

JavaScript discussion

TA Weijun Li

Agenda

- JavaScript variables
- Hoisting
- Callback Functions
 - Coding Exercise
- Array Destructuring
- Object Destructuring
 - Coding Exercise

Online Tools for This Discussion <https://codepen.io/>

Demo Example: <https://codepen.io/Weijun-Li-the-selector/pen/zxOmzad?editors=1010>

Javascript types of “variables”

- Var:
 - function scoping
 - If not in a function, it is global scoped
- Let:
 - block scoping
- Const:
 - can't change value

Javascript types of “variables”

Var:

```
function discountPrices (prices, discount) {  
  var discounted = []  
  for (var i = 0; i < prices.length; i++) {  
    var discountedPrice = prices[i] * (1 - discount)  
    var finalPrice = Math.round(discountedPrice * 100) / 100  
    discounted.push(finalPrice)  
  }  
  console.log(i) // 3  
  console.log(discountedPrice) // 150  
  console.log(finalPrice) // 150  
  return discounted  
}
```

```
discountPrices ([100, 200, 300], .5)
```

<https://tylermcginnis.com/var-let-const/>

Javascript types of “variables”

Const:

```
const PI = 3.142;
```

```
PI = 22/7;
```

```
console.log(PI); // Output: TypeError: Assignment to constant  
variable.
```

Javascript types of “variables”

Let:

```
function discountPrices (prices, discount) {  
    let discounted = []  
  
    for (let i = 0; i < prices.length; i++) {  
        let discountedPrice = prices[i] * (1 - discount)  
        let finalPrice = Math.round(discountedPrice * 100) / 100  
        discounted.push(finalPrice)  
    }  
    console.log(i) // ?  
    console.log(discountedPrice) // ?  
    console.log(finalPrice) // ?  
    return discounted  
}
```

Variable Hoisting

Hoisting: move variable declaration to the beginning. (To the top of the current scope)

Variable and **function** declarations get hoisted before the rest of the code!

Declaration != than initialization!


`Var thing;` -> Declaration, var is undefined.

`thing = "some thing";` -> Initialization

Variable Hoisting - Var

```
function discountPrices (prices, discount) {  
  console.log(discounted)  // undefined, because discounted is hoisted... no error  
  var discounted = []  
  for (var i = 0; i < prices.length; i++) {  
    var discountedPrice = prices[i] * (1 - discount)  
    var finalPrice = Math.round(discountedPrice * 100) / 100  
    discounted.push(finalPrice)  
  }  
  console.log(i) // 3  
  console.log(discountedPrice) // 150  
  console.log(finalPrice) // 150  
  
  return discounted  
}
```


Variable Hoisting - Let

```
function discountPrices (prices, discount) {  
  console.log(discounted)  //  ReferenceError  
  let discounted = []  
  for (let i = 0; i < prices.length; i++) {  
    let discountedPrice = prices[i] * (1 - discount)  
    let finalPrice = Math.round(discountedPrice * 100) / 100  
    discounted.push(finalPrice)  
  }  
  console.log(i) // 3  
  console.log(discountedPrice) // 150  
  console.log(finalPrice) // 150  
  return discounted  
}
```

Function Hoisting

- Functions are hoisted

```
catName("Chloe");  
function catName(name){  
  console.log("My cat's name is " + name);  
}  
/* The result of the code above is: "My cat's name is Chloe" */
```

Function Hoisting

- Function expression are **not** hoisted

```
expression(); //Output: "TypeError: expression is
not a function
catName("Chloe"); // works
function catName(name){
    console.log("My cat's name is " + name);
}
var expression = catName
```

Javascript types of “variables”

Variables in Demo

```
// Select the input field where new tasks are entered
const newTask = document.querySelector("#new-task-input");
// const is used for variables that will not be reassigned (e.g., DOM elements)
const addTaskBtn = document.querySelector(".add-task-btn");
const removeCompleteBtn = document.querySelector(".remove-complete-btn");
const taskList = document.querySelector(".task-list");
const taskTemplate = document.querySelector("#task-template");

// Initialize a unique ID for each task
let id = 1;
// let is used for mutable variables, as id will increment each time a task is added
```

