```
const checker = Symbol( 'check' );
const isArray = Symbol( 'array' );
const isObject = Symbol( 'object' );
const isInteger = Symbol( 'integer' );
const isDouble = Symbol( 'double' );
const isString = Symbol( "string" );
const isFunction = Symbol( "function" );
const isClass = Symbol( "class" );
const isElement = Symbol( "element" ); //Returns a new unique Symbol value.
const isEmpty = Symbol( "empty" );
const isEmail = Symbol( "email" );
const isUrl = Symbol( "url" );
export default class Util {
    constructor() {}
    get version() {
        return "1.0.0";
    }
    get Array() {
        return "Array";
    }
    get Object() {
        return "Object";
    }
    get Integer() {
        return "Integer";
    }
    get Double() {
        return "Double";
    }
    get String() {
        return "String";
    }
    get Function() {
        return "Function";
    }
    get Class() {
        return "Class";
    }
    get Letter() {
        return "Letter";
    }
    get Empty() {
        return "Empty";
    }
    get Blank() {
        return "Blank";
    }
    get Element() {
        return "Element";
    }
    get Email() {
        return "Email";
    }
    get Url() {
        return "Url";
    }
    [ isArray ]( data ) {
        return Array.isArray( data );
    }
    [ isObject ]( data ) {
        return data instanceof Object && data.constructor === Object;
    }
    [ isInteger ]( data ) {
        let x;
        if ( isNaN( data ) ) {
            return false;
        }
        x = parseFloat( data );
        return ( x | 0 ) === x;
    }
    [ isDouble ]( data ) {
        let isNan = isNaN( data );
        let isDouble = false;
        if ( isNan ) {
            isDouble = false;
        } else {
            isDouble = !( Math.round( data ) === data );
        }
        return isDouble;
    }
    [ isString ]( data ) {
        return data.constructor === String && Object.prototype.toString.call( data ) === '[object
String]';
    }
    [ isFunction ]( data ) {
        let isFunc = ( ( Object.prototype.toString.call( data ) === '[object Function]' ||
                data.constructor === Function ) &&
            this.startsWith( data, '(' ) );
        return isFunc || this.startsWith( data, 'function' );
    }
    [ isClass ]( data ) {
        let isClass = ( ( Object.prototype.toString.call( data ) === '[object Function]' ||
            data.constructor === Function ) );
        return isClass && this.startsWith( data, 'class' );
    }
    [ isElement ]( data, type ) {
        let el = document.querySelectorAll( data );
        let result = false;
        el.forEach( e => {
            switch ( type ) {
                case 'visible':
                    result = window.getComputedStyle( e, null ).getPropertyValue( 'display' ) !==
'none' &&
                        window.getComputedStyle( e, null ).getPropertyValue( 'visibility' ) !==
'hidden';
                    break;
                case 'checked':
                    result = e.hasAttribute( 'checked' );
                    break;
                default:
                    break;
            }
        } );
        return result;
    }
```

```javascript
    [ isEmpty ]( data, type ) {
        let ret = false;
        switch ( type.toString() ) {
            case 'Empty':
                ret = ( !data || 0 === data.length )
                break;
            case 'Blank':
                ret = ( data.length === 0 || !data.trim() );
                break;
            default:
                ret = false;
                break;
        }
        return ret;
    }
    [ isElement ]( data ) {
        return data instanceof HTMLElement || data in HTMLElement;
    }
    [ isEmail ]( data ) {
        let re = /^(([^<>()\[\]\.,;:\s@\"]+(\.[^<>()\[\]\.,;:\s@\"]+)*)|(\".+\"))@(([^<>()[\]\.,;:\s@\"]+\.)+[^<>()[\]\.,;:\s@\"]{2,})$/i;
        return re.test( data.toLowerCase() );
    }
    [ isUrl ]( data ) {
        let re = /^(?:(?:https?|ftp):\/\/)?(?:(?!(?:10|127)(?:\.\d{1,3}){3})(?!(?:169\.254|192\.168)(?:\.\d{1,3}){2})(?!172\.(?:1[6-9]|2\d|3[0-1])(?:\.\d{1,3}){2})(?:[1-9]\d?|1\d\d|2[01]\d|22[0-3])(?:\.(?:1?\d{1,2}|2[0-4]\d|25[0-5])){2}(?:\.(?:[1-9]\d?|1\d\d|2[0-4]\d|25[0-4]))|(?:(?:[a-z\u00a1-\uffff0-9]-*)*[a-z\u00a1-\uffff0-9]+)(?:\.(?:[a-z\u00a1-\uffff0-9]-*)*[a-z\u00a1-\uffff0-9]+)*(?:\.(?:[a-z\u00a1-\uffff]{2,})))(?::\d{2,5})?(?:\/\S*)?$/i;
