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);
```

```
(4) [1, 2, 3, 4]
```

Constants

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

Constants

```
const objC = { a: 1, b: 2, c: 3 };
objC.d = 4;
console.log(objC);  {a: 1, b: 2, c: 3, d: 4}
```

Let and var

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

```
'varLet' was used before it was declared,
which is illegal for 'let' variables.
```

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

```
undefined
```

```
varVar
```

Let and var

```
if(true){
   let varLet =1;
}
console.log(varLet);   ReferenceError: varLet is not defined

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

Let and var

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

console.log(varVar); //1
varVar = 2;
console.log(varVar); //2
var varVar = "varVar";
console.log(varVar); // varVar
var varVar = "xxx";
console.log(varVar); // xxx
```

Rest parameters

```
function ShowData(a, b, ...c) {
  console.log("main data");
  console.log(a);
  console.log("secondary data");
  console.log(b);
  console.log("extra data");
  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); //1
console.log(id2); //2
console.log(id3); //3
console.log(ids); //(4) [1, 2, 3, 4]
```

Destructing array

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

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

console.log(mainId); //1
console.log(remainingIds);//(3) [2, 3, 4]
```

Destructing array

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

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

console.log(mainId); // undefined
console.log(remainingIds); // (3) [2, 3, 4]
```

Destructing array

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

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

console.log(mainId); //1
console.log(remainingIds);// (2) [3, 4]
```

Destructing objects

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

let { id, name } = person;
console.log(id,name); //  1 Karol
```

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);
```

Expected an assignment or function call and instead saw an expression.

//1 Karol

Destructing objects

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

let id, name, year;
({id, name,year} = person);
console.log(id,name,year);// 1 Karol undefined
```

Spread

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

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

Spread

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

let text1 = 'ab';
ShowData(...text1);// a b

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

let text3 = 'abc';
ShowData(...text3);// a b
```

Functions (...again)

Functions (in depth)

- Function 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);// variable1 is not defined
```

Function Scope

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

outerFunction('example data');
```

Function Scope

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

outerFunction('example data');
```

IIFE

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

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

one(); // one
```

IIFE

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

console.log(iife); // {x: 5}
```

Closure

```
let iife = (function() {
  let var1 = "inner";
  let getValue = function() {
    return var1;
  };
  let setValue = function(newValue) {
    if (newValue) var1 = newValue;
  };
  return {
    getInnerData: getValue,
    setInnerData: setValue
  };
})();

console.log(iife);// {getInnerData: ƒ, setInnerData: ƒ}
console.log(iife.getInnerData());  // inner
```

this

```
(function(){
    console.log(this);
})(); //global {DTRACE_NET_SERVER_CONNECTION: ƒ,
DTRACE_NET_STREAM_END: ƒ,
DTRACE_HTTP_SERVER_REQUEST: ƒ,
DTRACE_HTTP_SERVER_RESPONSE: ƒ,
DTRACE_HTTP_CLIENT_REQUEST: ƒ}
```


This is where the fun begins.

this

```
let id = 3;
let obj = {
  id: 1,
  getThisId: function () {
    let id = 2;
    return this.id;
  },
  getId: function () {
    let id = 2;
    return id;
  },
  getOuterId: function () {
    return id;
  },
};

console.log(obj.getThisId()); //1
console.log(obj.getId()); //2
console.log(obj.getOuterId()); //3
```

Call

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

let obj1 = { x: 1 };
let obj2 = {};
let obj3 = { arr: [1, 2, 3] };

thisExample(); //global
thisExample.call(obj1); //{ x: 1 }
thisExample.call(obj2); // {}
thisExample.call(obj3); // { arr: [1, 2, 3] }
```

Call

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

let contextObject = {id:2};

console.log(obj.getId()); //1
console.log(obj.getId.call(contextObject)); //2
```

Apply

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

let contextObject = { id: 2 };

console.log(obj.getId("p", "s")); //p1s
console.log(obj.getId.apply(contextObject, ["apply prefix ", " apply sufix"]));
//apply prefix 2 apply sufix
console.log(obj.getId.call(contextObject, "call prefix ", " call sufix"));
//call prefix 2 call sufix
```

Bind

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

let contextObject = { id: 2 };
let newGetId = obj.getId.bind(contextObject);
console.log(newGetId()); //2
contextObject.id = 3;
console.log(obj.getId()); //1
console.log(newGetId()); //3
```

Arrow function

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

Arrow function

```
let fun2 = prefix => prefix + 'fun1';
console.log(fun2('p’)); //pfun1
```

Arrow function

```
let fun3 = (prefix,sufix) => prefix + 'fun1' + sufix;
console.log(fun3('p','s’)); //pfun1s
```

Arrow function

```
let funSum = (x, y)=>{
    let result = x+y;
    return result
};
console.log(funSum(4,7)); //11
```

Default values

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

showInfo(); //P undefined S
showInfo("example"); //P example S
showInfo("example", null, "My Sufix"); // null example My Sufix
showInfo("example", undefined, "My Sufix"); //P example My Sufix
showInfo("example", "My Prefix", "My Sufix"); //My Prefix example
My Sufix
```