# JavaScript + Angular

Part 2

Errors, DOM, RegExp, DateTime, AJAX

Author: Andrey Kucherenko

## Errors

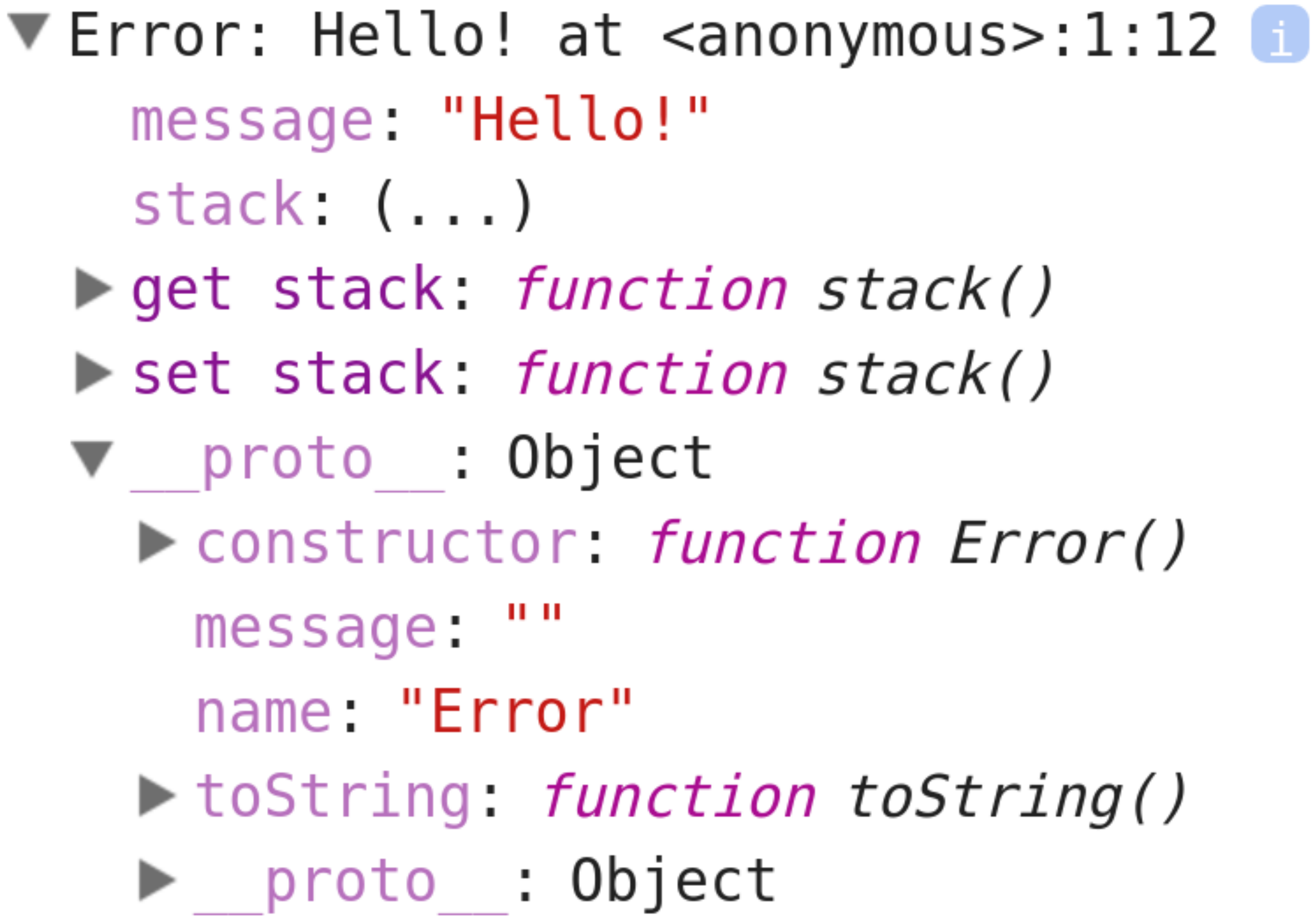
```
try {

  throw new Error('Hello Error');

} catch (err) {

}

try {
  setTimeout(function() {
    throw new Error('Async Error');
  }, 0);
} catch (e) {
  alert( "Error Error!" );
}
```


```
▼ Error: Hello! at <anonymous>:1:12
    message: "Hello!"
    stack: (...)
  ▶ get stack: function stack()
  ▶ set stack: function stack()
  ▼ __proto__: Object
    ▶ constructor: function Error()
      message: ""
      name: "Error"
    ▶ toString: function toString()
    ▶ __proto__: Object
```

```
function MyError(params) {
  Error.call(this, params) ;
  this.params = params;

  this.name = "MyError";
  this.stack = (new Error()).stack;
  this.message = "my error";
}

MyError.prototype = Object.create(Error.prototype);

try {
  throw new MyError('error params');
} catch (err) {
  if (err instanceof MyError) {
    alert(this.params);
  } else if (err instanceof SyntaxError) {
    alert( err.message );
  } else {
    throw err;
  }
}
```

```
<body onerror="handleError()">
    <script>
        window.onerror = handleError;
        function handleError() {
            alert('Error!');
        }
    </script>
    <script src="error.js"></script>
</body>
```

