Programming for the Web (CS-544-01)
Homework – 3

Name – Chaitanya Kulkarni **B-Number: B00814455**

Ans 1 –

Replacing all occurrences of anonymous functions which use the function keyword with fat-arrow functions can be problematic because –

- The `this` keyword is bound to values based on the *context* in which it is called. Hence, `this` is lexically bound when using fat-arrow functions. It means that it uses `this` from the code that contains the arrow function i.e. this keyword will refer to the parent context/scope in which the arrow function is declared.
- Another case where fat-arrow functions are problematic is when the function is constructor. If a function is constructible, it can be called with new, i.e. new FunctionName(). If a function is callable, it can be called without new. And Arrow functions are only callable i.e arrow functions can never be used as constructor functions. Hence, they can never be invoked with the new keyword.

Ans 2 –

- As we know that pending events and loops will block program exit in nodeJs hence the program exits only when these events and loops are executed. The given code seems to work without any top-level `await` because nodeJs program is waiting for events and loops to finish first.
- To explicitly terminate the program, we can exit nodeJs program using `process.exit()` function call which terminates the program with integer exit code.
- One of the drawbacks of not using top-level `await` is we can access the await part prior to the promise getting resolved.

Ans 3 –

- To maintain an audit log whenever the `obj.secure` property is used, we can use Accessor descriptors getter and setter methods in the object that will read and write the every use of that property into the log
- Getter and Setter methods are useful for doing things behind-the-scenes
- The getter method will run when `obj.secure` property is read and write the data into audit log
- We can use the same as below

```
obj = {
     secure : { },  //Security Information
     get audit() {
          console.log("Obj.secure is used");
          // write into audit log.
     }
}
```

Ans 4 –

a) A service for listing the grades for all students for a particular assignment.

- GET method
- URL = /assignment?assignmentId=1
- Input parameters = Here we can provide assignmentId as the query parameter for which the grades are to be shown
- Output = Object showing the assignment info along with the list of objects of grades for all students for that particular assignment

```
{
     "assignmentId" : "1",
     "students" : [{
          "studentId" : "B78453432",
```

```
                "grade" : "100"
            }, {
                "studentId" : "B78565676",
                "grade" : "98"
            }]
        }
```

- Links for implementing HATEOAS
    self, next assignment, prev assignment

b) A service for listing all the grades for all assignments for a particular student.

- GET method
- URL = /student?studentId=B00814455
- Input parameters = Here we can provide studentId as the query parameter for which the grades for all assignments are to be shown for that student
- Output = Object showing the student info along with the list of objects of grades for all assignments for that particular student

```
        {
            "studentId" : "B00814455",
            "name" : "Chaitanya",
            "assignment" : [{
                "assignmentId" : "1",
                "grade" : "100"
            }, {
                "assignmentId" : "2",
                "grade" : "98"
            }]
        }
```

- Links for implementing HATEOAS
    - self, next student, prev student

c) A service for listing a matrix of grades for specific students and assignments.

- GET method
- URL = /student?studentId=B00814455,B78453432,B78565676&assignmentId=1,2,3
- Input parameters = Here we can pass multiple studentIds and assignmentIds comma seperated as query parameters. These parameters can then be separated and stored into query object by splitting it by comma (,)
- Output = //object of list of lists (Matrix) of grades for specific students for specific assignments

```
        {
            [{
                "studentId" : "B00814455",
                "name" : "Chaitanya",
                "assignment" : [{
                    "assignmentId" : "1",
                    "grade" : "100"
                }, {
                    "assignmentId" : "2",
                    "grade" : "98"
                }, {
                    "assignmentId" : "3",
                    "grade" : "90"
                }]},{
```

```
                "studentId" : "B78453432",
                "name" : "Jack",
                "assignment" : [{
                     "assignmentId" : "1",
                     "grade" : "90"
                }, {
                     "assignmentId" : "2",
                     "grade" : "80"
                }, {
                     "assignmentId" : "3",
                     "grade" : "95"
                }]
           }]
     }
```

- Links for implementing HATEOAS
    - Self

d) A service to obtain details about a student, like their ID number, email address, etc.

