Div

I stedet for å lage en options-liste sånn her:

```
<select name="drop1" id="Select1">
  <option value="volvo">Volvo</option>
  <option value="saab">Saab</option>
  <option value="mercedes">Mercedes</option>
  <option value="audi">Audi</option>
</select>
```

Lag den sånn her (dynamisk)

```
var myParent = document.body;

//Create array of options to be added
var array = ["Volvo","Saab","Mercades","Audi"];

//Create and append select list
var selectList = document.createElement("select");
selectList.id = "mySelect";
myParent.appendChild(selectList);

//Create and append the options
for (var i = 0; i < array.length; i++) {
    var option = document.createElement("option");
    option.value = array[i];
    option.text = array[i];
    selectList.appendChild(option);
}
```

Prototype

Tillater deg å legge til nye properties(props) til objekt konstruktøren

```
function Person(first, last, age, eyecolor) {
  this.firstName = first;
  this.lastName = last;
  this.age = age;
  this.eyeColor = eyecolor;
}

Person.prototype.nationality = "English";
```

Tillater deg å legge til nye metoder i objekt konstruktøren

```
function Person(first, last, age, eyecolor) {
  this.firstName = first;
  this.lastName = last;
  this.age = age;
  this.eyeColor = eyecolor;
}

Person.prototype.name = function() {
  return this.firstName + " " + this.lastName;
};
```

Forskjell på klasse og objekt

I objektorientert programmering er en klasse en abstrakt beskrivelse av en type objekt. En klasse definerer de egenskapene og metodene som er tilgjengelige for alle objekter av denne typen.

Et objekt, derimot, er en konkret instans av en klasse. Det er et unikt eksemplar av en klasse med sine egne spesifikke egenskaper og metoder.

For eksempel kan du tenke på en klasse som en oppskrift for å lage en kopp kaffe. Oppskriften definerer hva slags ingredienser og verktøy som trengs, og trinnene som trengs for å lage en kopp kaffe. Når du faktisk brygger en kopp kaffe ved hjelp av oppskriften, skaper du et objekt av klassen "kopp kaffe". Den konkrete koppen kaffe du har laget, med sin spesifikke smak og temperatur, er et eksempel på et objekt av klassen "kopp kaffe".

Så, en klasse er en beskrivelse av en type objekt, mens et objekt er en konkret instans av denne typen.

Eksempel

La oss si at du ønsker å lage en applikasjon for å håndtere biler. Du kan da definere en klasse kalt "Car" som vil representere konseptet av en bil, med egenskaper som modell, farge, antall dører, etc. Eksempel på kode for å definere en "Car"-klasse i JavaScript kan se slik ut:

```
class Car {
  constructor(model, color, numOfDoors) {
    this.model = model;
    this.color = color;
    this.numOfDoors = numOfDoors;
  }

  start() {
    console.log("Car started");
  }

  stop() {
    console.log("Car stopped");
  }
}
```

I dette eksempelet definerer vi "Car"-klassen med tre egenskaper - "model", "color" og "numOfDoors" - som alle settes når et objekt av klassen opprettes. Vi definerer også to metoder - "start" og "stop" - som kan kalles på et objekt av klassen for å starte eller stoppe bilen.

Når vi ønsker å bruke denne klassen, kan vi opprette et eller flere objekter av "Car"-klassen og sette egenskapene deres. Her er et eksempel på hvordan vi kan opprette et objekt av "Car"-klassen og kalle metoden "start":

```
let myCar = new Car("Toyota", "red", 4);
myCar.start();
```

Her oppretter vi et objekt av "Car"-klassen med modellen "Toyota", fargen "red" og 4 dører. Vi lagrer dette objektet i en variabel kalt "myCar", og kaller deretter "start" - metoden på objektet for å starte bilen.

Så, i korte trekk, kan vi si at en klasse er en oppskrift eller en mal for å lage objekter, mens et objekt er en spesifikk instans av denne klassen med unike egenskaper og metoder.

Typer

I TypeScript kan man bruke mange av de samme datatypene som i JavaScript, samt noen ekstra datatyper. Her er en liste over noen av de vanligste datatypene som kan brukes i TypeScript og JavaScript:

1. **number**: Brukes til tall, både heltall og desimaltall.
2. **string**: Brukes til tekststrenger.
3. **boolean**: Brukes til å representere sant eller usant.
4. **null**: Brukes til å representere at en verdi ikke eksisterer.
5. **undefined**: Brukes til å representere at en verdi ikke er definert.
6. **any**: Brukes når man ikke ønsker å spesifisere en datatype, eller når en variabel kan ha en hvilken som helst datatype.
7. **object**: Brukes til å representere et objekt.
8. **Array**: Brukes til å representere en liste av verdier av samme datatype.
9. **Tuple**: En spesiell type array som har en fastsatt lengde og spesifisert datatype for hvert element.
10. **Enum**: Brukes til å definere en samling av navngitte konstanter.
11. **Void**: Brukes som return type for funksjoner som ikke returnerer noen verdi.
12. **Never**: Brukes som return type for funksjoner som aldri terminerer eller returnerer noe.

I tillegg til disse datatypene, finnes det også noen spesifikke datatyper som brukes i JSX, som er en syntaks for å beskrive brukergrensesnitt i React-applikasjoner:

1. JSX.Element: Brukes til å representere et React-element, for eksempel en knapp eller et tekstfelt.
2. React.ReactNode: Brukes til å representere et React-element eller en liste av React-elementer.
3. React.ComponentType: Brukes til å representere en React-komponent.