## Tasks

- Create your own Error
- Throw your error in seprate file
- Catch error with body onerror and window.onerror
- Catch with try/catch block

## Date Time

```
var date = new Date();
alert( date );

var date1 = new Date(1000 * 24 * 3600 * 10);

// new Date(year, month, date, hours, minutes, seconds, ms)
var date2 = new Date(2016, 6, 14, 12, 0, 0, 0);
```

```
Date.UTC()
Date.now()
Date.parse()

Date.prototype.getDate()
Date.prototype.getDay()
Date.prototype.getFullYear()
Date.prototype.getHours()
Date.prototype.getMilliseconds()
Date.prototype.getMinutes()
Date.prototype.getMonth()
Date.prototype.getSeconds()
Date.prototype.getTime()
Date.prototype.getTimezoneOffset()
Date.prototype.getUTCDate()
Date.prototype.getUTCDay()
Date.prototype.getUTCFullYear()
Date.prototype.getUTCHours()
Date.prototype.getUTCMilliseconds()
Date.prototype.getUTCMinutes()
Date.prototype.getUTCMonth()
Date.prototype.getUTCSeconds()
Date.prototype.getYear()
Date.prototype.setDate()
Date.prototype.setFullYear()
Date.prototype.setHours()
Date.prototype.setMilliseconds()
Date.prototype.setMinutes()
Date.prototype.setMonth()
Date.prototype.setSeconds()
Date.prototype.setTime()
Date.prototype.setUTCDate()
Date.prototype.setUTCFullYear()
Date.prototype.setUTCHours()
Date.prototype.setUTCMilliseconds()
Date.prototype.setUTCMinutes()
Date.prototype.setUTCMonth()
Date.prototype.setUTCSeconds()
Date.prototype.setYear()
Date.prototype.toDateString()
Date.prototype.toGMTString()
Date.prototype.toISOString()
Date.prototype.toJSON()
Date.prototype.toLocaleDateString()
Date.prototype.toLocaleFormat()
Date.prototype.toLocaleString()
Date.prototype.toLocaleTimeString()
Date.prototype.toSource()
Date.prototype.toString()
Date.prototype.toTimeString()
Date.prototype.toUTCString()
Date.prototype.valueOf()
```

## Tasks

- Create array of dates
- Sort array from old to new
- Create new object with keys "year-month-day" and array of times inside

## Timeouts

```
function hello_js() {
  alert( 'Hello JS' );
}

var timeout = setTimeout(hello_js, 1000);

function hello_js1(name) {
  alert( 'Hello' + name );
}

var timeout1 = setTimeout(hello_js1, 1000, 'JS');


clearTimeout(timeout1);
```

```
var timerId = setInterval(function() {
  alert( 'Hello!' );
}, 5000);


clearInterval(timerId);
```

