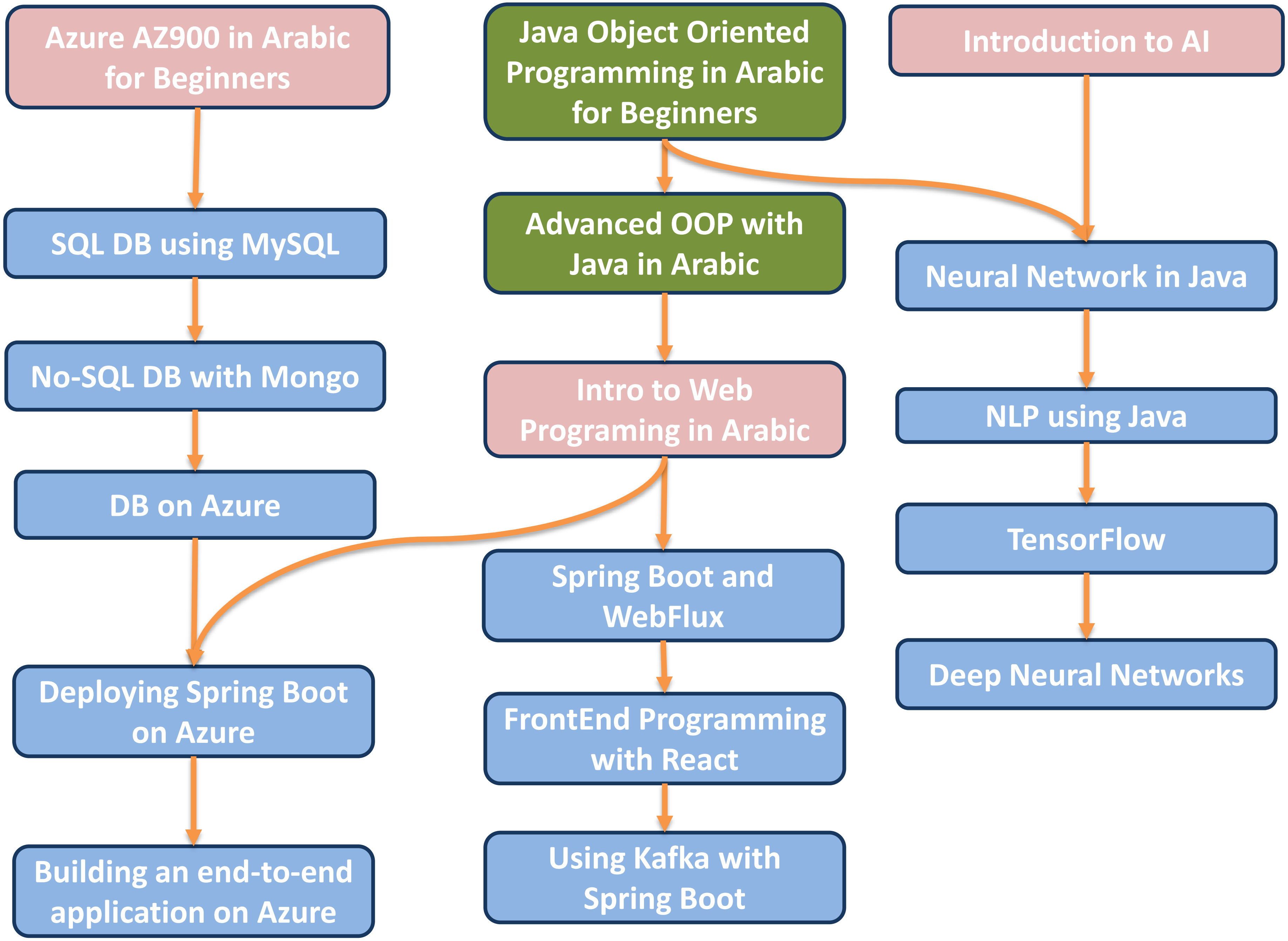

Azure AZ900 in Arabic for Beginners
SQL DB using MySQL
No-SQL DB with Mongo
DB on Azure
Deploying Spring Boot on Azure
Building an end-to-end application on Azure
Java Object Oriented Programming in Arabic for Beginners
Advanced OOP with Java in Arabic
Intro to Web Programing in Arabic
Spring Boot and WebFlux
FrontEnd Programming with React
Using Kafka with Spring Boot
Introduction to AI
Neural Network in Java
NLP using Java
TensorFlow
Deep Neural Networks

