KF6012 - Web Application Integration

2020

Comments

Northumbria University NEWCASTLE

Comments

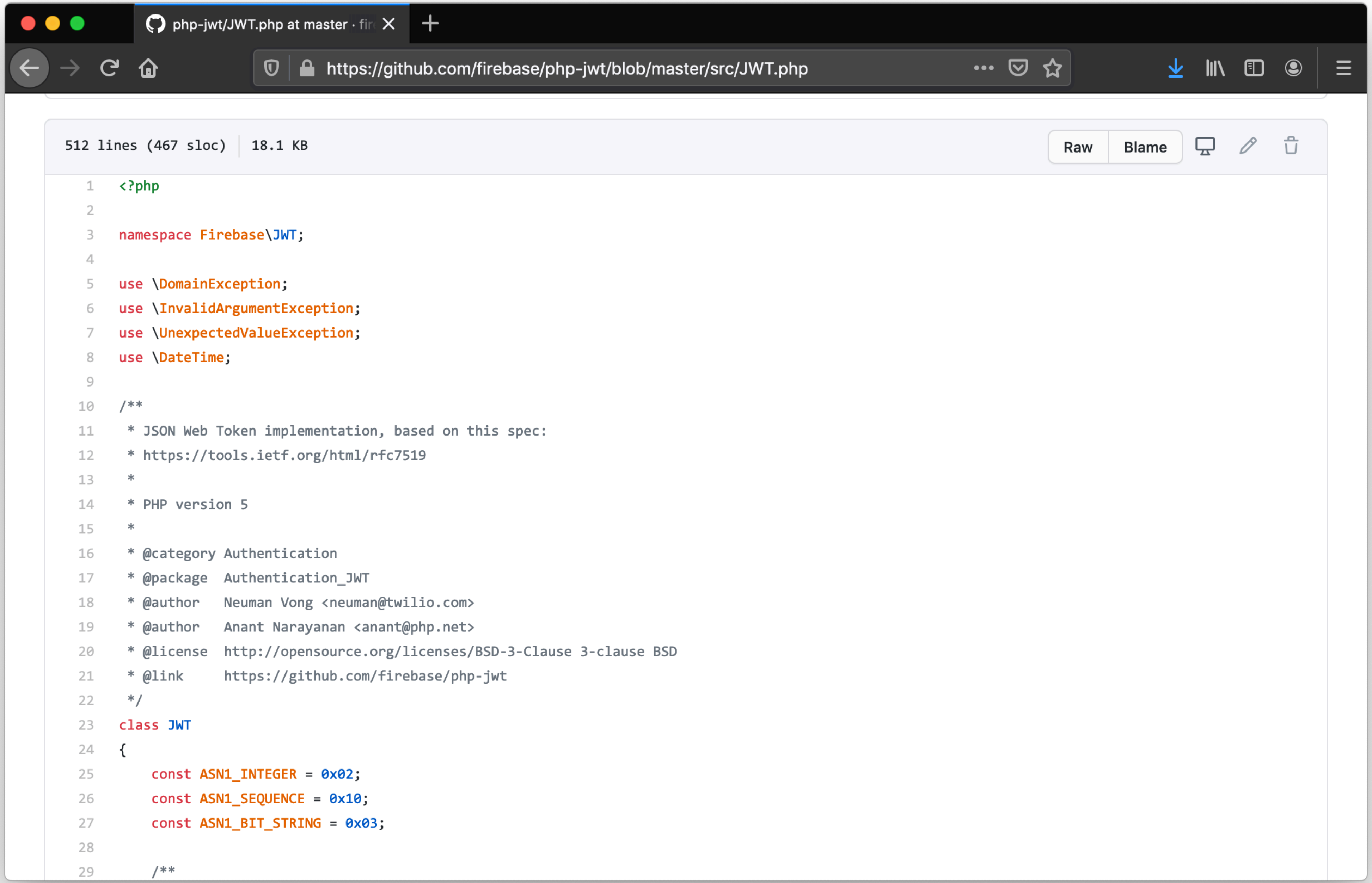
```
// inline comment

# inline comment

/*
	multi
	line
	comment
*/
```

“Doc Comments”

- In professional PHP programming a style called “DocComments” is used
- These are mainly just a convention for writing consistent and helpful comments
    - Comments that are helpful for people using your code
    - Comments that are helpful to people maintaining your code
    - Comments that are helpful to future you
- They can be used by documentation generators
- The are automatically generated by some code editors

