Table of Contents

Lecture 10 jQuery

1. **5 signatures of jQuery function**
2. window.onload vs $(document).ready(), $(function(){});

   First Way: $(document).ready(function() { }); // jquery function
   Second Way: $(function() { }); //another jquery shortcut

3. **node identification**
   a. **jQuery selectors**

```
const elem = $("#myid");
const elems = $('input');
```

   b. **context identification**

```
<ul id="myid">
  <li class="special"> okey</li>
```

```
  <li> not okey</li>
</ul>
<script>
  const $elem = $("#myid");
  const specials = $("li.special", $elem); //or

  const notspecial = $elem.find("li.special");
  // elem.find("li.special");
  console.log(specials.text());
  console.log(notspecial.text())
</script>
```

c. jQuery function versus jQuery object versus DOM elements

the jQuery function refers to the global jQuery function that is normally aliased as $ for brevity (keringkasasn)
a jQuery object the object returned by the jQuery function that often represents a group of elements
selected elements the DOM elements that you have selected for, most likely by some CSS selector passed to the jQuery function and possibly later filtered further

d. traversing dom tree

An ancestor is a parent, grandparent, great-grandparent, and so on.
• **parent()** - returns the direct parent element of the selected element.
• **parents()** - returns all ancestor elements of the selected element. You can also use an optional parameter to filter the search for ancestors.
• **parentsUntil()** - returns all ancestor elements between two given arguments.
$("span").parent(); $("span").parents(); $("span").parents("ul"); // returns all ancestors of all elements that are
elements $("span").parentsUntil("div");
• **children()** - returns all direct children of the selected element • You can also use an optional parameter to filter the search for children
• find() - returns descendant elements of the selected element

4. modify DOM nodes

- **looping with $().each**
- ```
  $("#ex2 span.special").each(function(i, elem) {
    var img = $("<img>")
      .attr("src"
        , "../images/laughing_man.jpg")
      .attr("alt", "laughing man")
      .css("vertical-align", "middle")
      .css("border", "2px solid black")
      .click(function() {
        alert("clicked");
      });
    $(elem).prepend(img);
  });
  ```
- **Chaining**
- ```
  $("img").css("color", "red");
  $("img").attr("id", "themainarea");
  $("img").addClass("special");

  // The implictly returned jQuery object allows for chaining of method
  calls.
  ```

```
$("img") // good jQuery style
  .css("color", "red")
  .addClass("special")
  .attr("src"
    , "foo.png");
```

- **get/set CSS classes and unobtrusive styling**

```
function okayClick() {
  this.style.color = "red";
  this.className = "highlighted";
}
.highlighted { color: red; }
```

5. **create new nodes**

- **innerHTML hacking**
- ```
  document.getElementById("myid").innerHTML +=
    "<p style='color: red; " +
    "margin-left: 50px;' " +
    "onclick='myOnClick();'>" +
    "A paragraph!</p>";
  ```
- **append, prepend, before, after**
- ```
  <div class="container">
    <div class="inner">Hello</div>
    <div class="inner2">Good</div>
  </div></body>
  <script src="https://code.jquery.com/jquery-3.4.1.js"></script>
  <script>
    $(".inner").prepend("<p> test</p>"); // before the inner
    $(".inner").append("<p> test 3</p>"); // after the inner
  </script>
  ```
- ```
  $(".container").after($("h2")); /// first: h2 front -> then h2 will
  after container
  ```
- ```
  $(".inner").after("<p>Test After </p>"); // after inner there will be
  p test after.
  ```

Lecture 11 Events:

1. Event handlers and events

2. Unobtrusive assignment of event handlers

3. event object(methods: preventDefault() , stopPropagation() , stopImmediatePropagation())

4. keyword this in event handlers

5. event bubbling

a. preventDefault Prevents the browsers default behaviour (such as opening a link), but does not stop the event from bubbling up the DOM.

b. stopPropagationevent.stopPropagation(), Prevents the event from bubbling up the DOM, but does not stop the browsers default behaviour.
c. stopImmediatePropagation

d. event delegation

6. **callbacks, concurrency and event loop**

Lesson 12: Servlets

1.Steps of web dynamics

2.Features of HTTP

3.HTTP Request type: POST vs GET

4.Web server: what do web servers serve? Limitations?