Advanced JavaScript

Dr. Magdi AMER

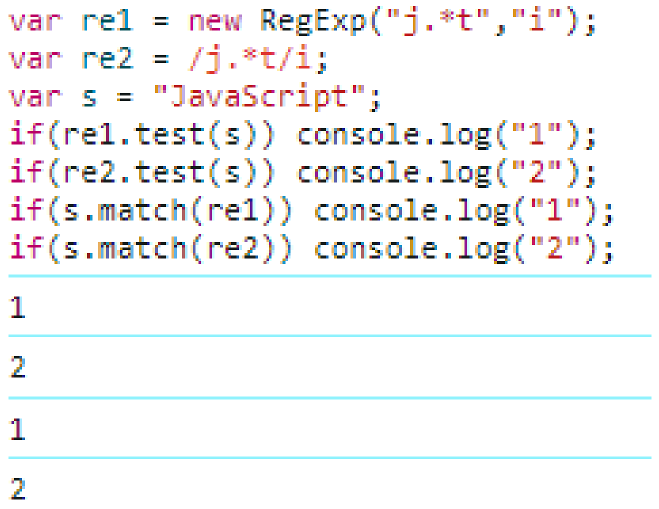
Regular Expression

Regular Expression

```
var re1 = new RegExp("j.*t","i");
var re2 = /j.*t/i;
var s = "JavaScript";
if(re1.test(s)) console.log("1");
if(re2.test(s)) console.log("2");
if(s.match(re1)) console.log("1");
if(s.match(re2)) console.log("2");
```

/j.*t/; ➔ case sensitive

/j.*t/i; ➔ case insensitive



```
var re1 = new RegExp("j.*t","i");
var re2 = /j.*t/i;
var s = "JavaScript";
if(re1.test(s)) console.log("1");
if(re2.test(s)) console.log("2");
if(s.match(re1)) console.log("1");
if(s.match(re2)) console.log("2");
```

```
1
2
1
2
```

.	Any single character
[ ]	Any character listed
[a-zA-Z]	Any English character
[^]	Any character not listed
^	Start of a string
$	End of a string
+	1 or more
*	0 or more
?	0 or 1
\	To escape meta-character
|	OR
{m,n}	Minimum m, max n
{m}	Exactly m times
\t	a tab character
\n	a newline character
\r	a carriage-r eturn character
\s	matches any whitespace character (space, tab, newline, etc..)
\S	anything not \s
\w	[a-zA-Z0-9R]
\W	anything not \w, i.e., [^a-zA-Z0-9R]
\d	[0-9], i.e., a digit
\D	anything not \d, i.e., [^0-9]
\b	Word boundary
\B	Not a word boundary
\num	Octal representation of a character
\xnum	Octal representation of a character

abc

abc (that exact character sequence, but anywhere in the string)

^abc

abc at the beginning of the string

abc$

abc at the end of the string

a|b

either of a and b

^abc|abc$

the string abc at the beginning or at the end of the string

ab{2,4}c

an a followed by two, three or four b’s followed by a c

ab{2,}c

an a followed by at least two b’s followed by a c

ab*c

an a followed by any number (zero or more) of b’s followed by a c

ab+c

an a followed by one or more b’s followed by a c

ab?c

an a followed by an optional b followed by a c; that is, either abc or ac

a.C

an a followed by any single character (not newline) followed by a c a\.c

a.c exactly

[abc]

any one of a, b and c

[Aa]bc

either of Abc and abc

[abc]+

any (nonempty) string of a’s, b’s and c’s (such as a, abba, acbabcacaa)

[^abc]+

any (nonempty) string which does not contain any of a, b and c (such as defg)

\d\d

any two decimal digits, such as 42; same as \d{2}

\w+

a “word”: a nonempty sequence of alphanumeric characters and underscores such as foo and 12bar8 and foo_1

100\s*mk

the strings 100 and mk optionally separated by any amount of white space

abc\b

abc when followed by a word boundary (e.g. in abc! but not in abcd)

Java\B

Java when not followed by a word boundary (e.g. in Javascript but not in Java Beans)

The string contains from 5 to 8 characters, no more.

```
^([^a-zA-Z])*([a-zA-Z][^a-zA-Z]*){5,8}([^a-zA-Z])*$
```

functions

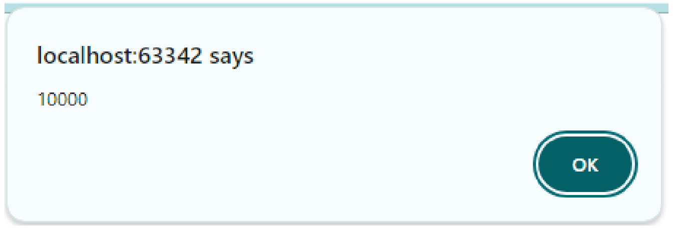
Functions

Both these functions are equivalent

```
function sandwich(bread, meat)
{
alert(bread + meat + bread);
}

function sandwich2()
{
alert(arguments[0] + arguments[1] + arguments[0]);
}
```

Functions Arguments

```
<!DOCTYPE html>
<html lang="en">
<head>    <meta charset="UTF-8">    <title>Title</title> </head>
<body>
<script type='text/javascript' >
function max()
{
var max = Number.NEGATIVE_INFINITY;
for(var i = 0; i < arguments.length; i++)
{
if (arguments[i] > max)
max = arguments[i];
}
return max;
}
var largest = max(1, 10, 100, 2, 3, 1000, 4, 5, 10000, 6);
alert(largest);
</script>
</body>
</html>
```



```
localhost:63342 says
10000
```