## Tasks

- Write your own setInterval function
- Create console clock

## Regexp

```
var regexp = new RegExp("string pattern", "flags");

var regexp = /string pattern/;
var regexp = /string pattern/flags;

var str = "hello js"
var regexp = /js/;
console.log( str.search(regexp) );

/**
Flags:

i - not case sensitive
g - all not first
m - multi-line

**/
```

```
str.search(reg);
str.match(reg);
str.split(reg);
str.replace(reg);

var str = "Hello JS";

alert(str.replace(/(Hello) (JS)/, '$2, $1'));

var str = "Hello JS";

alert(str.replace(/Hello JS/, 'Hello $&!'));

alert("Hello JS JS".replace(/js/gi, function(str) {
  return str + '!';
}));

function replac_func(str, greet, js, offset, s) {
  return greet + "  " + name + "!";
}
```

```
RegExp.prototype.exec() //
RegExp.prototype.test() // = str.search()

var str = "Hello JS";
var patt = new RegExp("e");
var res = patt.exec(str);
```

```
/**
[1-9] - all digits
[az] - a or z
[^az] - not a and not z
{n}, {n,m} - count of symbols

\d - digits [0-9]
\s - space [\t\n\v\f\r ]
\w - worlds [a-zA-Z0-9_]

\b - border

\D - not digit [^0-9]
\S - not space [^\s]
\W - not words

. - any symbol except new line

+ - {1,}
? - {0,1}
* - {0,}

^ - start of string
$  - end of string
/<\/?[a-z][a-z0-9]*>/i
**/
```

## Tasks

- Split string by number
- Find all e-mails in string
- find all internal scripts in html

## DOM

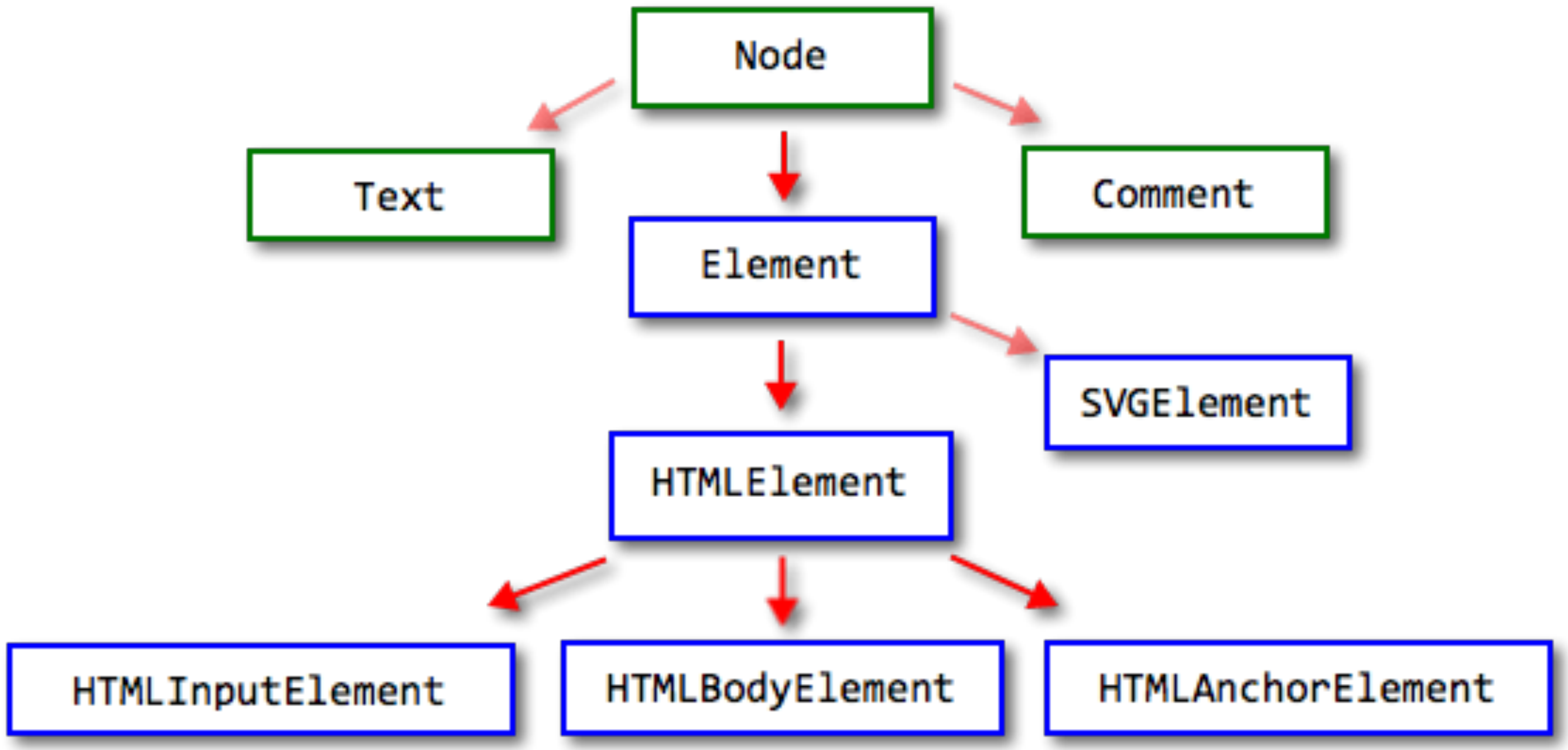
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title></title>
    <script>
        var doc = document;
    </script>
  </head>
  <body>