4.Servlet: what is servlet? Who manages ? how to create a servlet? How to configure

5.servlet url(annotation, web.xml)?

6.What is web container? What kind of support does the container provide?

7.Lifecyle of how container handles an HTTP request

a. Especially the thread model and its implications

8. Lifecyle of a servlet

Lesson 13: Managing State

1. **forward vs Redirect**

2. how to get request parameter?

3. Scope: request, session, application/context

4. attribute vs parameter

5. **session: how to get session, how to get rid of session**

6. cookie: how to send a cookie, read cookie, characteristics of cookie

7. 6 ways to maintain state demo

Lesson14: JSP

1. Main idea of JSP

2. Two types of data in JSP page

3. **JSP elements: directive, declaration, scriptlet, expression, EL expression, action, comment(vs HTML comment)**

4. JSP Lifecycle

5. What does the container do with your JSP?

6. Implicit objects(request, response, out, session, application, config, pageContext, page, exception)

7. **EL (expression language): how is it evaluated, "null friendly"**

8. JSP actions and tags, e.g., c:out and c:forEach

Lesson15: Ajax & JSON

1. **Synchronous web communication vs asynchronous web communication**

2. **Typical ajax request**

3. **Ajax callback and debugging Ajax calls in browser console**

4. Different syntax for making jQuery ajax request ($.ajax(), $.get(). $.post(), object literal)

5. JSON – basic syntax of JSON

6. JSON.parse(String), JSON.stringify(Object)

7. JSON expression - how to retrieve JSON data

8. **How servlet handles Ajax request and partial page refreshing on client (e.g., guestlist code)**

Lecture 07 Scope:

1. Implicit global
2. Scope(Lexical, hoisting)
3. Best practice for var, let, const
4. 2 phases of JavaScript compiler
5. Execution context and scope chain
6. Closures and free variables

    1. Closures for callbacks and saving state info
    2. Closures for namespace protection
7. Overloading?
8. Arrow functions
9. Map, filter, reduce

Lecture 08/09 this and revealing module and inheritance:

1. Method versus function
2. this keyword: in methods, in functions

a. self-pattern and arrow function
3. Module Pattern (IIFE) – syntax, what's the problem it solves and how?

a. ES6 scope solution

4. revealing module pattern, public and private methods and properties
5. inheritance
    1. prototypal inheritance,
    2. Object.create(),
    3. Function Constructor
    4. Keyword 'new'
    5. __proto__ and .prototype
    6. How is functionality extended to all object instantiations with function constructors and prototypes

WAP Final Exam

True/ False Questions

1) All global variables in any Javascript file are visible to all other Javascript files of the same webpage. **True**

2) Each Javascript file has its own global namespace, separate from other files. **False**

3) All declarations of free variables are hoisted into the global object. **False**

4) Using the DOM, Javascript code can determine which Browser is being used. **True**

5) Unobtrusive Javascript means all Javascript code for a website is put in the same file. **False**

6) Javascript code is executed when the browser reads the script tag. **True**

7) An anonymous function can never be called because it has no name. **False**

8) The Javascript instruction x = $("p") stores a jQuery object in variable x. **True**

9) If y is a DOM object, then y.tagName contains the HTML tag name. **True**

10) HTML Form data always goes directly to the Server, it cannot be accessed by Javascript running on the Browser. **False**

11) An XML.HttpRequest to a Web Server must always return data in XML.format. **False**

12) When using a jQuery $.ajax() call, webpage form data in XML format. **True**

22) In a jQuery $.ajax() call, webpage format data can be sent to the server as a GET request without app query string to the ajax URL parameter. _____????

13) when using jQuery $.ajax(), both Get and Post requests are allowed. **True**

14) By using jQuery $.ajax(), the Javascript code running in the Web Browser can access SQL Databases stored on the client Computer **False**

15) Most Web Browsers have jQuery built into the Browser. **False**
16) When the Javascript "new" operator creates an object, the constructor function defines the prototype of the object. **True**
17) A Javascript object created with Object.create() can be extended with new fields without changing its prototype object. **True**
18)In Javascript, an event handler is always a function. **True**
19) The DOM document object contains the window object. **False**
20) Javascript variables defined with let have block scope**. True**
21) When using a jQuery $.ajax() call, webpage form data can be sent to the server.

