize()
includes: function includes()
indexOf: function indexOf()
italics: function italics()
lastIndexOf: function lastIndexOf()
length: 0

Adding Properties and Methods to Objects

```
var str = 'abc'; // str là string, cha nó là String.prototype


// nhân đôi chuỗi đưa vào
String.prototype.duplicate = function() { return this + this; }


console.log(str.duplicate()); // Tìm thấy hàm duplicate trong prototype
```

Object.prototype

```
var person = {
   firstName: 'Hoang',
   lastName: 'Pham',
   showName: function() {
      console.log(this.firstName + ' ' + this.lastName);
   }
}; // object này có prototype là Object.prototype
```

Prototype

```
function Person(firstName, lastName) {
  this.firstName = firstName;
  this.lastName = lastName;
  this.showName = function() {
     console.log(this.firstName + ' ' + this.lastName);
  };
}

var otherPerson = new Person('Hoang', 'Pham'); // object này có
// Prototype mới: Person.prototype được tạo ra
// Person.prototype kế thừa Object.prototype
```

Prototype Inheritance

```
// Viết một Constructor Function khác
function SuperMan(firstName, lastName) {
  this.firstName = firstName;
  this.lastName = lastName;
}

// Ta muốn SuperMan sẽ kế thừa các thuộc tính của Person
// Sử dụng prototype để kế thừa
SuperMan.prototype = new Person();

// Tạo một object mới bằng Constructor Function
var sm = new SuperMan('Hoang', 'Pham');
sm.showName(); // Hàm này kế thừa từ prototype của Person
```

References

- https://www.w3schools.com/js/default.asp