Javascript & Typescript

Javascript

- programming language created initially for making webpages alive (animations or manipulations of DOM)
- interpreted language
- first language in Github

Javascript

- Where is Javascript used today?
  - Mobile platforms: Cordova, React Native
  - Backend: Node JS
  - Frontend
  - Wearables: JerryScript, Fitbit Developer Platform

- It is possible to build a full stack system only with JS

Javascript

- Data types
- Objects
- Variable scope
- Callbacks
- ECM6 Features: Classes

Javascript – Primitive Data Types

- String, Number, Boolean, Undefined and Null
- Value types
- Immutable

```
var a = "Hello!"
var b = a
a = "Hi!"
console.log(a) // "Hi!"
console.log(b) // "Hello"
```

Javascript - Objects

- an unordered list of primitive data types that is stored as a series of name-value pairs
- mutable
- reference vs value

```
var a = {"value": "Hello!"}
var b = a
a.value = "Hi!"
console.log(a.value) // "Hi!"
console.log(b.value) // "Hi!"
```

Javascript – Objects

- How to create JS objects?

```
var peter = {"name": "Peter", "age": 3}
console.log(peter.name) // Peter
```

Javascript – Objects

- How to create JS objects?

```
function Person(name, age) { //constructor
  this.name = name
  this.age = age
}
var peter = new person("Peter", 30)
console.log(peter.name) // Peter
```

Javascript - Scope

- the context in which the variable exists
- function level scope
- global scope

Javascript - Scope

- innermost scope first

```
var variable = "I am a global variable";
(function() {
  var variable = "I am an inner variable";
  (function() {
    var variable = "I am an inner-inner
variable";
    console.log(variable);
  })();
  console.log(variable);
})();
console.log(variable);
```

Javascript - Scope

- JS doesn't have block level scope (not true in JS6)

```
function f(value) {
  if (value) {
    var x = "Something..."
  }

  console.log(x)
}

f(true) // "Something..."
f(false) // undefined
```

Javascript - Callbacks

```
function downloadResource(completionBlock) {
  // downloading the resource asynchronously
  completionBlock();
}

downloadResource(function () {
  console.log("Download done!")
})
```

Javascript - Classes

```
class Rectangle {
    constructor(height, width) {
        this.height = height;
        this.width = width;
    }

    calcArea() {
        return this.height * this.width;
        }
}

let y = new Rectangle(120, 10)
y.calcArea()
```

Javascript - Classes

```
class Cat {
  constructor(name) {
    this.name = name;
  }
  speak() { console.log(`${this.name} makes a noise.`); }
}

class Lion extends Cat {
  speak() { super.speak(); console.log(`${this.name} roars.`); }
}

let l = new Lion('Fuzzy');
l.speak();
```

Typescript

- superset of Javascript
- pure oriented object
- statically typed
- compiles to plain Javascript

Typescript – Basic types

- Boolean, Number, String, Array, Tuple, Enum, Any, etc.
- The **any** data type is the super type of all types in TypeScript.

```
var isValid: boolean = true;
var greeting: string = "Hello";
var tupleValue: [string, string] =
["Hello", "World"];
enum Color {Red, Green, Blue};
```

Typescript - If

```
if (boolean_expression) {
  // statement(s) will execute if the boolean expression is true
} else {
  // statement(s) will execute if the boolean expression is false
}
```

Typescript - Loops

```
var i: number = 1
while (i<5) {
    console.log("Print " + i)
    i++;
}

do {
    console.log("Print " + i)
    i++;
} while(i<5)
```

Typescript - Loops

```
var i: number = 0;
for(i=0;i<=10;i++) {
  // loop
}
```

Typescript - Functions

```
function hello(name: string):string {
  return "Hello " + name
}

console.log(hello("Ion"))
```

Typescript: Functions

- Optional parameters

```
func hello(name: string, title?:string): string {
  if (title != undefined) {
    return "Hello, " + title + " " + name
  } else {
    return "Hello, " + name
  }
}
```

Typescript - Classes

```
class Person {
 firstName: string
 lastName: string

  constructor(firstName: string, lastName : string) {
    this.firstName = firstName;
    this. lastName = lastName;
  }

  sayHi(): void {
    console.log("Hello, " + this.firstName + " " + this.lastName);
  }
}

var john = new Person("John", "Doe")
john.sayHi()
```

Typescript - Interfaces

```
interface Greetable {
   sayHi(): void
}

class Person {
  firstName: string
  lastName: string

  constructor(firstName: string, lastName: string) {
    this.firstName = firstName;
    this.lastName = lastName;
  }

  sayHi(): void {
    console.log("Hello, " + this.firstName + " " + this.lastName);
  }
}

var person: Greetable = new Person("John", "Doe")
person.sayHi()
```

ES6 Features

- Promises
- Default Parameters
- Template Literals
- Let and Const
- Classes
- Import

ES6 Features - Promises

```
setTimeout(function() {
    console.log('Yay! Timeout expired')
}, 1000)
```

ES6 Features - Promises

```
new Promise(function(resolve, reject) {
    setTimeout(resolve, 1000)
}).then(function() {
    console.log('Yay!')
})
```

ES6 Features - Promises

```
makeRequest(url, function(response) {
    makeRequest(other_url, response.data, function(response) {
        makeRequest(yet_another_url, response.data, function(response) {
            makeRequest(yet_yet_another_url, response.data, function(response) {
                //do something
            }
        }
    }
})
```

ES6 Features - Promises

```
makeRequest(url)
    .then(function(response) {
        return makeRequest(other_url, response.data)
    })
    .then(function(response) {
        return makeRequest(yet_another_url, response.data)
    })
    .then(function(response) {
        return makeRequest(yet_yet_another_url, response.data)
    })
    .then(function(response) {
        //do something
    });
```

ES6 Features – Default Parameters

```
function sendRequest(url, timeout = 10) {

}
```

ES6 Features – Template Literals

```
var name = "Peter"
var age = 10
var details = name + " " + age
var templateLiteralsDetails = `{name} {age}`
```

ES6 Features – let and const

- **let** is used for block scope
- **const** block scoped + immutable

```
function someFunction() {
    var x = 10
    if (x > 0) {
        var x = 12
        console.log(x)
    }

    console.log(x)
}

someFunction()
```

ES6 Features - Classes

```
class Person {
    constructor(firstName, lastName) {
        this.firstName = firstName;
        this.lastName = lastName;
    }

    details() {
        return `{firstName} - {lastName}`
    }
}
```

ES6 Features - Modules

- in modules.js

```
export var activity = 10
export function details() {
    console.log("Details!")
}
```

- in index.js

```
import {activity, details} from 'module'
console.log(activity) // 10
details()
```

or

```
import * as service from 'module'
```

Exercises

- Write a software module that will be used by a library.
  - add books
  - retrieve books
  - query book entries
- Implement the business logic of a WebServer that allow users to sign up, login and show details about their account.