Day 3.1 Inheritance CkQuiz

1.foo (x)[
return function baz(y) [alert(x); alert(y);] ;
]
const boo = foo(5);
The value of boo will be function baz(y) [alert(5); alert(y)]
2. Complete the following so that the code produces 2 alerts with values 5 and 10.: foo (5)(10) ;
3. You can use __proto__ to set the [[prototype]] property, but it is not recommended for production code because it is deprecated. true
4. The __proto__ property is identical to the Foo.prototype property for a function Foo.
False : They are related but not identical.
5. The value of the Foo.prototype property for a function Foo will be set to be the value of the new object returned when Foo is called as a function constructor.
I.e., newObject === Foo.prototype

False : newObject.__proto__ === Foo.prototype
6. //assume the standard curly brackets where you see [ ] (sakai issue) let creature = [look: function() [this.isLooking = true;]]; let monkey = [__proto__: creature]; monkey.look(); The code above:
B. monkey will gain an isLooking property with value true

7. //assume the standard curly brackets where you see [ ] (sakai issue) function User(name)[this.name = name;] let newUser = User("Fred"); newUser.name will show
C. Uncaught TypeError: Cannot read property 'name' of undefined : missing new keyword

Day 3.2 Class Ckquiz

1. In JavaScript a free variable is a variable used in an inner function and is declared outside that function and will be bound by value in a closure.
Answer Key: False
Feedback:Only partially true. Statement is not "true" if it is partially false. In this case, first part is true, but it is not bound by value. That part is false so the whole statement is logically false.
2. Convert

```
let animal = {
  eats: true
};
function Dog(name) {
  this.name = name;
```

```
}

Dog.prototype = animal;
let snoopy = new Dog("Snoopy");
alert( snoopy.eats ); // true
let animal = {

  eats: true

};


function Dog(name) {

  this.name = name;
}

let snoopy = new Dog("Snoopy");

snoopy.__proto__ = animal;
alert( snoopy.eats ); // true
```

3. from prototype to object.create

```
let animal = {
  eats: true
};
function Dog(name) {
  this.name = name;
}

Dog.prototype = animal;
let snoopy = new Dog("Snoopy");
alert( snoopy.eats ); // true

Object Create :

let animal = {
  eats: true
};
let dog = Object.create(animal);
dog.name="Snoopy";
alert( dog.eats ); // true
```

4. **JavaScript classes are functions , true**

5. Convert to class

```
function User(name) {
  this.name = name;}

User.prototype.sayHi = function() {
  alert(this.name);};

let user = new User("John");
user.sayHi();


//convert to class
class User{
  constructor(name) {
    this.name= name;
  }

  sayHi()
  {
    alert(this.name);
  }
```

```
}

let obj = new User("hello");
obj.sayHi();
```

6. **function Counter() {**
let count = 0;

this.up = function() { return ++count; };
this.down = function() { return --count; };
}

let counter = new Counter();

alert(counter.up()); // ?
alert(counter.up()); // ?
alert(counter.down()); // ?

Answer the following:

1. what is the free variable? count

2. How many closures? 2

3. What is the private variable? 0

4. What are the public methods?
1

```
7. function Rabbit(name) {
  this.name = name;
}

Rabbit.prototype.sayHi = function () {
  alert(this.name);
};
let rabbit = new Rabbit("Rabbit");

//Do all of these calls do the same thing?
rabbit.sayHi();
Rabbit.prototype.sayHi();
Object.getPrototypeOf(rabbit).sayHi();
rabbit.__proto__.sayHi();
```

False

Day 3.3 jQuery Check Quiz

1.the $ function returns (choose best)
jQuery array-like object of DOM objects

2. if the argument to $(myArg) is a JavaScript function what will jQuery do

A. evaluate the function when the document.onload event fires

3. window will be the value of 'this' for the callback function given to setInterval.

4. Complete the jQuery function call with a single argument that will find all spans inside divs with class 'sunny' that are descendents of elements with id = 'happyDay'. Use only the jQuery function, do not use any jQuery helper methods like find.
#happyDay div.sunny span
5. What is the difference of the jQuery function and 'a jQuery object' ?
jQuery Function -> related to the global jQuery function that (using $).
The jQuery function is the global jQuery function, i.e., '$', that contains all of the jQuery functionality. 'jquery object ' refers to the array-like collection of DOM elements that is returned by the 'jQuery function'.

