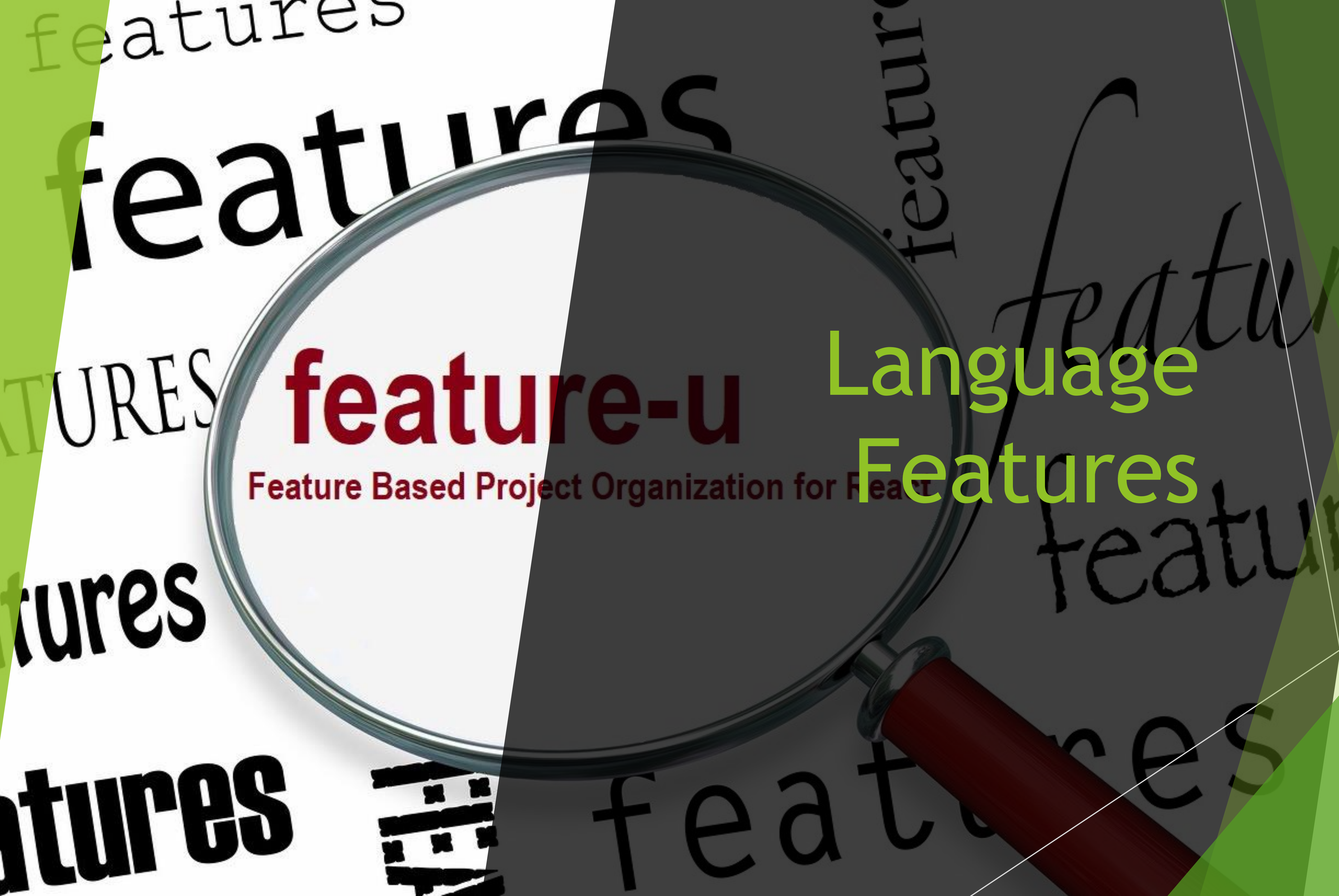
S1 Z3



Language Features

Language Features

- Constants
- Let and Var
- Rest Parameters
- Destructing Array
- Destructing Object
- Spread

Constants

```
const constVar =2;
console.log(constVar);
```

Constants

```
const constVar;
console.log(constVar);
```

Constants

```
const constVar =2;

constVar =3;

console.log(constVar);
```

Constants

```
const cArray = [1, 2, 3];
cArray = [3, 2, 1];
```

Constants

```
const cArray = [1, 2, 3];
cArray.push(4);
console.log(cArray);
```

Constants

```
const objC = { a: 1, b: 2, c: 3 };
objC = {};
```

Let and var

```
console.log(varLet);
let varLet = 'varLet’;

console.log(varVar);
var varVar = 'varVar';
console.log(varVar);
```

