Review: JavaScript Functions & destructuring assignment

Asst.Prof. Dr. Umaporn Supasitthimethee
ผศ.ดร.อุมาพร สุภสิทธิเมธี

Review: Functions

- A function is a **block of JavaScript code that is defined once** but may be **executed**, or invoked, **any number of times**.
- **JavaScript functions are parameterized:** a function definition may include a list of identifiers, known as parameters, that work as local variables for the body of the function.
- If a **function** is assigned to **a property of an object**, it is **known as a method** of that object.
- When a function is invoked on or through an object, that object is the invocation context or `this` value for the function.
- Functions designed to initialize a **newly created object** are called **constructors**.
- In JavaScript, **functions are objects**, and they can be manipulated by programs. JavaScript can **assign functions to variables and pass them to other functions**

Review: Function Declarations

Function declaration:

- the function keyword
- the name of the
- a list of parameters to the function
- the JavaScript statements that define the function, enclosed in curly brackets, {...}.

```
function name([param1[, param2[, ..., paramN]]]) {
               statements

}
```

Review: Function Expressions

Function expressions look a lot like function declarations, but they appear within the context of a larger expression or statement, and *the name is optional*. However, a name can be provided with a function expression.

Function expression: function expression defines a function and assign it to a variable

```
//functions as Values
const getRectangleArea = function(width, height) {
          return width * height;
};
```

Named function expression : Function expressions can include names, which is useful for recursion.

```
//functions as Values
let fact = function factorial(n) {
                  console.log(n) ;
                  if (n <= 1) {
                    return 1;
                  }
                 return n * factorial(n - 1);
             } ;
fact (5); //120
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/function

Review: Calling Functions

- *Defining* a function does not *execute* it. Defining it names the function and specifies what to do when the function is called.
- Calling the function actually performs the specified actions with the indicated parameters.

//06_Functions/script1.js

```
//function declaration
function square (side) {
        return side * side;
}

square(3); //calling functions
```

```
//function expression
let area=function square(side){
        return side* side;
}
area(3); //calling functions
```

Functions must be *in scope* when they are called, but the function declaration can be hoisted:

```
square(3); //hoisting

function square (side) {
        return side * side;
}
```

function hoisting only works with function *declarations* —not with function *expressions*.

Review: Arrow Function Expressions

Arrow Function expression is a compact alternative to a traditional function expression but is limited and can't be used in all situations.

```
    // Traditional Function (no arguments)
    let a = 4;
    let b = 2;
    function (){
        return a + b + 100;
    }

    // Arrow Function
    let a = 4;
    let b = 2;
    () => a + b + 100;
```

```
//No param
 () => expression

// One param
 param => expression

// Multiple param
 (param1, paramN) => expression

// Multiline statements
 param1 => {
        statement1;
        …
        statementN;
 }

 (param1, paramN) => {
        statement1;
        …
        statementN;
 }
```

Review: Comparing traditional functions to arrow functions

```
// Traditional Function (one argument)
function  (a){
    return a + 100;
}
// Arrow Function Break Down
// 1. Remove the word "function" and place arrow
between the argument and opening body bracket
(a)=> {
    return a + 100;
}

// 2. Remove the body braces and word "return" --
the return is implied.
(a)=> a + 100;

// 3. Remove the argument parentheses
a => a + 100;
```

```
// Traditional Function (multiple arguments)
function (a, b){
        return a + b + 100;
}

// Arrow Function
(a, b) => a + b + 100;
```

```
// Traditional Function (multiline statements)
function (a, b){
        let chuck=42;
        return a + b + chuck;
}

// Arrow Function
(a, b) => {
        let chuck=42;
        return a + b + chuck;
}
```

Review: destructuring assignment

- The destructuring assignment syntax is a JavaScript expression that makes it possible to **unpack values from arrays, or properties from objects, into distinct variables.**

```
let a, b, rest;
[a, b] = [5, 10];

console.log(a); //5
console.log(b); //10

[a, b, ...rest] = [5, 10, 15, 20, 25];
console.log(rest); // [15,20,25]
```

```
({ a, b } = { a: 10, b: 20 });
console.log(a); // 10
console.log(b); // 20

({ a, b, ...rest } = { a: 10, b: 20, c: 30, d: 40 });
console.log(a); // 10
console.log(b); // 20
console.log(rest); // {c: 30, d: 40}
```