        return re.test( data.toLowerCase() );
    }
    [ checker ]( data, type ) {
        let elemArr = [ 'checked', 'visible' ]
        let types = {
            'Array': this[ isArray ]( data ),
            'Object': this[ isObject ]( data ),
            'Integer': this[ isInteger ]( data ),
            'Double': this[ isDouble ]( data ),
            'String': this[ isString ]( data ),
            'Function': this[ isFunction ]( data ),
            'Class': this[ isClass ]( data ),
            'Empty': this[ isEmpty ]( data, type ),
            'Blank': this[ isEmpty ]( data, type ),
            'Element': this[ isElement ]( data ),
            'Email': this[ isEmail ]( data ),
            'Url': this[ isUrl ]( data )
        }
        if ( elemArr.indexOf( type ) > -1 ) {
            type = 'Element';
            types[ 'Element' ] = his[ isElement ]( data, type );
        }
        return types[ type ];
    }
    startsWith( data, search, pos ) {
        return data.toString().substr( !pos || pos < 0 ? 0 : +pos, search.length ) === search;
    }
    format( strVal ) {
        let str = strVal.toString();
        if ( arguments.length ) {
            let t = typeof arguments[ 1 ];
            let key;
            let args = ( "string" === t || "number" === t ) ?
                Array.prototype.slice.call( arguments ) :
                arguments[ 1 ];
            for ( key in args ) {
                str = str.replace( new RegExp( "\\{" + key + "\\}", "gi" ), args[ key ] );
            }
        }
        return str;
    }
    is( data, type ) {
        return this[ checker ]( data, type );
    }
    includes( strVal, search ) {
        if ( !String.prototype.includes ) {
            return strVal.indexOf( search ) > -1;
        } else {
            return strVal.includes( search );
        }
    }
    count( str, search ) {
        let re = new RegExp( '(' + search + ')', 'g' );;
        let count = 0;
        try {
            count = str.match( re ).length;
        } catch ( error ) {
        }
        if ( search === this.Letter && !this.is( str, this.Array ) ) {
            count = str.length;
        } else if ( search === this.Array && this.is( str, this.Array ) ) {
            count = str.length;
        } else if ( search === this.Object && this.is( str, this.Object ) ) {
            count = Object.keys( str ).length;
        } else
            return count;
    }
    url( what = '', all = false ) {
        const props = [ 'hash', 'host', 'hostname', 'href', 'origin', 'pathname', 'port',
'protocol', 'search' ];
        let url = {}
        props.map( p => url[ p ] = document.location[ p ] );
        if ( !all ) {
            url = url[ what ];
        }
        return url;
    }
    page( data, prop ) {
        let title = document.title;
        let charset = document.characterSet;
        let doctype = document.doctype.name;
        let domain = document.domain;
        let location = document.location;
        let design = document.designMode;
        let scripts = Array.from( document.scripts ).map( m => m.src );
        let styles = Array.from( document.styleSheets ).filter( f => f.href ).map( m => m.href )
        let referrer = document.referrer;
        let modified = document.lastModified;
        let dataObj = {
            title: title,
            charset: charset,
            doctype: doctype,
            domain: domain,
            location: location,
            design: design,
            scripts: scripts,
            styles: styles,
            referrer: referrer,
            modified: modified
        }
        if ( !prop ) {
            return dataObj[ data ];
        }
        switch ( data ) {
            case 'title':
                document.title = prop;
                break;
            case 'charset':
                let checkCharset = document.querySelector( "meta[charset]" );
                try {
                    checkCharset.setAttribute( 'charset', prop );
                } catch ( Error ) {
                    let el = document.createElement( "meta" )
                    el.setAttribute( 'charset', prop );
                    document.head.insertBefore( el, document.head.firstElementChild );
                }
