University Ferhat Abbas Setif 1 Department of Computer Science Sunday, January 21, 2024

Advanced Web Programming (AWP) Exam

First name: last name: group:

Multiple Choice Questions (MCQ): (0.75 correct answer, -0.5 wrong answer, 0 if you don't answer a question)

1) What is true about **Ajax**?
   a) AJAX is a web development technique for creating interactive web applications.
   b) Ajax update a web page without reloading the page
   c) Ajax request data from a server after the page has loaded
   [✓ d)] All of the above
2) Which of the following feature makes the **Ajax** *unique*?
   a) It can work with all the databases
   b) It can use Python & C++ for programming
   [✓ c)] It makes data requests asynchronously
3) In JavaScript, can we use a function as a variable value?
   [✓ a)] Yes
   b) No
4) Which JavaScript method that takes an array and returns a new array?
   a) forEach()
   [✓ b)] map()
   c) reduce ()
   [✓ d)] filter()
5) What will be the output of:

```
console.log(`${(x => x)('I love')} programming`);
```

   [✓ a)] I love programming
   b) undefined programming
   c) ${(x => x)('I love') programming
   d) TypeError
6) What will be the output of:

```
Promise.resolve(5);
```

   a) 5
   b) Promise {<pending>: 5}
   [✓ c)] Promise {<resolved>: 5}
   d) Error
7) What will be the output of:

```
async function getData() {
  return await Promise.resolve('I made it!');
}
const data = getData();
console.log(data);
```

   a) "I made it!"
   b) Promise {<resolved>: "I made it!"}
   [✓ c)] Promise {<pending>}
   d) Undefined
8) What is a **callback** function in JavaScript?
   a) A function that performs asynchronous tasks.
   b) A function that is called at the end of the program's execution.
   [✓ c)] A function that is passed as an argument to another function and is executed inside that function.
   d) A function that is used for error handling.
9) Which keyword is used to declare **block-scoped** variables in JavaScript?
   a) var
   [✓ b)] let
   [✓ c)] const
   d) variable
10) What will be the output of the following code?

```
function delayLog() {
 for (var i = 1; i <= 5; i++) {
  setTimeout(function () {
   console.log(i);
  }, 1000);
 }}
delayLog();
```

   a) 1, 2, 3, 4, 5
   b) 5, 5, 5, 5, 5
   [✓ c)] 6, 6, 6, 6, 6
   d) 1, 6, 6, 6, 6
11) Why do we use promises instead of **callbacks**?
   a) Because it is faster than callback
   [✓ b)] Because it is more readable
   [✓ c)] To escape call back hell
   [✓ d)] Because it makes the code more maintainable
12) What is the main function of **event loop**?
   a) It executes the code of JavaScript
   b) It creates separate threads for asynchronous tasks to run
   [✓ c)] It checks for the call stack and pushes *callbacks* from the ***callback queue***
   d) It sets timer for setTimeout()
13) Javascript is a multithreaded language
   a) True
   [✓ b)] False
14) What is the role of the *onreadystatechange* event in **AJAX**?
   a) To specify the URL of the server
   b) To specify the request data sent to the server
   c) To specify the response data received from the server
   [✓ d)] To specify the function to be executed when the **AJAX** request status changes
15) What is a "**closure**" in JavaScript?
   a) A function that is stored as a property of an object.
   b) A function that can be accessed globally from any part of the code.
   [✓ c)] A function that is defined inside another function and has access to its outer function's variables.
   d) A function that takes an unlimited number of arguments
16) What will be the output of:

```
const promise1 = Promise.resolve('First')
const promise2 = Promise.resolve('Second')
const promise3 = Promise.reject('Third')
const promise4 = Promise.resolve('Fourth')
const runPromises = async () => {
const res1 = await Promise.all([promise1, promise2])
const res2  = await Promise.all([promise3, promise4])
      return [res1, res2]
}
runPromises()
      .then(res => console.log(res))
      .catch(err => console.log(err))
```

a) [['First', 'Second'], ['Fourth']]
b) [['First', 'Second'], ['Third', 'Fourth']]
c) [['First', 'Second']]
[✓ d)] 'Third'

17) Which of these are true about selecting **DOM** elements?
a) You can select elements by CSS class name
b) You can select elements by id attribute value
c) You can select elements by tag name
[✓ d)] All of the above

18) What will be the output of the following JavaScript code?

```
let x=[6,5,8,9,7];
let a=x.reduce((a,c)=>c>a?a:c,7);
console.log(a);
```

a) 7
b) 9
[✓ c)] 5
d) Error

19) What will be the output of the following JavaScript code?

```
let a=[5,3,2,9].map(i=>(j)=>i+j);
let x=a.filter(i=>i(2)>5)
      .forEach(i=>console.log(i(5)))
```

a) 10,6,15
[✓ b)] 10,14
c) 11,15
d) undefined

20) What is the main benefit of using **WebAssembly**?
[✓ a)] Faster performance than JavaScript
b) Easier to learn than JavaScript
c) More secure than JavaScript
d) More compatible with older browsers than JavaScript

Exercice (5 points):

Given the following code:

```
let Person =()=> '[{"name":"Ahmed"}, {"name":"adem"},
{"name":"Ali"}]';
let P=JSON.parse(Person());
let getPerson=(i,x)=>{setTimeout(()=>x[i].name,2000)};
console.log(getPerson(2,P));
```

1. What will be the output of the code and why?
2. In order to obtain the correct result, update the above code using:
   a. Promise
   b. Async and await.
3. Using the DOM API, create an HTML element <ul>, then fill it with all the names of the Person variable.

Solution:

1. Undefined: getPerson(2,P) is asynchronous task as the result will be returned after 2 seconds. 0.5
2. Promise: 1.5

```
let getPerson=(i,x)=>{
    return new Promise(resolve=>{
     setTimeout(()=> resolve(x[i].name),2000)
    })
  };
  getPerson(2,P)
  .then(name=>console.log(name))
```

Async and await: 1.0

```
 async function getdata(){
   let p= await getPerson(2,P);
   console.log(p);
   }
  getdata();
```

3. 2.0

```
let ul=document.createElement('ul');
     P.forEach(i => {
       let li=document.createElement('li');
       li.textContent=i.name;
       ul.appendChild(li);
     });
     document.body.appendChild(ul);
```