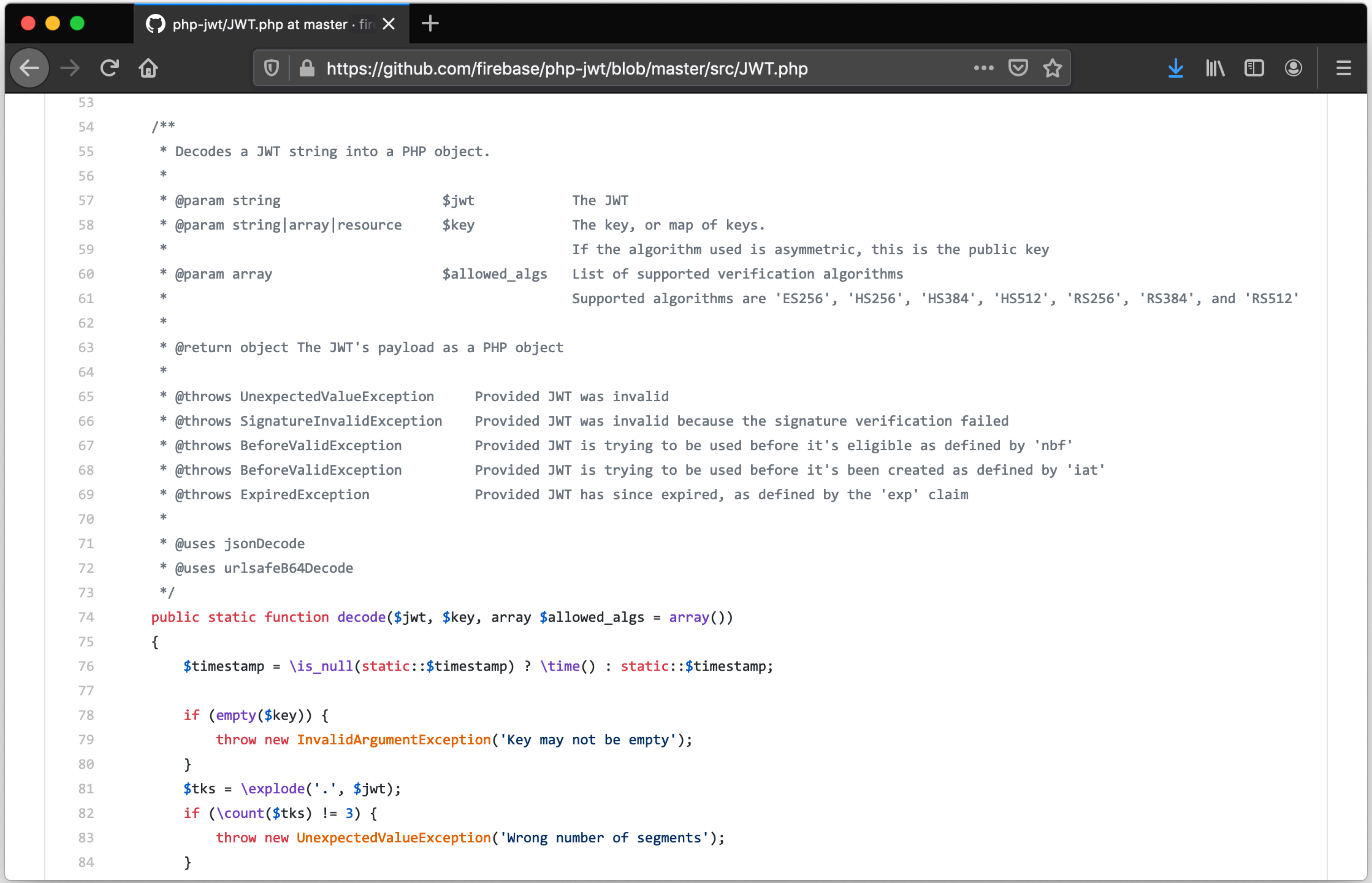
“Doc Comments”

```
/**
 * This is a summary
 *
 * This is a description
 *
 * @tag this is a tag
 */
[Class | Property | Method]
```