Callback Functions

- Function Recap

- `function` name(param1, param2) {
 }
- Parameters are optional
- All functions are object methods ()
- Functions are **objects**
 - `var` student = {
 firstName: "Peter",
 lastName: "Anteater",
 printName: function() {
 console.log(`this`.firstName + " " + `this`.lastName);
 }
}
 - `student.printName();` // Invokes the function

Callback Functions

- Callback functions: a function that is passed to another function to be executed

```
function addTwoNumbers(a, b, callback) {  
    console.log("Adding two numbers..");  
    var result = a + b;  
    callback(a,b,result);  
}  
  
function printResults(x,y,z) {  
    console.log(x + " + " + y + " = " + z);  
}  
  
addTwoNumbers(10,15,printResults);
```

Callback Functions

Callback Functions in Demo

```
1 // Add an event listener to the input field for the "Enter" key press (callback function example)
2 newTask.addEventListener("keyup", (e) => {
3     // Event listeners use callback functions that execute when the event occurs
4     if (e.keyCode === 13 && inputValid()) {
5         addTask(); // Call the function to add a new task
6     }
7 });
8
9 // Add an event listener to the "Add Task" button (callback function example)
10 addTaskBtn.addEventListener("click", () => {
11     // If the input is valid, call addTask() to add the new task
12     if (inputValid()) {
13         addTask();
14     }
15 });
```

Array callback

- Reminder of `forEach()`

```
const array1 = ['a', 'b', 'c'];  
  
array1.forEach(element => console.log(element));  
  
// expected output: "a"  
// expected output: "b"  
// expected output: "c"
```


Array callback

- Reminder of `reduce()`

```
const array1 = [1, 2, 3, 4];
```

```
// 1 + 2 + 3 + 4
```

```
var total = array1.reduce((accumulator, currentValue) => {  
  return accumulator + currentValue;  
}, 0)
```

```
console.log(total) // expected output: 10
```

Array callback

Array Callback in Demo

```
6
7 removeCompleteBtn.addEventListener("click", () => {
8   // Select all task elements, generating a NodeList (similar to an array)
9   const tasks = document.querySelectorAll(".task");
10
11   // forEach is an array callback method that iterates over each task
12   tasks.forEach((task) => {
13     // Check if the task's checkbox is checked
14     const checked = task.querySelector("input").checked;
15
16     // If the checkbox is checked, remove the task from the list
17     if (checked) {
18       task.remove();
19     }
20   });
21 });
```

Callback Exercise

- Calculate the average of an array of numbers
 - First by using `forEach()`
 - Then by using `reduce()`
- Pastebin: <https://pastebin.com/XJevk7dT>

```
var myArray = [6, 2, 8, 1, 2, 5, 3];  
console.log('Array is ' + myArray);
```

```
/*TODO:
```

```
- Use forEach() to calculate the average of all numbers in an array
```

```
- Use reduce() to calculate the average of all numbers in an array
```

```
*/
```

Callback Exercise (Solutions)

```
var myArray = [6, 2, 8, 1, 2, 5, 3];  
console.log('Array is ' + myArray);
```

```
var total = 0;  
myArray.forEach(function(item) {  
    total += item;  
});
```

```
var avg = total / myArray.length;  
console.log("Avg: " + avg);
```

```
var myArray = [6, 2, 8, 1, 2, 5, 3];  
console.log('Array is ' + myArray);
```

```
Var total = myArray.reduce(function(sum, current)  
{  
    return sum + current;  
}, 0);
```

```
var avg = total / myArray.length;  
console.log("Avg: " + avg);
```

Object and Array Destructuring

- Main Purpose: Unpack values from arrays (or objects) into distinct variables
- Array Destructuring

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

```
Let [a, b, ...rest] = [10, 20, 30, 40, 50];  
console.log(a); // 10  
console.log(b); // 20  
console.log(rest); // [30, 40, 50]
```

Object and Array Destructuring

- Default values for assignees

```
var a, b;  
  
[a=5, b=7] = [1];  
console.log(a); // 1  
console.log(b); // 7
```

