Main Points/Key Points	Notes

	ES6 Features
	
	1. Arrow Functions.
	a. It is a new way of writing a compact function. Developers called them as *lambda* or *fat arrow* functions.
	b. Arrow functions eliminate the use of *function* and *return* keywords for shorter syntax.
	c. There are few ways to declare a function in JavaScript ES5, these include *function declaration*, *expression*, *named* and *object*.
	d. Examples:
	
	ES5 Functions
	`// Function Declaration`
	`function sum(x,y){`
	`  return x+y;`
	`}`
	
	`// Function Expression`
	`let sum = function(x,y){`
	`  return x+y;`
	`}`
	
	`// Named Function Expression`
	`let sum = function sum(x,y){`
	`  return x+y;`
	`}`
	
	`// Function Object`
	`function car(model, manufacturer){`
	`  this.model = model;`
	`  this.manufacturer = manufacturer;`
	`}`

	Summary

Main Points/Key Points	Notes

	ES6 Features
	
	e. In ES6, function expression is used to represent the arrow function together with => operator.
	f. Examples:
	ES6 Arrow Functions
	`let sum = (x,y) => {return x+y};`
	`let sum = (x,y) => x+y;`
	`//both are identical or implicit return`
	
	`let sum = () => x+y;`
	`let sum = x => x+10;`
	`let sum = (x = 10) => x+y;`
	`let sum = (x = 10, y = 23) => x+y;`
	
	g. Arrow functions are **NOT** suitable for *Object Methods* and *Constructors*.
	h. Example:
	ES6 Method & Constructor
	`let car = (model, manufacturer) =>{`
	`  this.model = model;`
	`  this.manufacturer = manufacturer;`
	`}`
	
	`var myCar = new car();`
	`// Not a constructor`
	
	`let car = {`
	`  manufacturer: 'Honda',`
	`  display: () => {`
	`  console.log(this.manufacturer);`
	`  }`
	`}`
	`car.display()`
	`// undefined`

	Summary

Main Points/Key Points	Notes

	ES6 Features
	
	i. Arrow functions are useful for returning an *object*.
	j. Example:
	
	ES6 Return An Object
	`const getInfo = () => ({ name: 'Hassan Basri', company: 'Google', job: 'Data Engineer', });`
	
	`getInfo(); // return { name: 'Hassan Basri', company: 'Google', job: 'Data Engineer' }`
	
	2. **Template String (Literal).**
	a. It allows developers to write or display an output with dynamic contents.
	b. It removes the need for **"+" operator** to concatenate or join multiple strings.
	c. It uses back ticks **(` `)** to define the string and pass the variable with **${ }** template.
	d. Example:
	
	ES Template String
	`const total = (x, y) => { return `Total, ${x+y}! Reduce ${x-y}?`; };`

	Summary

Main Points/Key Points	Notes
	ES6 Features **3. For of Loop** a. It is similar to **for in** and **for loop** in ES5 but with compact declaration. b. Example:

ES6 For of Loop

```
let arr = [2,3,4,1];

for (let value of arr) {
 console.log(value);
}

Output:
2
3
4
1

let string = "Javascript";

for (let char of string) {
 console.log(char);
}

Output:
J
a
v
a
s
c
r
i
p
t
```

Summary

Main Points/Key Points	Notes

	ES6 Features
	
	4. Object Destructuring.
	a. It allows developers to **break down** or **select** useful information from an object.
	b. The selected information is assigned to predefined variables.
	c. It is useful for object and array.
	
	ES6 Object Destructuring
	
	`const info = {`
	`  name: 'Spencer',`
	`  company: 'Handlebar Labs',`
	`  location: {`
	`  city: 'Nashville',`
	`  state: 'Tennessee',`
	`  },`
	`};`
	
	`const { name, location } = info;`
	`const { city, state } = location;`
	
	`console.log(name);`
	`// name is Spencer`
	`Console.log(city);`
	`// city is Nashville`

	Summary

Main Points/Key Points	Notes
	ES6 Features 5. **Object Spread.** a. It allows developer to copy one object to another. b. It mostly used in React Native and represented with (**...**) symbol to copy the existing object. c. Example:

ES6 Object Spread

```
const handlebarLabsInfo = {
  company: 'Handlebar Labs',
  location: {
    city: 'Nashville',
    state: 'Tennessee',
  },
};

const spencerInfo = {
  ...handlebarLabsInfo,
  name: 'Spencer',
}

console.log(spencerInfo);

//
{ name: 'Spencer',
company: 'Handlebar Labs',
location: {
city: 'Nashville',
state: 'Tennessee' }
}
```

Summary

Main Points/Key Points	Notes

	ES6 Features
	
	6. **Classes.**
	a. It is a special function and a better way to define an object in JavaScript.
	b. It allows developer to define an object in clean and Java like syntax.
	c. Example:
	
	ES6 Classes
	
	`class People {`
	`  constructor(name) {`
	`  this.name = name;`
	`  }`
	
	`  get Name() {`
	`  return this.name;`
	`  }`
	
	`  set Name(name) {`
	`  this.name = name;`
	`  }`
	`}`
	
	`let person = new People("Jon Snow");`
	`console.log(person.Name);`
	`person.Name = "Dany";`
	`console.log(person.Name);`
	
	d. The class definition allows inheritance or subclass similar to Java.

	Summary

Main Points/Key Points	Notes

	ES6 Features
	
	7. Modules.
	a. Modules in JavaScript can be:
	i. **Import** allows developers to use or call exported functions, objects and primitive values.
	ii. **Export** allows developers to distribute functions, objects, and primitive values to be called by another page or file.
	b. Example:
	
	Export.js
	`let func = a => a + a;`
	`let obj = {};`
	`let x = 0;`
	
	`export { func, obj, x };`
	
	`// or export default class {…}`
	
	Import.js
	`import { func, obj, x } from './Export.js';`
	
	`console.log(func(3), obj, x);`
	
	`// or import class from ‘./Export.js’;`

	Summary

Main Points/Key Points	Notes
	ES6 Features **8. References:** *a.* Carli, S. (2017). *A Brief Overview of ES6 for React Native Developers*. Retrieved from https://medium.com/the-react-native-log/a-brief-overview-of-es6-for-react-native-developers-15e7c68315da b. Rascia, T. (2018). *ES6 Syntax and Feature Overview*. Retrieved from https://www.taniarascia.com/es6-syntax-and-feature-overview/#arrowfunctions c. ES6 Tutorial (2018). Retrieved from https://www.tutorialspoint.com/es6/index.htm
	Summary