  </body>
</html>
```

```
document.documentElement = <html>
document.body = <body>

document.parentNode
document.body.childNodes //collection of nodes(!Array)
document.body.firstChild
document.body.lastChild
document.body.previousSibling
document.body.nextSibling

document.body.parentElement
document.body.children // collection of elements
document.body.firstElementChild
document.body.lastElementChild
document.body.previousElementSibling
document.body.nextElementSibling
```


Node
Text
Element
Comment
SVGElement
HTMLElement
HTMLInputElement
HTMLBodyElement
HTMLAnchorElement

```
Element
    Properties
        accessKey
        attributes
        childElementCount
        children
        classList
        className
        clientHeight
        clientLeft
        clientTop
        clientWidth
        currentStyle
        firstElementChild
        id
        innerHTML
        lastElementChild
        localName
        name
        namespaceURI
        nextElementSibling
        ongotpointercapture
        onlostpointercapture
        onwheel
        outerHTML
        prefix
        previousElementSibling
        runtimeStyle
        scrollHeight
        scrollLeft
        scrollLeftMax
        scrollTop
        scrollTopMax
        scrollWidth
        tabStop
```

```
Element
    Methods
        after()
        animate()
        append()
        before()
        closest()
        getAttribute()
        getAttributeNode()
        getAttributeNodeNS()
        getAttributeNS()
        getBoundingClientRect()
        getClientRects()
        getElementsByClassName()
        getElementsByTagName()
        getElementsByTagNameNS()
        hasAttribute()
        hasAttributeNS()
        hasAttributes()
        insertAdjacentElement()
        insertAdjacentHTML()
        insertAdjacentText()
        matches()
        prepend()
        querySelector()
        querySelectorAll()
        releasePointerCapture()
        remove()
        removeAttribute()
        removeAttributeNode()
        removeAttributeNS()
        replaceWith()
        requestFullscreen()
        requestPointerLock()
        scrollIntoView()
```