- GET method
- URL = /student/studentId
- Input parameters = There will be no input parameters since we can directly provide studentId in the URL itself
- Output = // Student Information by studentId

```
{
     "studentId" : "B00814455",
     "Name" : "Chaitanya",
     "email" : "ckulkar2@binghamton.edu"
}
```

- Links for implementing HATEOAS
    - self, delete student, next student, prev student, last student, first student

e) A service to obtain details about an assignment, like the date assigned, the date it was due, etc.

- GET method
- URL = /assignment/assignmentId
- Input parameters = There will be no input parameters since we can directly provide assignmentId in the URL itself
- Output = // Assignment Information by assignmentId

```
{
     "assignmentId" : "1",
     "dateAssigned" : "1 April 2020",
     "dateDue" : "10 April 2020",
     "totalMarks" : "100"
}
```

- Links for implementing HATEOAS
    - self, delete assignment, next assignment, prev assignment, last assignment, first assignment

f) A service for adding a new assignment to the course along with the grades obtained by the students in that assignment.

- POST method
- URL = /assignment

- Input body parameters

```
{
    "assignmentId" : "6",
    "studentId" : "B00814455",
    "grade" : "100"
}
```

- Output = Empty JSON object

```
{}
```

g) A service updating an individual grade for a particular student and assignment.

- PATCH method
- URL = /student/studentId?assignmentId=2
- Input query parameters = We can provide assignmentId as the query parameter to the url along with the input body parameters specifying the fields to be updated
- Input body parameters

```
{
    "grade" : "90"
}
```

- Output = Empty JSON object

```
{}
```

Ans 5 –

According to the definition, **H**ypermedia **A**s **T**he **E**ngine **O**f **A**pplication **S**tate is constraint of the REST application architecture that keeps the RESTful style architecture unique from most other network application architectures. It lets you use hypermedia links in the response contents so that the client can dynamically navigate to the appropriate resource by traversing the hypermedia links

- Which means, if client needs a related resource, even if it's the next page of results, the **API should guide the client** to it. This is represented by using Hypermedia: media with links. Hypermedia guides the client to the next page, the next application state
- In the HTTP protocol, Media Types are specified with identifiers like `text/html`, `application/json`, and `application/xml`, which correspond to HTML, JSON, and XML respectively
- HAL (Hypermedia Application Language) format offers a consistent and easy way to hyperlink between resources in API. HAL can be used either in JSON or XML
- Each link in HAL may contain the following properties:

Target URI: It indicates the target resource URI represented as href.

Link relation: The link relation between the current resource and target resource represented as rel.

Type: This indicates the expected resource media type. This is represented by the type attribute.

- Example:

JSON example –

```
{
    "accountNumber": 12345,
    "balance": {
            "currency": "usd",
            "value": 500.00
        },
    "links": [
        {
            "href": "/accounts/12345",
            "rel": "balance",
            "type" : "GET"
```

```
        },
          {}   //Similar links for deposit, withdraw, transfer and close
    ]
}
```

XML example –

```
<?xml version="1.0"?>
<account>
     <accountNumber>12345</accountNumber>
     <balance currency="usd">500.00</balance>
     <link rel="balance" href="https://..../accounts/12345/balance"
type="GET">
     <link rel="deposit" href="https://..../accounts/12345/deposit"
type="PATCH">
     <link rel="withdraw" href="https://..../accounts/12345/withdraw"
type="PATCH">
     <link rel="transfer" href="https://..../accounts/12345/transfer"
type="PATCH">
     <link rel="close" href="https://..../accounts/12345/close"
type="DELETE">
</account>
```

HTML example –

```
<!doctype html>
<div>
     <p accountNumber="12345"></p>
     <p currency="usd">Balance = 500.00</p>
      <a rel="balance" href="https://..../accounts/12345/balance"
type="GET">Check Balance</a>
     <a rel="deposit" href="https://..../accounts/12345/deposit"
type="PATCH">Deposit</a>
     <a rel="withdraw" href="https://..../accounts/12345/withdraw"
type="PATCH">Withdraw</a>
     <a rel="transfer" href="https://..../accounts/12345/transfer"
type="PATCH">Fund Transfer</a>
     <a rel="close" href="https://..../accounts/12345/close"
type="DELETE">Close Account</a>
 </div>
```