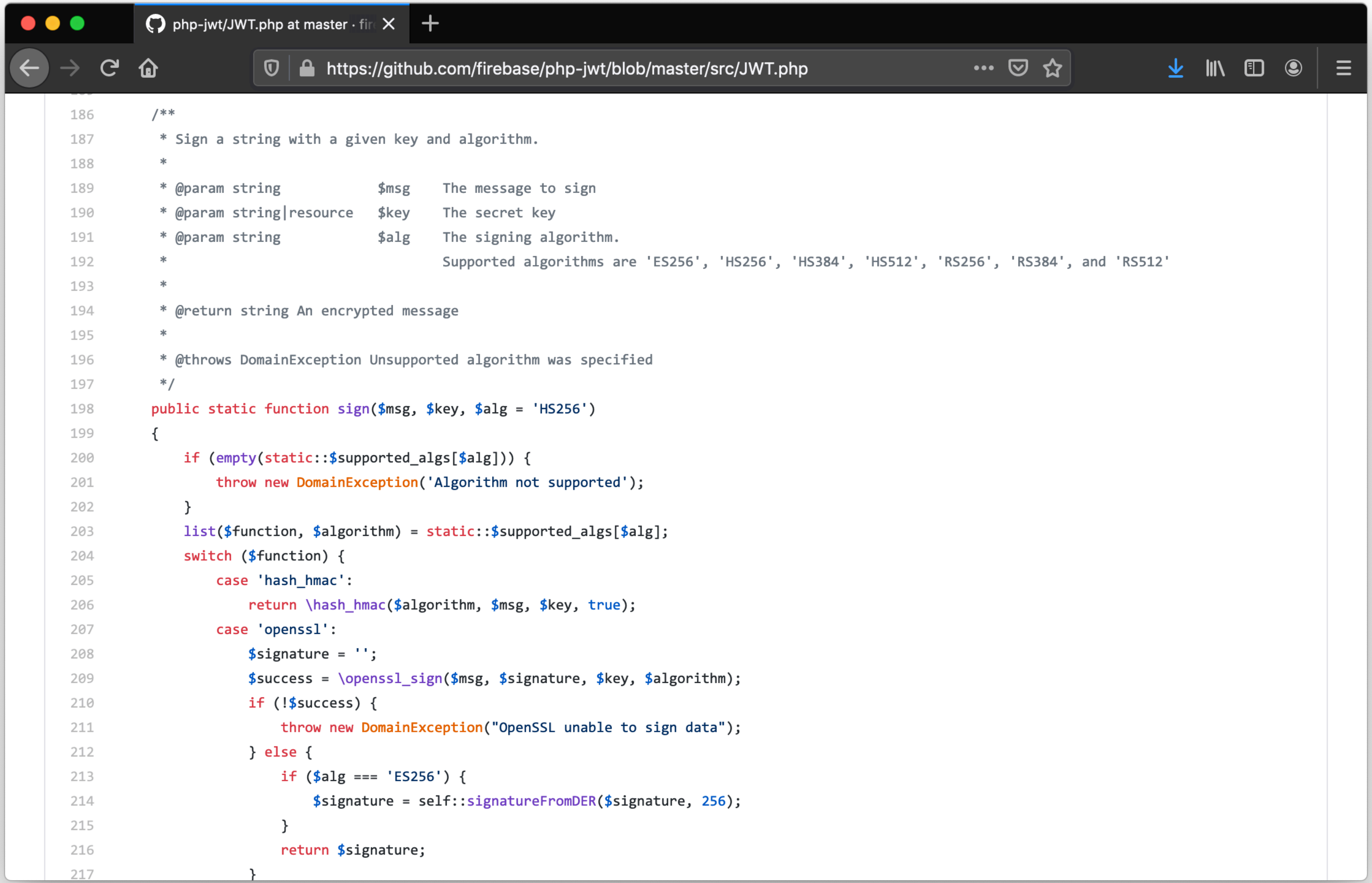

php-jwt/JWT.php at master · fir
https://github.com/firebase/php-jwt/blob/master/src/JWT.php
Raw
Blame