                break;
            case 'location':
                document.location = prop;
                break;
            case 'design':
                if ( prop === "on" || prop === "off" ) {
                    document.designMode = prop;
                } else {
                    throw "Prop must be on or off";
                }
                break;
            default:
                break;
        }
    }
    when( element, event, fn ) {
        if ( element === 'document' || element === 'body' ) {
            element = 'html';
        }
        let ev = event;
        let el = document.querySelectorAll( element );
        if ( element === 'html' && ( event === 'load' || event === 'DOMContentLoaded' ) ) {
            el = document.addEventListener( ev, ( data ) => {
                fn( data )
            } );
        } else {
            el.forEach( ( k, v ) => {
                let data = {
                    event: ev,
                    index: v,
                    html: el[ v ],
                    text: el[ v ].innerText,
                }
                el[ v ].addEventListener( ev, () => {
                    fn( data );
                } );
            } );
        }
    }
    where( data, column, search ) {
        let result = data.filter( m => m[ column ] === search );
        return result;
    }
}
const u = new Util();
console.log( u.format( 'Hi {0}. Did you see the {1}?', [ 'Pikachu', 'Meow' ] ) );
console.log( u.includes( 'Ali Baba', 'Baba' ) )
console.log( u.is( [ 1, 2, 3 ], u.Array ) );
console.log( u.is( {
    'a': 'v'
}, u.Object ) )
console.log( u.is( 5.1, u.Integer ) );
console.log( u.is( 3.2, u.Double ) );
console.log( u.is( 'Ali' + 123, u.String ) );
const a = ( d ) => {}
console.log( u.is( a, u.Function ) );
console.log( u.is( function a( d ) {}, u.Function ) );
console.log( u.is( class b {}, u.Class ) );
console.log( u.count( 'My name is no name when I do not like names. What is your name? Can u say your naming conversion', 'name' ) );
console.log( u.count( '😚 You now I\'ll be happy', u.Letter ) );
console.log( u.count( [ 1, 2, 3 ], u.Array ) );
console.log( u.count( {
    'w': 't',
    'w2': 't2',
    'wf': 'wf',
    'wfs': 'wfs2'
}, u.Object ) );
console.log( u.is( '#button', 'visible' ) );
console.log( u.is( "#chk", 'checked' ) );
console.log( u.is( "#rdb", 'checked' ) );
let t = '';
let b = ' ';
console.log( u.is( t, u.Empty ) );
console.log( u.is( b, u.Blank ) );
let anElement = '';
console.log( u.is( anElement, u.Element ) );
anElement = document.querySelector( "button" );
console.log( u.is( anElement, u.Element ) );
console.log( u.url( null, true ) );
console.log( u.url( 'search' ) );
console.log( u.is( 'goren.ali@yandex.com', u.Email ) );
console.log( u.is( 'tchİŞ@hotmaıl.com', u.Email ) );
console.log( u.is( 'tchİŞ@hotmaıl.', u.Email ) );
console.log( u.is( 'https://aligoren.com', u.Url ) );
console.log( u.is( 'ftp://github.com', u.Url ) );
console.log( u.is( 'http://localhost', u.Url ) );
console.log( u.is( 'www.google.com', u.Url ) );
console.log( u.page( 'title' ) )
u.page( 'title', 'New Page title' )
console.log( u.page( 'charset' ) )
u.page( 'charset', 'UTF-8' );
console.log( u.page( 'doctype' ) );
console.log( u.page( 'location' ) );
console.log( u.page( 'location', 'https://www.google.com.tr' ) );
console.log( u.page( 'design' ) );
u.page( 'design', 'on' || 'off' );
console.log( u.page( 'scripts' ) );
console.log( u.page( 'styles' ) );
console.log( u.page( 'referrer' ) );
console.log( u.page( 'modified' ) );
u.when( 'button', 'click', ( data ) => {
    console.log( data.event );
    console.log( data.index );
    console.log( data.text );
    console.log( data.html );
} );
u.when( 'document', 'click', ( data ) => {
    console.log( data.event );
    console.log( data.index );
    console.log( data.text );
    console.log( data.html );
} );
u.when( 'document', 'DOMContentLoaded', ( data ) => {
    console.log( data ); // from event
} );
u.when( 'document', 'load', ( data ) => {
    console.log( data ); // from event
} );
const arr = [ {
        name: 'test',
        surname: 'test1'
    },
    {
        name: 'test23',
        surname: 'newsname'
    },
    {
        name: 'test23',
        surname: 'newsname'
    }
]
console.log( u.where( arr, 'surname', 'newsname' ) );

```