**Assignment 4.2**

*<!doctype html>*

*<html lang="en">*

*<head>*

*<meta charset="utf-8">*

*<title>focus </title>*

*<style>*

*span {*

*display: none;*

*}*

*</style>*

*<script src="https://code.jquery.com/jquery-3.5.0.js"></script>*

*</head>*

*<body>*

*<h3>Example of focus event HAndler.</h3>*

*<p>Enter your id :<input type="text" placeholder="id"> <span>focus fire</span></p>*

*<p>Enter your Name :<input type="text" placeholder="john"> <span>focus fire</span></p>*

*<script>*

*$( "input" ).focus(function() {*

*$( this ).next( "span" ).css( "display", "inline" ).fadeOut( 1000 );*

*});*

*</script>*

*</body>*

*</html>*

**Assignment 1.4**

*JWT or JSON Web Token, is an open standard used to share security information between two parties — a client and a server. Each JWT contains encoded JSON objects, including a set of claims. JWTs are signed using a cryptographic algorithm to ensure that the claims cannot be altered after the token is issued.*

*IT WORKS AS*

*JWTs differ from other web tokens in that they contain a set of claims. Claims are used to transfer information between two parties. What these claims are depends on your specific use case. For example, claims can determine who issued the token, how long the token is valid, or the permissions granted to the client.  
  
A JWT is a three-part string separated by periods (.) and serialized using base64. In compact serialization, which is the most common serialization format, the JWT looks like xxxxx.yyyyy.zzzzz.  
  
After decoding, we get the following two JSON strings.  
  
Header and payload.  
signature.  
Header JOSE (JSON Object Signing and Encrypting) contains the type of the token (in this case her JWT) and the signing algorithm.  
  
Payload contains claims. This appears as a JSON string and typically contains 12 or fewer fields to keep the JWT compact. This information is typically used by servers to verify that the user is authorized to perform the requested action.  
JWTs don't have binding claims, but they can be bound by overriding the standard. For example, when using JWT as a bearer access token with OAuth2.0, iss, sub, aud, and exp must be present. Some are more common than others.  
  
A signature ensures that the token has not been altered. The party creating the JWT signs the header and payload with a private key known by both the issuer and receiver, or known only by the sender. When the token is used, the receiver verifies that the header and payload match the signature.*

*BENEFITS*

1. *No Database Table : This implies fewer DB queries, which implies faster response time. In case you are using paid services like DynamoDb that charge per query basis, JWT might reduce the costs marginally.*
2. *Simpler to use if careful : If your architecture doesn’t user client Sessions and your security basics are clear, the development time in case of JWT is faster using the existing libraries.*
3. *Used across services : You can have one authorization server that deals with the Login/Registration and generates the token, all the subsequent requests will need not have to go to the authorization server as the only the Auth-server will have have the private key, and rest of the severs will have the public-key to verify the signature.*

**Assignment 8.2**

*// checks prime number*

*function isPrime(n)*

*{*

*if (n <= 1) return false;*

*if (n <= 3) return true;*

*if (n%2 == 0 || n%3 == 0) return false;*

*for (let i=5; i\*i<=n; i=i+6)*

*if (n%i == 0 || n%(i+2) == 0)*

*return false;*

*return true;*

*}*

*// next prime function*

*function nextPrime(N)*

*{*

*if (N <= 1)*

*return 2;*

*let prime = N;*

*let found = false;*

*while (!found) {*

*prime++;*

*if (isPrime(prime))*

*found = true;*

*}*

*return prime-N;*

*}*

*// Main method*

*let N = 7*

*// this will return difference between next prime number x and x*

*document.write(nextPrime(N));*