Questions : 6-8

```
/* Object.create */
const personProto = {eat: alert("eating")};
const bob = Object.create(personProto);
bob.name = "Bob";
bob.eat();

/* rewrite the code to use __proto__ instead of Object.create */
const personProto = {eat: alert("eating")};
const bob = {};
bob.__proto__ = personProto;
bob.name = "Bob";
bob.eat();

/* rewrite the code to use a constructor function */
function Person(name) {
  this.name = name;
}

const personProto = {eat: alert("eating")};
Person.prototype = personProto;
const bob = new Person("Bob");
bob.eat();

//rewrite the code to use Class
class Person {
  constructor(name) {
    this.name = name;
  }

  eat() {
    alert("eating")
  }
}

const bob = new Person("Bob");
bob.eat();
```

Day 3.4 Event Handling Ck Quiz

1. What will the value of 'this' be if it is in an event handler assigned to an event such as a click on a button and the event handler function is written as an standard named function declaration, e.g., myClickHandler(){ ... }? (assume strict mode)
the object that calls the event handler
2. What will the value of 'this' be if it is in an event handler assigned to an event such as a click on a button and the event handler function is written as an anonymous function declaration? (assume strict mode)
the object that calls the event handler

3. 'this' has lexical scoping when used in an arrow function. What will the value of 'this' be if it is in an event handler assigned to an event such as a click on a button and the event handler function is written as an arrow function? (assume strict mode)
the 'this' of the enclosing lexical scope

4. Which event fires first if there are multiple event handlers assigned to the same element (and same event)?
A. first one assigned
C. best not to depend on the order of assignment

5. Event delegation requires events to bubble

True
6. Asynchronous JavaScript events are inserted
C. at the back of the event queue

Day 3.5 Ckquiz - Servlets and Containers and JS

1.What is the difference of a web server and a web container?

Web server is a machine that receives an HTTP request and sends an HTTP response, while web container is a specific server that can manage java servlets.

2. Who creates HttpServletRequest and HttpServletResponse objects?
C.Web container
3. Who calls service() and when?
Container calls service() method when the request comes. Depending on the request, service() calls either doGet or doPost.
4. What are the implications of multi-threading for servlet instance variables?
There is only one instance of servlet, and container creates thread for each request. There is possibility of race condition, so it is advised to avoid instance variables in servlets.
5. Instance variables are shared by all requests that come to the same servlet at the same time. Unless you control for mutual exclusion there might be race conditions. Therefore, you should avoid using instance variables in servlets.

true

6.the init method runs before service on every request to a servlet. If there are 100 requests init will run 100 times.

False

7. If there are 1000 requests for a servlet at the same time there will be 1000 instances (objects) of that servlet.

False

8. $('div').each(myFunction); In the above line of code 'each' is best described as
C. a method on the jQuery object '
G. a property whose value is a function
$('div') returns a 'jQuery object' (array-like collection of objects)

```
<ul>
```

```
  <li>foo</li>
  <li>bar</li>
</ul>
```

```
$( "li" ).each(function( index ) {
  console.log( index + ": " + $( this ).text() );
});
```

```
0: foo
1: bar
```

9.

```
(function() {
  $.fn.extend( {toObject: function() {
      var result = {}
      var test = [{name: 'x',value: 1}, {name: 'y', value: 1}];
      $.each(test, function(i, v) {result[v.name] = v.value; });
      return result;
    } )();
```

A. result[v.name] = v.value assigns v.value to be the value of the v.name property of the result object
B. when this code runs the IIFE will be invoked and it has a single statement that will be executed, the $.fn.extend method.
D. result is an object
E. when this code runs the IIFE will be invoked
F. extend() is a method and is also a property of the object $.fn
H. There is a single argument being sent to extend(), which is an object literal

10. Asynchronous JavaScript events are inserted
B. at the back of the event queue
11. The deployment descriptor file is a good thing to check if the browser reports that it cannot find
B. the servlet
12. Which of the following servlet methods are empty methods that are intended for application developers to override

A. destroy
B. init
D. doPost
E. doGet

Day 3.6 Ckquiz - State management

1. same with no 9 prev day quiz
2. What is the difference between attributes and parameters?

