

INT201

Javascript

โดย

พี่เจมส์

ปัญหาที่พบเจอบ่อยใน Module 1

การใช้งาน if

1. `if ((str = "")) {..}`
2. `if (str = null) {..}`
3. `if(str, 'uppercase'){..}`
4. `if (typeof str !== String) {..}`
 - ใช้ `==` หรือ `===` สำหรับการเปรียบเทียบค่า
 - ใช้ `typeof` อย่างถูกต้องสำหรับการตรวจสอบประเภทข้อมูล โดยผลลัพธ์จะเป็น string เช่น "string", "number"
 - หลีกเลี่ยงการใช้เครื่องหมาย `==` สำหรับการตรวจสอบเพราะเป็นการกำหนดค่า
5. `if (!typeof str === "string") {..}`
 - การใช้ `!typeof` ทำให้การประเมินเงื่อนไขเป็นการดำเนินการ ! ก่อน แล้วจึงเปรียบเทียบกับ "string" ซึ่งผลลัพธ์จะไม่ถูกต้องตามที่ต้องการเทียบ

การใช้งาน String

การใช้งาน String

1. `String.prototype.toLowerCase(str);`

2. `str.toLowerCase(str);`

3. `str.toLocaleLowerCase(str);`

4. `str.toupperCasa();`

- การเรียกใช้งานไม่ถูกต้องแนะนำให้นักศึกษาควรตรวจสอบ Method ที่จะใช้งานก่อนทำการเรียกใช้เพื่อป้องกันความผิดพลาดจากการ

การตั้ง conditions

นอกจาก 2 เรื่องข้างต้นที่พบเจอบ่อยจะเป็นปัญหาในการลำดับความสัมพันธ์ของ
การใช้งานนักศึกษาควรตรวจสอบเงื่อนไขการทำงานของโปรแกรมตามลำดับเพราะ
ไม่เช่นนั้นโปรแกรมอาจเสร็จสิ้นก่อนที่เรากำหนดอาทิตัวอย่างเช่น

```
if (formatType != "uppercase" || formatType != "lowercase"){throw  
new Error("...")}
```

...

.

หาก formatType เป็น "uppercase", การตรวจสอบว่า formatType !=
"lowercase" จะเป็นจริง

หาก formatType เป็น "lowercase", การตรวจสอบว่า formatType !=
"uppercase" ก็จะเป็นจริงเช่นกัน

ดังนั้นโปรแกรมจะสิ้นสุดการทำงานทันที

practice

1. **Lowercase Conversion** (`toLowerCase`): If the `type` is `'lower'`, the string will be converted to lowercase.
2. **Uppercase Conversion** (`toUpperCase`): If the `type` is `'upper'`, the string will be converted to uppercase.
3. **Default Case**: If neither `'lower'` nor `'upper'` is passed, the string is returned unchanged.

```
console.log(formatString('eiei', 'lower')); // 'eiei'  
console.log(formatString('Test', 'upper')); // 'TEST'
```

practice

test-totalPages

ให้เขียน Function ชื่อ totalPages (arrayItems, rowsPerPage) โดยที่ arrayItems คือ array ของรายการ Items ทั้งหมด และ rowsPerPage คือ จำนวนรายการ Item ที่จะแสดงต่อ 1 Page โดยฟังก์ชันจะ return จำนวน page ที่คำนวณได้

- กรณี arrayItems เป็น null หรือ undefined ให้ return undefined
- กรณี rowsPerPage เป็น null หรือ undefined หรือเป็น 0 จะ return 1
- ตัวอย่างเช่น
 - i. arrayItems = [item1, item2, item3,..., item35] , rowsPerPage = 15 return 3
 - ii. arrayItems = [item1, item2, item3,..., item10] , rowsPerPage = 20 return 1
 - iii. arrayItems = [item1, item2, item3,..., item40] , rowsPerPage = 10 return 4
 - iv. arrayItems = [item1, item2, item3,..., item25] , rowsPerPage = 0 return 1
 - v. arrayItems = [item1, item2, item3,..., item25] , rowsPerPage = null return 1
 - vi. arrayItems = [item1, item2, item3,..., item25] , rowsPerPage = undefined return 1
 - vii. arrayItems = null, rowsPerPage = 20 return undefined
 - viii. arrayItems = undefined, rowsPerPage = 20 return undefined

practice

Name of Practice: Password Validator

Instruction:

Write a function `validatePassword` that accepts a password string and checks whether it meets the following criteria:

1. It must be at least 8 characters long.
2. It must contain at least one uppercase letter.
3. It must contain at least one lowercase letter.
4. It must contain at least one digit.
5. It must contain at least one special character from the set: `@, #, $, %, ^, &, *, !`.