```
HTMLElement
    Properties
        contentEditable
        dataset
        dir
        isContentEditable
        lang
        offsetHeight
        offsetLeft
        offsetParent
        offsetTop
        offsetWidth
        onabort
        onblur
        onchange
        onclick
        onclose
        oncontextmenu
        oncopy
        oncut
        ondblclick
        onerror
        onfocus
        oninput
        onkeydown
        onkeypress
        onkeyup
        onload
        onmousedown
        onmousemove
        onmouseout
        onmouseover
        onmouseup
        onpaste
        onpointercancel
```

```
HTMLElement
     Methods
        blur()
        click()
        focus()
        forceSpellCheck()
```

```
document.body.innerHTML = 'hello';
document.body.outerHTML
document.body.textContent

var div = document.createElement('div');
div.className = "message";
div.innerHTML = "<div>!!!</div>";

document.body.appendChild(div);

var div2 = div.cloneNode(true);
document.body.insertBefore(div2, document.body.children[0]);

parentElem.removeChild(elem)
parentElem.replaceChild(newElem, elem)
```

```
var myElement = document.getElementById('my_id');

var divs = document.getElementsByTagName('div');

var blocks = document.getElementsByClassName('blocks');



var items = document.querySelectorAll('div > ul > li:firstChild');

var el = elem.querySelector('.article');

var p_el = elem.closest('div.content');

if (el.matches('a[target="_blank"]')){ ... }
```

```
var result = document.evaluate(
    "/body//div[contains(., 'JS')]",
    document.documentElement,
    null,
    XPathResult.ORDERED_NODE_SNAPSHOT_TYPE,
    null
);

for (var i = 0; i < result.snapshotLength; i++) {
  console.log( result.snapshotItem(i).outerHTML );
}
```

## Tasks

- Create html file with different types of nodes
- Iterate all divs as collection
- Iterate all links with querySelector
- Create new node at end of body
- Make 10 clones of first div and append it to start of body
- Replace last div with span
- From object tree make visual tree

## Events

```
<input onclick="sayHey()" type="button" value="Click!" />
<input id="el" type="button" value="Click!!!" />
<script>
  var el = document.getElementById('el');
  elem.onclick = sayHey;

  function sayHey() {
    alert('Hey!');
  };

  el.addEventListener('click', sayHey, true);
  el.removeEventListener('click', sayHey);
</script>
```

```
var el = document.getElementById('el');
function sayHey(event) {
   console.log(event);
 };

el.addEventListener('click', sayHey);

Event
    Properties
        bubbles
        cancelable
        currentTarget
        defaultPrevented
        eventPhase
        isTrusted
        srcElement
        target
        timeStamp
        type
    Methods
        createEvent()
        preventDefault()
        stopImmediatePropagation()
        stopPropagation()
```

```
<button id="elem" onclick="alert('!!!');">Self clicker</button>

<script>
  var event = new Event("click");
  elem.dispatchEvent(event);
</script>
```

## Tasks

- Make tree of elements
- Each clicks on element should highlight all parent elements
- Click on last item of last list

## Ajax/Comet

```javascript
var xhr = new XMLHttpRequest();

xhr.open('GET', 'list.json', false);
xhr.send();

if (xhr.status != 200) {
  alert( xhr.status + ': ' + xhr.statusText );
} else {
  alert( xhr.responseText );
}

var myImage = document.querySelector('img');

window.fetch('flowers.jpg')
    .then(function(response) {
      return response.blob();
    })
    .then(function(myBlob) {
      var objectURL = URL.createObjectURL(myBlob);
      myImage.src = objectURL;
    });
```

```
XMLHttpRequest
    Methods
        open(method, url, async, user, password)
        send(body)
        abort()
        setRequestHeader(name, value)
        getResponseHeader(name)
        getAllResponseHeaders()
    Properties
        timeout
        responseText
        responseXML
        status
        statusText
        onreadystatechange
        ontimeout
        onerror
        onload
        onprogress
        onabort
        onloadstart
        onloadend
```