Attributes are map of key and value pairs bound in either of the three scopes (request, session, application), while parameters are map of key and value pairs that is submitted from HTML form or in URL header.

3. How long does a session last?

- A. Until some number of seconds, which are specified in web.xml as session-timeout, pass without any use
- B. Until session.invalidate() is called
- C. Until application is undeployed

4. What is a web cookie?
Name/value pair of strings sent as message header between client and server

5. A JSP page is interpreted and compiled into a servlet every time there is a request for that page.
False
Once the JSP servlet has compiled the servlet class from the JSP source code, it just forwards the request to this servlet class.

6. Syntax A:
function myAFunction()
{ // do something }
Syntax B:
var myBFunction = function() { // do something }

B. myAFunction can be called before it's definition appears in the file
C. myAFunction and its definition is hoisted
7. If you do not set a maximum age for a cookie then it will be a permanent cookie.

False

8. The container sometimes puts input request parameters in HttpSession objects.
False
9. When a servlet forwards (via request dispatch) a request to another servlet (possibly a JSP page), the next servlet continues to process the same request. True
10. Suppose person.pet.name appears as the expression inside a JSP EL expression, e.g. ${person.pet.name} . How will person be used by JSP EL?
C. person, even though it is not in quotes, will be the value of a string that is a key in a collection of attributes

11. $.each($('p'), myFunction); In this line of code, which of the following are true for 'each' (more than one may be true, check all that are true)
B. a method on the jQuery function

F. a property whose value is a function
12. What is the difference between redirect and request dispatch (forward)?
Redirect asks the client to make another request to a different servlet or URL, while request dispatch forwards the current request on behalf of the client so that the client doesn't know this is happening.

State Management Quiz

1. Can post requests be bookmarked? What are the problems?

A HTTP post can be bookmarked, but since a bookmark only consists of the URL, all of the parameters will be lost.

2. What is the purpose of request dispatching?

The request dispatching has the purpose to forward the request processing to another local resource.

3. What is the difference between redirect and request dispatch?

The request dispatching forward the request to another local resource, servlet or JSP, for example, and the redirect will redirect to another URL, and it may be to another site or the same site.

4. What is an attribute ?

Is an object bound into one of the three servlet API objects and it has a name value pair and can be used to store values in a server-side purpose.

5. What is the difference between attributes and parameters?

Parameters are those that are passed from the client to the server and they are represented by string type and attributes are for server-side usage and can be used not for just, string, but objects as well. Another difference is that parameters are readonly and attributes are read/write objects.

6. What are dangers of using attributes?

Context scope attributes has application level state, then they are global and shared by every servlet and every request in the application, because of that they are not thread safe.

7. What does it mean to say that http is stateless? Give an example of a stateful protocol.

A stateless protocol is a protocol in which each communication is handled as an independent unrelated to other similar communications. A TCP connection-oriented session is a stateful protocol because both systems during the connection maintain information about the session itself during its life.

8. Give 5 different methods for maintaining state information (count each attribute scope as one method).

1. request scope: destroyed when servlet finishes processing request
2. session scope: destroyed when user closes browser
3. application scope destroyed when the container stopped.
4. Cookies saved on browser, either temporary (deleted when the browser closes) or permanent
5. Hidden fields on a form

9. How long does a session last?

By default, a session lasts 30 minutes but you can adjust this limit.

10. What is a cookie, and how long does a cookie last?

HTTP cookies are usually small text files, given ID tags that are stored on your computer's browser directory. They are created to maintain the state about some user information.

11. What is the purpose of URL rewriting?

It is the process of altering the parameters in a URL. Changing the URL can help with users access and also helps the site visibility. And can be used by hackers to redirect users without their knowledge.

12. Why does the request attribute report 'null' for the maintaining state demo?

Because the request scope it is destroyed when a servlet finishes processing request. So in the next request the value doesn't exists anymore.

reasonably likely explanation is that the requests are not in the same session. Maintaining a session requires cooperation from the client, which is not guaranteed to be given. The most common mechanisms for associating requests with sessions are cookies and URL rewriting. If the client refuses cookies and is making its requests to a static URL then every request will likely be in its own session.

