Definició de classes i objectes

- **Àmbit:**

  Totes les propietats i mètodes dels objectes en JavaScript de manera predeterminada són públics

- **Modificar objectes:**

  Es pot definir un nou mètode per a qualsevol classe existent mitjançant la propietat **prototype.**

- Per exemple si volem afegir un nou mètode disponible per a tots els objectes:

```
Object.prototype.showValue = function () {
   alert(this.valueOf());
}
var sCadena = "Hola";
var iNum = 25;
sCadena.showValue(); //retorna "Hola"
iNum.showValue();    //retorna 25
```

Definició de classes i objectes

- **Paradigma de constructors:**
- Tot el codi definit dins de:
  function {

  }

  forma part del constructor.
- Els mètodes i propietats es defineixen en el constructor, i s'utilitza **this** per crear-los
- Quan es crea una instància, el constructor duplica les funcions i fa una còpia diferent per a cada objecte.

```
function Cotxe(sColor, iPortes, iCv) {
    this.color = sColor;
    this.portes = iPortes;
    this.Cv = iCv;
    this.showColor = function () {
        alert(this.color)
    };
}

var oCotxe1 = new Car("verd", 4, 80);
var oCotxe2 = new Car("blau", 3, 110);
oCotxe1.showColor();
oCotxe2.showColor();
```

Definició de classes i objectes

- **Paradigma de prototips:**
- El constructor està buit, i només s'utilitza per definir el nom de la classe.
- Fa ús de la propietat **prototype** per crear tant les propietats com els mètodes.
- No es poden establir els valors inicials a les propietats, de manera que els valors s'han d'assignar després de crear l'objecte.
- Quan una propietat apunta a un objecte, un array en l'exemple, els canvis afecten a la resta d'instàncies

```
function Cotxe() {
}

Cotxe.prototype.color = "verd";
Cotxe.prototype.portes = 4;
Cotxe.prototype.cv = 23;
Cotxe.prototype.conductors = new Array("Marc", "Miquel");
Cotxe.prototype.showColor = function () {
    alert(this.color);
};

var oCotxe1 = new Cotxe();
var oCotxe2 = new Cotxe();

oCotxe1.conductors.push("Pol");
oCotxe1.color = "vermell";

alert(oCotxe1.color);
alert(oCotxe2.color);
alert(oCotxe1.conductors);
alert(oCotxe2.conductors);
```

Definició de classes i objectes

- **Paradigma híbrid de constructors i prototips:**
- Les propietats es defineixen en el constructor.
- Els mètodes van fora i es fa ús de la propietat **prototype** per crear-los. Cada objecte pot tenir la seva instància de les propietats independent.
- Es crea només una instància dels mètodes
- Aquest paradigma combina els aspectes positius dels paradigmes de constructor i prototypes sense els efectes secundaris.

```
function Cotxe(sColor, iPortes, iCv) {
    this.color = sColor;
    this.portes = iPortes;
    this.Cv = iCv;
    this.conductors = new Array("Marc", "Miquel");
}

Cotxe.prototype.showColor = function () {
    alert(this.color);
};

var oCotxe1 = new Cotxe("verd", 4, 80);
var oCotxe2 = new Cotxe("blau", 3, 110);

oCotxe1.conductors.push("Pere");

alert(oCotxe1.conductors);
alert(oCotxe2.conductors);
```

Definició de classes i objectes

- **Mètode de prototips dinàmics:**
- Les propietats es defineixen en el constructor.
- Els mètodes fan ús de la propietat **prototype** per crear-los. La sentència: **typeof Cotxe.initialized==”undefined”** comprova si el mètode no ha estat definit, si és així el crea, si no continua i no el torna a definir. D'aquesta manera només es crea una còpia.
- Aquest paradigma té la mateixa estructura que en altres llenguatges com Java per definir classes.

```
function Cotxe(sColor, iPortes, iCv) {
    this.color = sColor;
    this.portes = iPortes;
    this.Cv = iCv;
    this.conductors= new Array("Marc", "Miquel");

    if (typeof Cotxe._initialized == "undefined") {

        Cotxe.prototype.showColor = function () {
            alert(this.color);
        };

        Cotxe._initialized = true;
    }
}

var oCotxe1 = new Cotxe("verd", 4, 80);
var oCotxe2 = new Cotxe("blau", 3, 110);

oCotxe1.conductors.push("Pere");

alert(oCotxe1.conductors);
alert(oCotxe2.conductors);
```