Let and var

```
if(true){
   let varLet =1;
}
console.log(varLet);

if(true){
   var varVar =1;
}
console.log(varVar);
```

Let and var

```
if (true) {
    var varVar = 1;
}

console.log(varVar);
varVar = 2;
console.log(varVar);
var varVar = "varVar";
console.log(varVar);
var varVar = "xxx";
console.log(varVar);
```

Rest parameters

```
function ShowData(a,b,...c){
    console.log(a);
    console.log(b);
    console.log(c);
}

ShowData(1,2,3,4,5,6);
ShowData(1);
ShowData(1,2);
ShowData(1,2,3,'four','5',6);
```

Destructing array

```
let ids = [1,2,3,4];
let [id1, id2, id3] = ids;
console.log(id1);
console.log(id2);
console.log(id3);
```

Destructing array

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

let [mainId, ...remainingIds] = ids;

console.log(mainId);
console.log(remainingIds);
```

Destructing array

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

let mainId;
let [, ...remainingIds] = ids;

console.log(mainId);
console.log(remainingIds);
```

Destructing array

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

let [mainId,, ...remainingIds] = ids;

console.log(mainId);
console.log(remainingIds);
```

Destructing objects

```
var person = {
   id : 1,
   name : 'Karol'
}

let { id, name } = person;
console.log(id,name);
```

Destructing objects

```
var person = {
   id : 1,
   name : 'Karol'
}

let id, name;
{id, name} = person;
console.log(id,name);

({id, name} = person);
console.log(id,name);
```

Destructing objects

```
var person = {
   id : 1,
   name : 'Karol'
}

let id, name, year;
({id, name,year} = person);
console.log(id,name,year);
```

Spread

```
function ShowData(a,b){
   console.log(a,b);
}

let values = [1,2];
ShowData(...values);
```

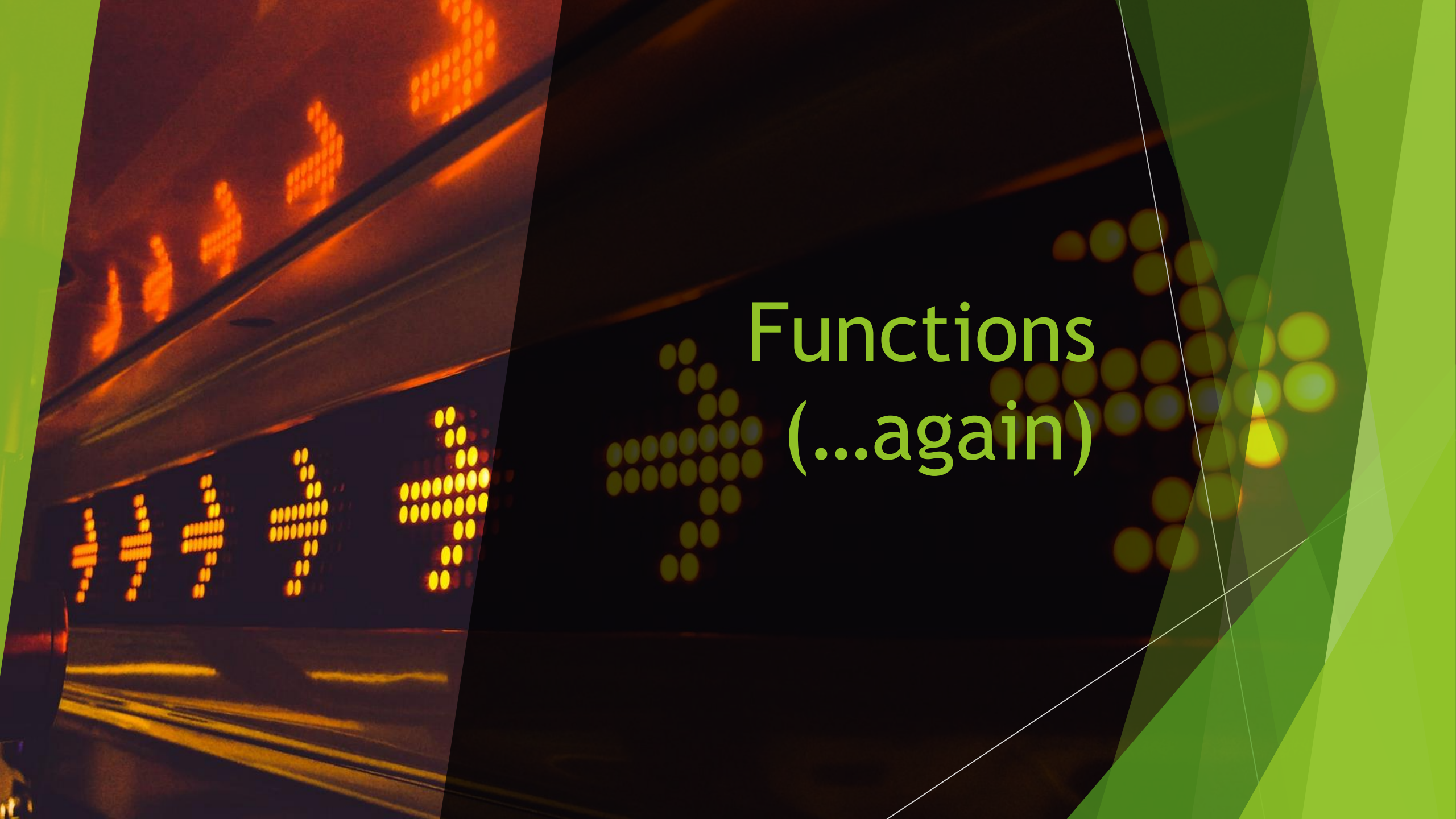
Spread