Servlet Quiz

1. **What is the difference between a web server and a web container?**

   Web Server is capable of handling HTTP requests, sent by client and respond back with HTTP response. Web Container is the component of a web server that interacts with java servlets.
2. **What is a servlet?**

   It is a Java class that is used to extend the capabilities of servers.
3. **How do web servers and web containers interact with servlets?**
   Firstly, the web server receives requests by the client and delegate these requests to the servlet container. Then, the servlet container loads the servlet. And the servlet container handles multiple requests for the same servlet by spawning multiple threads, one thread per request, each executing the service() method of a single instance of the servlet.
4. **Who creates request objects?**
   The servlet container.
5. **What are the states in the servlet lifecycle?**
   Initialization, Processing and Destroy.
6. **Who calls init and when?**
   The servlet container calls the init and it may happen at the container startup or at the first time the client invokes the servlet with a request.
7. **Which of init, service, and doGet should you override?**
   doGet, because the init it's called to initialize the servlet and the service method invokes the doGet or doPost method, that you should override.
8. **In what sense are servlets multi-threaded?**
   In the way that the servlet container handles multiple requests for the same servlet by spawning multiple threads, one thread per request, each executing the service() method of a single instance of the servlet.
9. **What are the implications of this for servlet instance variables?**
   The servlet has a single instance and each instance variables is shared for all the requests, so these variables are not thread safe.

JSP Quiz

1. Rewrite the JSP page in the first demo (forEach) to use scripting instead of the JSTL forEach.

```
<%

   Student[] students = (Student[])request.getAttribute("students");

   for (Student student :
         students)
   {
      out.print("<tr><td>"+ student.getName() +"</td><td>"+ student.getAge()
+"</td></tr>");
   }
%>
```

2. How do custom tags relate to JSTL?

JSTL is a standard library of JSP actions, and JSP allow us to create our own actions, that you call custom tags.

3. What is the role of the URI in the TLD and the taglib directive?

The role of the URI is to define the directory of the TLD file, and the taglib directive is a directive that is going have two main attributes, which are URI and prefix, the URI attribute is going to refer to an specific TLD file, in this case it's going to be a local URI in your project referring to the TLD created, and also it has a prefix attribute, which is the prefix that you have to use in the HTML to refer to the custom tag created.

4. What is a tag handler class?

It's a class that you have to create to manipulate the referent custom tag used in the HTML. It's a class that is going to have all the attributes that you are going to use in your jsp file and you the class must extends the javax.servlet.jsp.tagext.TagSupport class, override the doTag method and it is also required to implement set methods.

5. What is the role of attribute setters in a tag handler class?

In order to the container make the bind to the value of each attribute presented in the jsp file to the value of each attribute presented in the tag handler class, it is required to have a setter method.

6. What is the role of the doTag() method in a tag handler class?

The doTag method will manipulate the attributes of the tag and return a html to the browser based in the attributes manipulated, replacing the custom tag.

7. What does the operation getJspContext().getOut().write("Hi Bob") do when called in a doTag() method?

Will replace the custom tag in the jsp file to Hi Bob.

8. What does the operation getJspBody().invoke(null) do when called in a doTag() method?

If your custom tag has a body inside the container will process the JSP content found in the body of the tag just like any other JSP page content.

1. What is the main value of Java Server Pages?

It's separate the display from processing, in other words, separate html from java.

2. How are JSP pages related to servlets?

It's related to the servlets in the way that in the compilation process, the JSP is turned into a servlet and the life cycle is similar to the life cycle of servlets as well.

3. How are JSP pages related to HTML?

It's related to HTML in the way that JSP is a technology for developing web pages that allow developers to insert java code in HTML pages.