References –

- https://dzone.com/articles/rest-api-what-is-hateoas
- https://en.wikipedia.org/wiki/HATEOAS
- https://www.slideshare.net/josdirksen/rest-from-get-to-hateoas
- https://www.slideshare.net/vladimirtsukur/hypermedia-apis-and-hateoas?qid=0712645a-c10c-494a-ab8b-33c162a4514a&v=&b=&from_search=2
- https://restfulapi.net/hateoas/
- https://www.oreilly.com/content/how-a-restful-api-represents-resources/

Ans 6 –

Below are few HATEOAS deficiencies I found –

- According to HATEOAS definition, which states – HATEOAS lets you use hypermedia links in the response contents so that the client can dynamically navigate to the appropriate resource by traversing the hypermedia links. But in our project, there is no such proper navigation required for traversing from users to its own articles or comments. i.e. There are no requirements for providing direct links to articles and comments from each user and vice versa
- With HATEOAS, responses to REST requests return not just the data, but also actions that can be performed with the resource, hence, we can also add DELETE request methods along with its href link. But our specifications do not require any REST request like POST and PATCH methods to implement in project
- In our specifications, it is required to include the relations between the resource and linked resource with the IANA LINK Relations. We are required to include only self, next and previous relations. But there are other relations as well like – first, last, up, edit, search, index, etc that can also be included to make the navigations better and more effective
- HATEOAS supports making custom queries for the better navigation, in the project this is not required
- It is only given that we need to include the relations in HATEOAS, but not specified if we need to hardcode the same

Ans 7 –

A. **A blog which is updated at most once a day.**
   - `Cache-Control: max-age=86400, must-revalidate, public`
   - Since a blog is updated at most once a day, we can add max-age for a day so that it will refresh next day
   - Since blog is going to be updated once a day, we need to revalidate it, hence we can add must-revalidate directive as well
   - Also blog post is for everyone to read; it can be shared as public.

B. **A page which only contains a search form for a library.**
   - `Cache-Control: max-age=31536000, public`
   - Assuming that the search form for library will remain same throughout lifetime or much longer time, we can cache it for a year (or infinite period)
   - Library can be accessed by anyone so it can be shared hence, considered as public.

C. **A page which shows the results for a library search.**
   - `Cache-Control:  no-cache, public`
   - Assuming the books in the library are added/removed or issued by other people. Hence search parameters can change dynamically, so we would want caches to revalidate before display. Hence, we can use no-cache directive.
   - Library can be accessed by anyone so it can be considered as public.

D. **A bank statement.**
   - `Cache-Control: no-store`
   - Since a bank statement contains sensitive information, it should not be stored in any cache for a cache-control header.

E. **A page which displays the contents of a shopping cart.**
   - `Cache-Control: no-cache, private`
   - Since contents of shopping cart is never shared publicly, it should be private
   - Let's assume if the price of any item in the cart is changed, hence we need to revalidate the same before loading every time. Hence, we can add no-cache directive.

Ans 8 –

A. **FALSE –** Any changes done to object affects the object's own property and has no effect on the prototype. Hence, if two JavaScript objects share the same prototype, then any update to a property of the first object will only affect the first object. It will not affect the second object even though they share the same prototype. l

B. **TRUE –** POST method is non idempotent method used to add a new resource while PUT is generally used to completely replace the resource and if that resource is not found then it creates the new resource. PUT is idempotent which means even if the user double clicks the update button, it will not change anything.

C. **TRUE –** User preferences can be stored in the form of cookies. Hence it is possible to store user preferences in the session context in server-side web framework.

D. **TRUE -** Basically SessionStorage maintains a separate storage area that Stores data only for a session, meaning that the data is stored until the browser (or tab) is closed. Which in turn means that this session storage is tied to the session withing a server-side web framework.

E. **FALSE –** Javascript prototypes itself do not support multiple inheritance. While they can inherit the properties of the parent object. JavaScript cannot dynamically inherit from more than one prototype chain. Inheritance of property values occurs at run time by JavaScript searching the prototype chain of an object to find a value. Because an object has a single associated prototype.