# Object Definition

## Methods for Defining JavaScript Objects

1. Using an Object Literal
2. Using the new Keyword
3. Using an Object Constructor
4. Using Object.assign()
5. Using Object.create()
6. Using Object.fromEntries()

### 1. JavaScript Object Literal

```
{firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
```

### 2. Creating a JavaScript Object

```
// Create an Object
const person = {};

// Add Properties
person.firstName = "John";
person.lastName = "Doe";
person.age = 50;
person.eyeColor = "blue";
```

### 3. Object Constructor Functions

- Sometimes we need to create many objects of the same type.
- To create an object type we use an object constructor function.
- It is considered good practice to name constructor functions with an upper-case first letter.
- You can not add a new property to an existing object constructor:

```
function Person(first, last, age, eye) {
  this.firstName = first;
  this.lastName = last;
  this.age = age;
  this.eyeColor = eye;
 this.nationality = "English";  //Property Default Values
}

const myFather = new Person("John", "Doe", 50, "blue");
```

**<u>JavaScript Object Methods</u>**

JavaScript Object Methods can be grouped into:

1. General Methods
2. Property Management Methods
3. Object Protection Methods

## 1. General Methods

- // Copies properties from a source object to a target object
  Object.assign(target, source)

- // Creates an object from an existing object
  Object.create(object)

- // Returns an array of the key/value pairs of an object
  Object.entries(object)

- // Creates an object from a list of keys/values
  Object.fromEntries()

- // Returns an array of the keys of an object
  Object.keys(object)

- // Returns an array of the property values of an object
  Object.values(object)

- // Groups object elements according to a function
  Object.groupBy(object, callback)

## 2. Property Management Methods

- // Adding or changing an object property
  Object.defineProperty(object, property, descriptor)

- // Adding or changing object properties
  Object.defineProperties(object, descriptors)

- // Accessing a Property
  Object.getOwnPropertyDescriptor(object, property)

- // Accessing Properties
  Object.getOwnPropertyDescriptors(object)

- // Returns all properties as an array
  Object.getOwnPropertyNames(object)

- // Accessing the prototype
  Object.getPrototypeOf(object)

3. **Object Protection Methods**

   - // Prevents re-assignment
     const car = {type:"Fiat", model:"500", color:"white"};

   - // Prevents adding object properties
     Object.preventExtensions(object)

   - // Returns true if properties can be added to an object
     Object.isExtensible(object)

   - // Prevents adding and deleting object properties
     Object.seal(object)

   - // Returns true if object is sealed
     Object.isSealed(object)

   - // Prevents any changes to an object
     Object.freeze(object)

   - // Returns true if object is frozen
     Object.isFrozen(object)

## Object Prototypes

### Adding Properties and Methods to Objects

- Sometimes you want to add new properties (or methods) to all existing objects of a given type.
- Sometimes you want to add new properties (or methods) to an object constructor.

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

★ **Using the prototype Property**

- ***Person.prototype.nationality = "English";***

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