512 lines (467 sloc) | 18.1 KB

```php
<?php

namespace Firebase\JWT;

use \DomainException;
use \InvalidArgumentException;
use \UnexpectedValueException;
use \DateTime;

/**
 * JSON Web Token implementation, based on this spec:
 * https://tools.ietf.org/html/rfc7519
 *
 * PHP version 5
 *
 * @category Authentication
 * @package  Authentication_JWT
 * @author   Neuman Vong <neuman@twilio.com>
 * @author   Anant Narayanan <anant@php.net>
 * @license  http://opensource.org/licenses/BSD-3-Clause 3-clause BSD
 * @link     https://github.com/firebase/php-jwt
 */
class JWT
{
    const ASN1_INTEGER = 0x02;
    const ASN1_SEQUENCE = 0x10;
    const ASN1_BIT_STRING = 0x03;

    /**
```


php-jwt/JWT.php at master · fir
https://github.com/firebase/php-jwt/blob/master/src/JWT.php

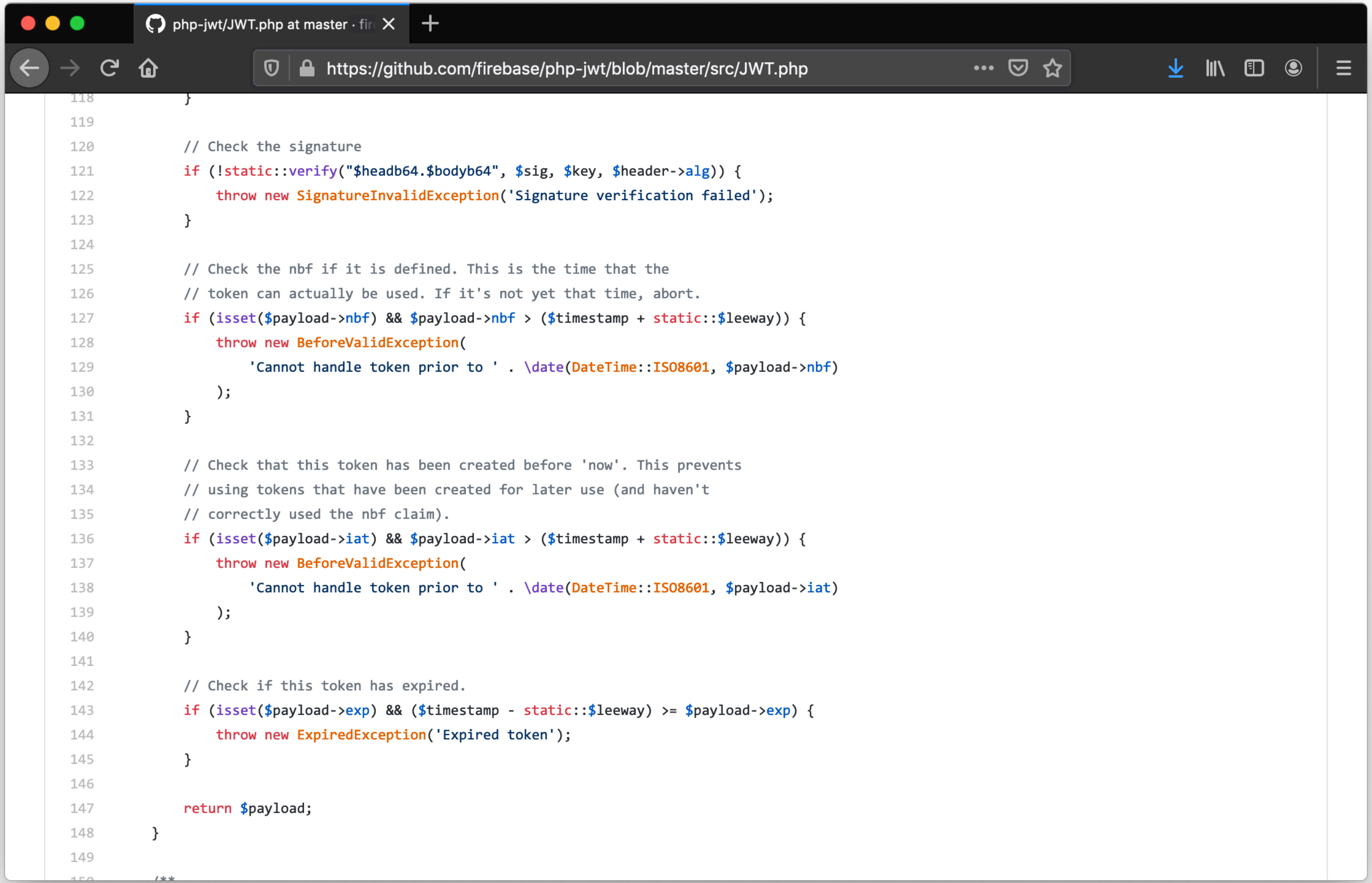
```php

    /**
     * Decodes a JWT string into a PHP object.
     *
     * @param string                    $jwt            The JWT
     * @param string|array|resource     $key            The key, or map of keys.
     *                                                  If the algorithm used is asymmetric, this is the public key
     * @param array                     $allowed_algs   List of supported verification algorithms
     *                                                  Supported algorithms are 'ES256', 'HS256', 'HS384', 'HS512', 'RS256', 'RS384', and 'RS512'
     *
     * @return object The JWT's payload as a PHP object
     *
     * @throws UnexpectedValueException     Provided JWT was invalid
     * @throws SignatureInvalidException    Provided JWT was invalid because the signature verification failed
     * @throws BeforeValidException         Provided JWT is trying to be used before it's eligible as defined by 'nbf'
     * @throws BeforeValidException         Provided JWT is trying to be used before it's been created as defined by 'iat'
     * @throws ExpiredException             Provided JWT has since expired, as defined by the 'exp' claim
     *
     * @uses jsonDecode
     * @uses urlsafeB64Decode
     */
    public static function decode($jwt, $key, array $allowed_algs = array())
    {
        $timestamp = \is_null(static::$timestamp) ? \time() : static::$timestamp;

        if (empty($key)) {
            throw new InvalidArgumentException('Key may not be empty');
        }
        $tks = \explode('.', $jwt);
        if (\count($tks) != 3) {
            throw new UnexpectedValueException('Wrong number of segments');
        }
```



