CAT Reloaded

Back-End Circle

NodeJS Beginner - Week 9

* Indicates required question

Section 1

Recap

1 - What is the purpose of a default case in a switch statement? * 1 point

- [x] To provide a fallback option when none of the other cases match
- [ ] To terminate the execution of the switch statement
- [ ] To provide a condition for the switch statement
- [ ] To specify the type of the expression

2 - Choose the type of "this" binding that occurs in line 10 * 1 point

index.js

```
1  function Counter(init) {
2      this.x = init;
3  }
4
5  Counter.prototype.inc = function() {
6      this.x++;
7  }
8
9  let c = new Counter(0);
10 c.inc();
```

- [ ] Default binding
- [x] Implicit binding
- [ ] Explicit binding
- [ ] none of the above

3 - Waleed has the Imparnumerophobia, which means he fears the odd numbers. Waleed asked you to design a program for him that modifies the strings by replacing any odd number with the word **"BEEP"**. Help him implement this program using your knowledge in regular expressions. * 3 points

Example:
"I have 12 cars, 11 of which are 89 years old"
Should be
"I have 12 cars, BEEP of which are BEEP years old"

Put your solution on https://ideone.com/ or codepen.io and paste the link below.

https://ideone.com/EyjTwh

Back Next Clear form

CAT Reloaded

Back End Circle

NodeJS Beginner - Week 9

abdalmoniemshagar@gmail.com Switch account

Your email will be recorded when you submit this form

* Indicates required question

Section 2

4 - Choose all the correct statements about JSON. * 1 point

- [x] JSON data is represented as string; so you need to convert it to JS object to work with it
- [ ] JSON is a programming language
- [ ] JSON is can only be used inside JavaScript
- [x] JSON is used to exchange data across different systems

5 - Which method is used to convert data from JS object to JSON / from JSON to JavaScript object? * 1 point

- ( ) json() / object()
- (•) JSON.stringify() / JSON.parse()
- ( ) JSON.parse() / JSON.stringify()
- ( ) JSON.encode() / JSON.decode()

6 - The fetch API is based on promises * 1 point

- (•) True
- ( ) False

7 - Explain why response.json() is preceded by the "await" keyword * 1 point

```
let get = async (url) => {
    let response = await fetch(url);
    let data = await response.json();
    return data;
}
```

The await keyword is used to "Wait" until the promise is either resolved, or rejected.

8 - Choose the correct category for each of the following JS component * 5 points

	Core JavaScript	WebAPIs
setTimeout / setInterval	( )	(•)
Arrays	(•)	( )
console	(•)	( )
Promises	( )	(•)
fetch api	( )	(•)

9 - Before proceeding, read Star Wars API Documentation * 1 point
What is the convenient URL for fetching all Star Wars planets?

- ( ) https://swapi.dev/api/
- (•) https://swapi.dev/api/planets/
- ( ) https://swapi.dev/api/planets/schema/
- ( ) https://swapi.dev/api/planets/1/

10 - Using Star Wars API Documentation * 1 point
what is the output of this code?

```
fetch('https://swapi.dev/api/people/7/')
.then(res => res.json())
.then(data => console.log(data.hair_color))
```

- ( ) blond
- ( ) light
- (•) brown
- ( ) black

11 - What is the return type of this function? * 1 point

```
const fun = async () => {
    return "Hey there!";
}
```

- ( ) string
- ( ) object
- (•) promise
- ( ) undefined

12 - Choose the method matching the following description: * 1 point

It takes a list of promises and return a single promise
the returned promise fulfills only when all input promises fulfull, it reject with the first rejection.

- (•) Promise.all()
- ( ) Promise.any()
- ( ) Promise.allSettled()
- ( ) Promise.race()

13 - Choose all answers that apply: * 1 point

Which of the following are best practices to avoid JavaScript callback hell?

- [x] Using promise chaining
- [ ] Skipping the implementation of error handlers
- [x] Using async/await syntax for asynchronous callbacks
- [ ] Using setTimeout

14 - The code inside the "finally" block runs only when some error happens * 1 point

- ( ) True
- (•) False

Back Next Clear form

CAT Reloaded

Back-End Circle

NodeJS Beginner - Week 9

* Indicates required question

Coding Task

15 - Solve the following 4 problems and paste the link of your solution. * 5 points

1. Add Two Promises
2. Sleep
3. Timeout Cancellation
4. Interval Cancellation

Bonus ✨
Promise Time Limit

https://leetcode.com/problems/add-two-promises/submissions/1622407278
https://leetcode.com/problems/sleep/submissions/1622775535
https://leetcode.com/problems/timeout-cancellation/submissions/1622923043
https://leetcode.com/problems/interval-cancellation/submissions/1623091718
https://leetcode.com/problems/promise-time-limit/submissions/1623187220

16 - One day Seif visited the fake store to buy some fake products. * 5 points

Seif bought 3 items of the product with id 1,
4 items of the product with id 4
and 5 items of the product with id 3

Seif is bad at math, can you help him calculate the total price of the products?

Use the Fake Store API to solve the problem and upload your solution to GitHub.

```
const purchases = {
   1: 3,
   4: 4,
   3: 5
}

// first solution
// let sum = 0;
// for(let [id, quantity] of Object.entries(purchases)){
//    await fetch(`https://fakestoreapi.com/products/${id}`)
//    .then(response => response.json())
//    .then(products => {
//       sum += products.price * quantity;
//    })
//    .catch(err => console.log(err));
// }
// console.log(sum);

// second solution
// let sum = 0;
// fetch(`https://fakestoreapi.com/products/`)
// .then(response => response.json())
// .then(products => {
//    for(let product of products){
//       if(purchases.hasOwnProperty(product.id))
//          sum += product.price * purchases[product.id];
//    }
// })
// .catch(err => console.log(err))
// .finally(() => console.log(sum));

// third solution
async function calculateTotal() {
   const res = await fetch("https://fakestoreapi.com/products/");
   if(!res.ok){
      throw new Error(`Failed to fetch products: ${res.status} ${res.statusText}`);
   }
   const products = await res.json();

   const wanted = products.filter(p => Object.hasOwn(purchases, p.id))

   const total = wanted.reduce((sum, p) => sum + p.price * purchases[p.id], 0);
   console.log(total);
}

calculateTotal()
.catch((err) => console.log(err));
```

Back Submit Clear form