```
function ShowData(a,b){
   console.log(a,b);
}

let text1 = 'ab';
ShowData(...text1);

let text2 = 'a';
ShowData(...text2);

let text3 = 'abc';
ShowData(...text3);
```



Functions (...again)

Functions (in depth)

- Function Scope
- Block Scope
- IIFE (Immediately Invoked Function Expression)
- Closure
- this
- Call / Apply
- Bind
- Arrow function
- Default values

Function Scope

```
function outerFunction(param1){
   let variable1 = 'variable1';
}

outerFunction('example data');
console.log(variable1);
```

Function Scope

```
function outerFunction(param1){
    let variable1 = 'variable1';
    let innerFunction = function innerFunctionDefinition(){
        console.log(variable1, param1);
    }
    innerFunction();
}

outerFunction('example data');
```

Function Scope

```
function outerFunction(param1){
    let variable1 = 'variable1';
    let innerFunction = function innerFunctionDefinition(){
        let variable1 = 'variable inner version';
        console.log(variable1);
    }
    innerFunction();
    console.log(variable1);
}

outerFunction('example data');
```

Block Scope

```
if(true){
   let var1 = 'var1';
}

console.log(var1);
```

Block Scope

```
let var1 = 'outer vaue'
if(true){
   let var1 = 'inner value';
   console.log(var1);
}

console.log(var1);
```

IIFE

```
function one(){
   console.log('one');
};

(function(){
   console.log('two');
})();

one();
```

IIFE

```
let iife = (function(){
    let var1 = 'iife value';
    console.log(var1);
    return {};
})();

console.log(iife);
```

Closure

```
let iife = (function(){
    let var1 = 'inner';
    let getValue = function(){
        return var1;
    };
    return {
        innerData: getValue
    };
})();

console.log(iife.innerData());
```

this

```
(function(){
   console.log(this);
})();
```

this

```
let obj = {
    id:1,
    getThisId: function(){
        let id =2;
        return this.id;
    },
    getId: function(){
        let id =2;
        return id;
    }
}
```

Call

```
let obj = {
    id:1,
    getId: function(){
        return this.id;
    }
}

let contextObject = {id:2};

console.log(obj.getId());
console.log(obj.getId.call(contextObject));
```

Apply

```
let obj = {
    id:1,
    getId: function(par1, par2){
        return par1+ this.id+par2;
    }
}

let contextObject = {id:2};

console.log(obj.getId('p','s'));
console.log(obj.getId.apply(contextObject,['prefix ',' sufix']));
```

Bind

```
let obj = {
    id:1,
    getId: function(){
        return this.id;
    }
}

let contextObject = {id:2};
let newGetId = obj.getId.bind(contextObject);

console.log(newGetId());
```

Arrow function

```
let fun1 = () => 'fun1';
console.log(fun1());
```

Arrow function

```
let fun2 = prefix => prefix + 'fun1';
console.log(fun2('p'));
```

Arrow function

```
let fun3 = (prefix,sufix) => prefix + 'fun1' + sufix;
console.log(fun3('p','s'));
```

Arrow function

```
let funSum = (x, y)=>{
    let result = x+y;
    return result
};
console.log(funSum(4,7));
```

Default values

```
let showInfo = function(main, prefix='P', sufix = 'S’){
    console.log(prefix, main, sufix);
};

showInfo();
showInfo('example');
showInfo('example','My Prefix');
showInfo('example','My Prefix','My Sufix');
```