# Objects in ES6

- In JavaScript variables can be object data types that contain many values called properties.
- An Object can also have properties that are functions defination called methods for performing actions on the object.
- ES6 introduces shorthand notations and computed property names that make declaring objects easier.
- The new defination shorthand notation does not require : or a function keyword.
- Example:

```
let tree = {
  height: 10,
  color: "green",
  grow() {
    this.height += 10;
  },
};

tree.grow();
console.log(tree.height); // 20
```

- While creating duplicate properties the last one will be used.

```
var a = { x: 1, x: 2 };
console.log(a.x); // 2
```

- While creating duplicate properties in ES5 generated SyntaxError.

## Quick Quiz

- Fill in the blanks for the code to output 50 :

```
let car = {
  speed: 30,
  accelerate() {
    ___.speed += 10;
  },
};
___.accelerate();
car.accelerate();
console.log(___.speed);
```

## Computed Property Names

- With ES6 we can use computed property names to create dynamic property names.
- ☐ syntax we can create a property name dynamically.
- Example:

```
let prop = "name";
let id = "1234";
let mobile = "09631";
let user = {
  [prop]: "John",
  [`user_${id}`]: `${mobile},
};
```

## Example

```
var i = 0;
var a = {
  ["foo" + ++i]: i,
  ["foo" + ++i]: i,
  ["foo" + ++i]: i,
};
console.log(a); // { foo1: 1, foo2: 2, foo3: 3 }
```

## Example

```
var param = "size";
var config = {
  [param]: 12,
  ["mobile" + param.charAt(0).toUpperCase() + param.slice(1)]: 5,
};
console.log(config.mobileSize); // 5
```

- Its very useful when we want to create dynamic property names, custom based on some variable rules

## Quick Quiz

- Fill in the blanks to create an object with its properties:

```
let prop = "foo";
let obj = {
 _prop]: "bar"_
  [prop.toUpperCase()]_ "BAR",
  [prop.length]: 3_
};
console.log(obj); // { foo: 'bar', FOO: 'BAR', 3: 3 }
```

## Object.assign() in ES6

- Object.assign() is a new built-in function that copies the values of all enumerable own properties from one or more source objects to a target object.
- Its usefull for creating deep copies of objects.
- Example:

```
let obj1 = { a: 1, b: 2 };
let obj2 = { b: 4, c: 5 };
let obj3 = { c: 6 };
let newObj = Object.assign({}, obj1, obj2, obj3);
console.log(newObj); // { a: 1, b: 4, c: 6 }
```

- Here we are creating a new object and copying the properties of the other objects into it, the first object is the target object {}.
- Every parameter passed to Object.assign() is an object that will be copied to the target object.
- The order of the parameters is important, the first object will be copied to the target object.
- The target object will be modified.
- The simple = assignment creates a reference to the object, so the target object will be modified in mutation.
- To avoid this behavior we can use Object.assign() to create a new object and copy the properties of the other objects into it.

## Quick Quiz

- What will be the output of the following code?

```
const ob1 = { a: 1, b: 2, c: 4 };
const ob2 = Object.assign({ c: 5, d: 6 }, ob1);
console.log(ob2.c, ob2.d);
```

> Answer: 4, 6 as the target object is a new object it will be modified.