If the password satisfies **all** of the conditions, return `true`. Otherwise, return `false`. If the input is `null`, `undefined`, or an empty string, return `false` as well.

```
validatePassword("Zky543#@");    // Output: true
validatePassword("1234abcd");    // Output: false
validatePassword("12345abcxyZ$");// Output: true
```


practice

Name of Practice: Find first and last occurrence of a character

Instruction:

Write a function `findFirstAndLast` that accepts a string and a character. The function should return an object containing:

- `firstIndex`: The index of the first occurrence of the character in the string.
- `lastIndex`: The index of the last occurrence of the character in the string. Return `-1` for both if the character is not found.

```
console.log(findFirstAndLast('javascript', 'a')) // output: { firstIndex: 1, lastIndex: 3 }
```

practice

Description: Create a library management system with books. Each book should have a `title`, `author`, and `isAvailable` (boolean). Write functions to:

- Add a new book.
- Borrow a book by title (mark it as not available).
- Return a book by title (mark it as available).
- List all available books.

```
addBook('The Great Gatsby', 'F. Scott Fitzgerald')
addBook('1984', 'George Orwell')
borrowBook('1984')
listAvailableBooks() // Expected output: List of available books except "1984"
returnBook('1984')
listAvailableBooks() // Expected output: All books including "1984"
```

Practice: Car Inventory Management System

Instructions:

1. Write your student id, firstname, and lastname in a single line comment before starting your program. Students who do not put this comment will get 50% taken off their score.
2. Implement a class called `CarInventory` that manages a collection of cars with the following requirements:

Properties:

- `cars` (an array of car objects): Each car object should have the following properties:
 - `make`: A string representing the make of the car.
 - `model`: A string representing the model of the car.
 - `year`: An integer representing the year the car was manufactured.
 - `color`: A string representing the color of the car.

Tasks:

1. Complete the functions in the `CarInventory` class:
 - `addCar`: Add a new car, checking for duplicates and missing details.
 - `getAllCars`: List all the cars.
 - `findCarsByColor`: Return cars matching the given color.
 - `updateCar`: Update the car's details.
 - `deleteCar`: Delete a car based on its make and model.
2. Test the code by adding cars, updating cars, finding cars by color, and deleting cars.

Challenges:

- Make sure the `addCar` function handles missing details and prevents duplicate entries.
- Allow the `updateCar` function to update any combination of details (year, color).
- The `findCarsByColor` function should handle case-insensitive matching for colors.

Functions:

1. `constructor()`:
 - Initializes the `cars` array as an empty array (`[]`).
2. `addCar(make, model, year, color)`:
 - Adds a new car to the `cars` array with the provided details (`make`, `model`, `year`, `color`).
 - If any of the details are missing, or a car with the same make and model (case-insensitive) already exists, return `undefined`.
 - Otherwise, return the new car object.
3. `getAllCars()`:
 - Returns the full array of cars currently in the inventory.
4. `findCarsByColor(color)`:
 - Returns an array of cars that match the given color (case-insensitive). If no cars match, return an empty array.
5. `updateCar(make, model, updatedDetails)`:
 - Updates the details of a car by its make and model (case-insensitive). The user can update one or more properties (`year`, `color`).
 - If no car is found with the given make and model, return `undefined`. Otherwise, return the updated car object.
6. `deleteCar(make, model)`:
 - Deletes a car based on its make and model (case-insensitive).
 - If the car is found and deleted, return the deleted car object. If no car is found, return `undefined`.

จบแล้วจ้า