4. Give an example of a JSP scriptlet and show how it will look in the JSP servlet .

JSP scriptlet

```
<%! Int count = 0 %>
```

JSP servlet

```
public class ExampleServlet()
{
        init(){ … }
        service(){
        int count = 0;
        }
        destroy(){ … }
}
```

5. Give an example of a JSP declaration and show how it will look in the JSP servlet.

JSP declaration

```
<%! Int count = 0 %>
```

JSP servlet

```
public class ExampleServlet()
{
        int count = 0;
        init(){ … }
        service(){ … }
        destroy(){ … }
}
```

6. Give an example of a JSP comment and show how it will look in the JSP servlet.

JSP Comment

```
<%-- jsp comment --%>
```

JSP Servlet

```
//jsp comment
```

7. Give an example of a JSP expression and show how it will look in the JSP servlet.

JSP Expression

```
<%= ++count%>
```

JSP Servlet

```
out.print(++count);
```

8. Give an example of a JSP directive and show how it will look in the JSP servlet.

JSP Directive

```
<%@ page import="java.util.Date" %>
```

JSL Servlet
import java.util.Date;

9. Explain how an EL expression is evaluated.

${something}

Firstly, the container checks the page scope for an attribute named “something”. If found use it. If not, it’s going to check sequentially and do the same with the followings scopes: the request scope, the session scope, and application scope. If it doesn’t find, then ignore the expression.

10. Explain how servlet attributes are involved in EL expressions.

They are involved because the container check all the attributes in created in all the scopes as request, session, application and page.

11. Explain how servlets and JSPs use request dispatch to interact.

It’s used when you apply the MVC architecture, so the servlet it’s represented by the controller and the role of the controller it’s to manipulate the model and return the model to the view, so, the view in this case it’s going to be the JSP’s and the servlet it’s going to use the request dispatch to return the processed information to the view.

JSP II

1. What is the main value of Java Server Pages?

It’s separate the display from processing, in other words, separate html from java.

2. How are JSP pages related to servlets?

It’s related to the servlets in the way that in the compilation process, the JSP is turned into a servlet and the life cycle is similar to the life cycle of servlets as well.

3. How are JSP pages related to HTML?

It’s related to HTML in the way that JSP is a technology for developing web pages that allow developers to insert java code in HTML pages.

4. Give an example of a JSP scriptlet and show how it will look in the JSP servlet .

JSP scriptlet

```
<% Int count = 0 %>
```

JSP servlet

```
public class ExampleServlet()
{
    init(){ … }
    service(){
    int count = 0;
```

```
}
    destroy(){ … }
}
```

5. Give an example of a JSP declaration and show how it will look in the JSP servlet.

```
JSP declaration
<%! Int count = 0 %>

JSP servlet
public class ExampleServlet()
{
    int count = 0;
    init(){ … }
    service(){ … }
    destroy(){ … }
}
```

6. Give an example of a JSP comment and show how it will look in the JSP servlet.

```
JSP Comment
<%-- jsp comment --%>
JSP Servlet
//jsp comment
```

7. Give an example of a JSP expression and show how it will look in the JSP servlet.

```
JSP Expression
<%= ++count%>
JSP Servlet
out.print(++count);
```

8. Give an example of a JSP directive and show how it will look in the JSP servlet.

```
JSP Directive
<%@ page import=”java.util.Date” %>
JSL Servlet
import java.util.Date;
```

9. Explain how an EL expression is evaluated.

${something} Firstly, the container checks the page scope for an attribute named “something”. If found use it. If not, it’s going to check sequentially and do the same with the followings scopes: the request scope, the session scope, and application scope. If it doesn’t find, then ignore the expression.

10. Explain how servlet attributes are involved in EL expressions.

They are involved because the container check all the attributes in created in all the scopes as request, session, application and page.

11. Explain how servlets and JSPs use request dispatch to interact.

It's used when you apply the MVC architecture, so the servlet it's represented by the controller and the role of the controller it's to manipulate the model and return the model to the view, so, the view in this case it's going to be the JSP's and the servlet it's going to use the request dispatch to return the processed information to the view.

JQUERY

2. Write jQuery code to find all h1 elements that are children of a div element and make their bac kground color red. Sample HTML:

```
<body>
    <h1>abc</h1><br>
    <div>
        <h1>div-1</h1>
        <h1>div-2</h1>
    </div>
    <h1>xyz</h1>
</body>
```

$("div h1").css("background-color", "red");

3. Use a jQuery method to insert the text "YES!" at the end of the <p> element.

```
<!DOCTYPE html>
<html>

<head>
<script>
$(document).ready(function () {
$("p").append("YES!");
});
</script>
</head>

<body>
<p>Is jQuery FUN or what? </p>
</body>

</html>
```