- Swapping variables

```
var a = 1;  
var b = 3;  
  
[a, b] = [b, a];  
console.log(a); // 3  
console.log(b); // 1
```

Object and Array Destructuring

- Object Destructuring

```
var student = {id: 88888888, firstName: "Peter", lastName: "Anteater"};  
var {id: a, firstName: b, lastName: c} = student;  
  
console.log(a); // 88888888  
console.log(b); // Peter  
console.log(c); // Anteater
```

Object and Array Destructuring

- Object destructuring when passed as a parameter

```
var student = { id: 88888888, firstName: "Peter", lastName: "Anteater" };

function studentID({ id }) {
    return id;
}

function fullName({ firstName, lastName }) {
    return "Fullname is " + firstName + " " + lastName;
}

console.log(studentID(student)); // 88888888
console.log(fullName(student)); // "Peter Anteater"
```


Object and Array Destructuring

- Object destructuring when passed as a parameter (with default values)

```
var student = { firstName: "Peter", lastName: "Anteater" };

function studentID({ id = 11111111 }) {
    return id;
}

function fullName({ firstName = "default", lastName = "default" }) {
    return "Fullname is " + firstName + " " + lastName;
}

console.log(studentID(student)); // 11111111
console.log(fullName(student));  // "Peter Anteater"
```

Object and Array Destructuring

in Demo

```
36
37 removeCompleteBtn.addEventListener("click", () => {
38   // Select all task elements, generating a NodeList (similar to an array)
39   const tasks = document.querySelectorAll(".task");
40
41   // The forEach method is an array callback function that iterates through each task
42   tasks.forEach((task) => {
43     // Object destructuring is used here to extract the "checked" property from the input element
44     const { checked } = task.querySelector("input");
45
46     // If the checkbox is checked, remove the task from the list
47     if (checked) {
48       task.remove();
49     }
50   });
51 });
```

Destructuring Exercise

- Create a function that receives the following object as parameter and prints a phrase with the name of the book and its publisher (e.g. “ The <book title here> was published by <publisher’s name here>”)

```
let book = {  
  title: 'The Fellowship of the Ring',  
  publisher: 'Allen & Unwin'  
};
```

Destructuring Exercise (Solution)

```
let book = {  
  title: 'The Fellowship of the Ring',  
  publisher: 'Allen & Unwin'  
};
```

```
function print({title, publisher}){  
  return title + ', published by ' + publisher;  
}  
  
console.log(print(book));
```

Destructuring Exercise

- Use `forEach()`
- Given a list of objects, write a callback function that gets the maximum score and the person's name who has the highest score

Download code here:

<https://pastebin.com/riJiKWvN>

```
var myList = [  
  {name: "Peter 1", score: 79},  
  {name: "Peter 2", score: 91},  
  {name: "Peter 3", score: 66},  
  {name: "Peter 4", score: 99},  
  {name: "Peter 5", score: 55}  
];  
  
/* TODO */  
var maxName;  
var max;  
  
console.log(maxName + " has the highest score: " + max);
```

Destructuring Exercise

- Solution

```
var myList = [  
  {name: "Peter 1", score: 79},  
  {name: "Peter 2", score: 91},  
  {name: "Peter 3", score: 66},  
  {name: "Peter 4", score: 99},  
  {name: "Peter 5", score: 55}  
];  
  
var max = Number.NEGATIVE_INFINITY;  
var maxName = "None"  
  
myList.forEach(function({name, score}) {  
  if (score > max) {  
    max = score;  
    maxName = name;  
  }  
});  
  
console.log(maxName + " has the highest score: " + max);
```

Destructuring Exercise

- Solution, now with arrow function

```
var myList = [  
  {name: "Peter 1", score: 79},  
  {name: "Peter 2", score: 91},  
  {name: "Peter 3", score: 66},  
  {name: "Peter 4", score: 99},  
  {name: "Peter 5", score: 55}  
];  
  
var max = Number.NEGATIVE_INFINITY;  
var maxName = "None"  
  
myList.forEach(({name, score}) => {  
  if (score > max) {  
    max = score;  
    maxName = name;  
  }  
});  
  
console.log(maxName + " has the highest score: " + max);
```