```
var socket = new WebSocket("ws://{url}");

socket.onmessage = function(event) {
  console.log(event.data);
};

socket.send(outgoingMessage);
```

```javascript
var WebSocketServer = new require('ws');

var clients = {};

var webSocketServer = new WebSocketServer.Server({
  port: 8081
});

webSocketServer.on('connection', function(ws) {

  var id = Math.random();
  clients[id] = ws;
  console.log("connection..." + id);

  ws.on('message', function(message) {
    console.log(message);

    for (var key in clients) {
      clients[key].send(message);
    }
  });

  ws.on('close', function() {
    console.log('connection closed.. ' + id);
    delete clients[id];
  });

});
```

## Tasks

- Create call to server for data, show error if resource not found
- Put json on server and try to get it with different functions
- Create persistent connection between server and client

## ES6/2015

http://babeljs.io/repl/

```
const Z = 5;
let z, x, a = 5;

if (a) {
    let b = 5;
}

[z, x] = ['z', 'x']

let [, , js] = "hello my js".split(' ');
let [h, ...all] = "hello my js".split(' ');

let [a=5] = [];

let options = {
  width: 300,
  height: 500,
  arr: [1, 2]
};
let {width, height} = options;
let {width: w, height: h, name="Andrey", arr: [a1, a2]} = options;
```

```javascript
function a(p = 3) {} // a.name == "a"

function b(...all) {}

if (true) {
  z();
  function z() {
    alert("!!!");
  }
}
z();


let inc = x => x+1;

let sum = (a, b) => {
    return a + b; // no own this, no arguments
}
```

```
let str = `ololo`;

let str = `ololo

    ololo

    olol`;

let zz = `${str} hello`;
```

```javascript
class User {

  constructor(name) {
    console.log(super);
    this.name = name;
  }

  say() {
    alert(this.name);
  }

  static createGuest() {
    return new User();
  }

}

let User = class {
  say() { alert('!!!'); }
};

new User().say();

class A extends B {}
```

```
let name = "Adam";
let isAdmin = true;

let user = {
  name,
  isAdmin
};

let prop = 'role';
let user = {
  [prop]: "admin"
};


let user = {
  get role() {
    return `SuperUser`;
  }
};

alert( user.role );

let sym = Symbol();
alert( typeof sym ); // symbol
```

```javascript
let arr = ["a", 'b'];
for (let value of arr) {
  alert(value);
}

let myIterator = {
  from: 1,
  to: 5
}

myIterator[Symbol.iterator] = function() {

  let current = this.from;
  let last = this.to;

  return {
    next() {
      if (current <= last) {
        return {
          done: false,
          value: current++
        };
      } else {
        return {
          done: true
        };
      }
    }
  }
};
alert(...myIterator)
for (let num of myIterator) {
  alert(num);
}
```

```
let map = new Map();

map.set('1', 'string');
map
    .set(1, 'number')
    .set(true, 'boolean');


alert( map.get(1)   );
alert( map.get('1') );

alert( map.size );

map.keys();
map.values();
map.entries();

let map = new Map([
  ['1',  'str1'],
  [1,    'num1'],
  [true, 'bool1']
]);

map.has('1');
map.delete('1');
map.clear();
```

```
let set = new Set();

set.add(1);
set.add(2);
set.add(3);
set.add(3);
set.add(3);

alert( set.size ); // 3

set.forEach( item => alert(item) );

set.clear();
set.delete(1);
set.has(3);
```

```
let arr = [
  {test: '1'},
  {test: '2'},
  {test: '3'}
];

let weakSet = new WeakSet();//WeakMap

weakSet.add(arr[0]);
weakSet.add(arr[1]);
weakSet.add(arr[2]);

delete arr[0];

alert(weakSet.size)
```

```
var promise = new Promise(function(resolve

})

promise.then(success, error);

promise.catch(errorHandler);

Promise.race(array);
Promise.all(array);
```

```javascript
//Generator
function* gen() {
  yield 'a';
  yield 'b';
  return 'c';
}
let generator = gen();
let one = generator.next();
```