1. Which, if any, of the following 3 code fragments are equivalent? Explain why they are different, if they are. Explain why they can have different parameters and be equivalent, if they are equivalent.

```
//code fragment 1
$("li").each(function(idx, e) {
 $(e).css("color", "yellow"); });
//code fragment 2
$("li").each(function() {
 $(this).css("color", "yellow"); });
//code fragment 3
$("li").each(function(idx) {
 $(this).css("color", "yellow"); });
```

All of them are equivalents in the way that all change the color of the same element, the only difference is that they pass different parameters in each situatoins, so the parameters of the each function are optional, if you are going to use, you should know that the first parameter it's going to keep track of the current iteration index and the second one it's going to refer to the current element in the iteration.

2. Write a jQuery expression to find all divs on a page that include an unordered list in them, and make their text color be blue.

```
<div>no ul here </div>
<div>
This does contain a ul.
<ul>
<li>the first item</li>
<li>the second item</li>
</ul>
</div>
<script>
<!—INSERT YOUR JQUERY CODE HERE - - >

$(document).ready(function ()
{
     $("div ul").css("color","blue");

});



</script>
</body>
```

3. Write jQuery code to append the following div element (and all of its contents) dynamically to the body element.

<h1>JQuery Core</h1>

HTML:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>JS Bin</title>
</head>
<body>
</body>
</html>
```

```
$("body)".append("<div><h1>JQuery Core</h1></div>");
```

1. Find the text in the first paragraph (stripping out the html), then set the html of the last paragraph to show it is just text (the red bold is gone).

```
<!doctype html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <title>text demo</title>
    <style>
        p {
            color: blue;
            margin: 8px;
        }

        span {
            color: red;
        }
    </style>
    <script src="http://code.jquery.com/jquery-1.9.1.js"></script>
</head>

<body>
    <p><span>Test</span> Paragraph.</p>
    <p></p>
    <script>
< !—INSERT YOUR JQUERY CODE HERE - - >


                    $(document).ready(function () {
                        $("body p:nth-child(2)").text($("body p:first-
child").text());
                    });



    </script>
</body>

</html>
```

2. Write jQuery code to create a red background for the level-2 list items.

```
<ul class="level-1">
    <li class="item-i">I</li>
    <li class="item-ii">II
        <ul class="level-2">
            <li class="item-a">A</li>
            <li class="item-b">B
                <ul class="level-3">
                    <li class="item-1">1</li>
                    <li class="item-2">2</li>
                    <li class="item-3">3</li>
                </ul>
            </li>
            <li class="item-c">C</li>
        </ul>
    </li>
    <li class="item-iii">III</li>
</ul>
```

```
<script>
    $(document).ready(function () {
        $("ul.level-2 > li").css("background-color", "red");
        $("ul.level-3").css("background-color", "white");
    });
</script>
```

3. Write jQuery code to select the element that comes immediately before item three and change its background color to blue.

```
<ul>
    <li>list item 1</li>
    <li>list item 2</li>
    <li class="third-item">list item 3</li>
    <li>list item 4</li>
    <li>list item 5</li>
</ul>
```

```
<script>
    $(document).ready(function () {
        $(".third-item").prev().css("background-color",
"blue");
    });
</script>
```

4. Let us one additional requirement for the "Go Vegetarian" button of the Webville Eatery Menu described in Chapter 4 of Head First jQuery: "Turkey" in the ingredient list of any entree is replaced by ”Mashed Potatoes" in the vegetarian version. You may assume there is a class "turkey" that identifies these items in the list. Describe the changes to the Javascript (jQuery) code to implement this new requirement.

$(“.turkey”).replaceWith(“<li class=’mashed-potatoes’>Mashed Potatoes”);

5. Write Javascript (jQuery) code to change the color of the parent and grandparent list items of the span of text ("some text") in the following to green.

```
<!DOCTYPE html>
<html>

<head>

        <script>
            $(document).ready(function () {
< !—INSERT YOUR JQUERY CODE HERE - - >

                        $(".ancestors
span").parent().parent().css("color", "red");
            });
        </script>


</head>

<body>
    <div class="ancestors">
        <div> (great-grandparent)
            <ul> (grandparent)
                <li> (direct parent)
                    <span>some text</span>
                </li>
            </ul>
        </div>
    </div>
</body>

</html>
```