php-jwt/JWT.php at master · fir

https://github.com/firebase/php-jwt/blob/master/src/JWT.php

```php
    /**
     * Sign a string with a given key and algorithm.
     *
     * @param string            $msg    The message to sign
     * @param string|resource   $key    The secret key
     * @param string            $alg    The signing algorithm.
     *                                  Supported algorithms are 'ES256', 'HS256', 'HS384', 'HS512', 'RS256', 'RS384', and 'RS512'
     *
     * @return string An encrypted message
     *
     * @throws DomainException Unsupported algorithm was specified
     */
    public static function sign($msg, $key, $alg = 'HS256')
    {
        if (empty(static::$supported_algs[$alg])) {
            throw new DomainException('Algorithm not supported');
        }
        list($function, $algorithm) = static::$supported_algs[$alg];
        switch ($function) {
            case 'hash_hmac':
                return \hash_hmac($algorithm, $msg, $key, true);
            case 'openssl':
                $signature = '';
                $success = \openssl_sign($msg, $signature, $key, $algorithm);
                if (!$success) {
                    throw new DomainException("OpenSSL unable to sign data");
                } else {
                    if ($alg === 'ES256') {
                        $signature = self::signatureFromDER($signature, 256);
                    }
                    return $signature;
                }
```



php-jwt/JWT.php at master · fir

https://github.com/firebase/php-jwt/blob/master/src/JWT.php

```
        }

        // Check the signature
        if (!static::verify("$headb64.$bodyb64", $sig, $key, $header->alg)) {
            throw new SignatureInvalidException('Signature verification failed');
        }

        // Check the nbf if it is defined. This is the time that the
        // token can actually be used. If it's not yet that time, abort.
        if (isset($payload->nbf) && $payload->nbf > ($timestamp + static::$leeway)) {
            throw new BeforeValidException(
                'Cannot handle token prior to ' . \date(DateTime::ISO8601, $payload->nbf)
            );
        }

        // Check that this token has been created before 'now'. This prevents
        // using tokens that have been created for later use (and haven't
        // correctly used the nbf claim).
        if (isset($payload->iat) && $payload->iat > ($timestamp + static::$leeway)) {
            throw new BeforeValidException(
                'Cannot handle token prior to ' . \date(DateTime::ISO8601, $payload->iat)
            );
        }

        // Check if this token has expired.
        if (isset($payload->exp) && ($timestamp - static::$leeway) >= $payload->exp) {
            throw new ExpiredException('Expired token');
        }

        return $payload;
    }

    /**
```

Comments and ”clean code”

- Don’t use comments where the code can explain itself
- Don’t add redundant comments.
- Don't use closing brace comments.
- Don't comment out code.
- Do use comments as explanation of intent and clarification of code.
- Do use comments as warning of consequences.

Thanks

John.rooksby@northumbria.ac.uk