Pointer to functions

```
function f(){return 1;}
var f = function(){return 1;}
>>> typeof f
```

```
"function"
```

functions

```
function sum(a, b) {
var c = a + b;
return c;
}
```

- `parseInt()`
- `parseFloat()`
- `isNaN()`
- `isFinite()`
- `encodeURI()`
- `decodeURI()`
- `encodeURIComponent()`
- `decodeURIComponent()`
- `eval()`

```
var str = "for(var i =0;i<3;i++) alert(i) ";
eval(str);
```

```
>>> var url = 'http://www.packtpub.com/scr ipt.php?q=this and that';
>>> encodeURI(url);
```

"http://www.packtpub.com/scr%20ipt.php?q=this%20and%20that"

```
>>> encodeURIComponent(url);
```

"http%3A%2F%2Fwww.packtpub.com%2Fscr%20ipt.php%3Fq%3Dthis%20and%20that"

functions

```
function f(){return 1;}
var f = function(){return 1;}
>>> typeof f
```

"function"

```
var sum = function(a, b) {return a + b;}
var add = sum;
delete sum
```

try... catch

```
function validatePassword(password)
{
 try
 {
  //Make sure password has at least 5 characters
  if(password.length < 5 )
  {
    throw "SHORT";
   }
  //Make sure password has no more than 10 characters
  if(password.length > 10 )
  {
    throw "LONG"; //too many characters
    }
   //Password ok
   alert("Password Validated!");
  }
```

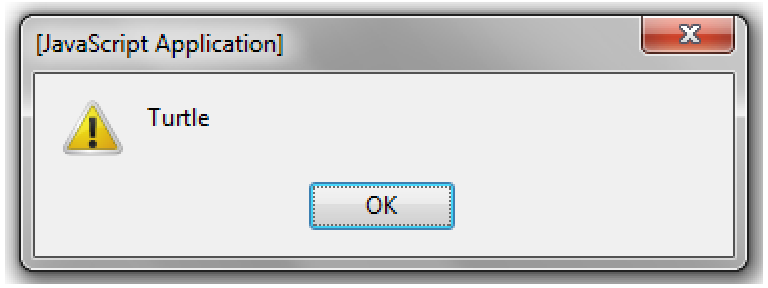
try... catch

```
catch(e)
 {
  if(e == "SHORT")
  {
    alert("Not enough characters in password!");
    }
  if(e == "LONG")
  {
    alert("Password contains too many characters!");
    }
  }
}

var pass = "123"
validatePassword(pass);
```

Object Oriented JavaScript

Object

```
var hero = {
breed: 'Turtle',
occupation: 'Ninja'
};

alert(hero.breed);
```


[JavaScript Application]

Turtle

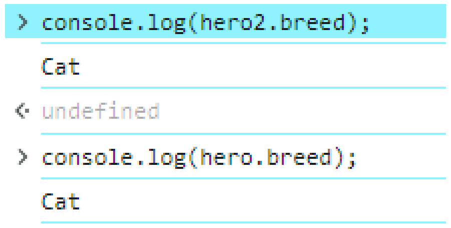
```
var hero = {};
//another way
//var hero= new Object();
hero.breed = 'turtle';
hero.name = 'Leonardo';
hero.sayName = function()
{alert( hero.name);};

hero.sayName();
alert(hero['breed']);
```

```
var dog = {
name: 'Benji',
talk: function(){
alert('Woof, woof!');
}
};
dog.talk();
```

```
var book = {
name: 'Catch-22',
published: 1961,
author: {
firstname: 'Joseph',
lastname: 'Heller'
}
};
```

Object

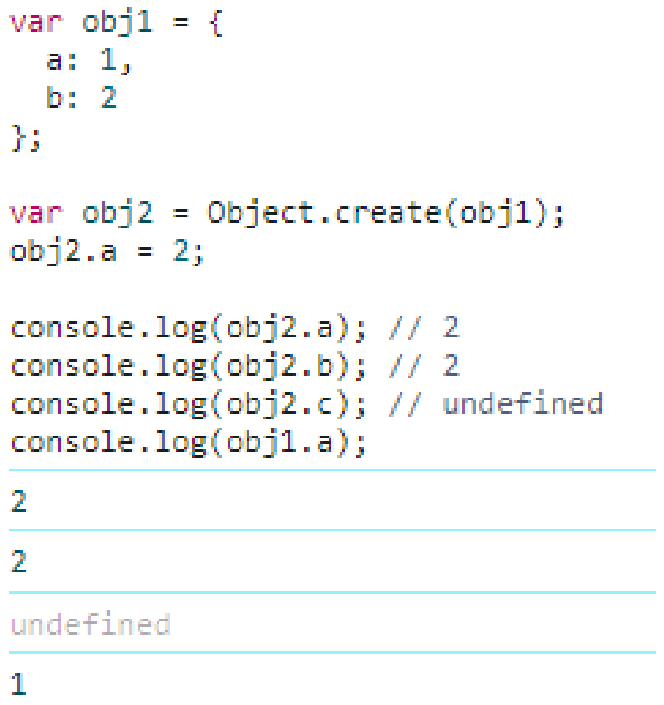
```
var hero = {
breed: 'Turtle',
occupation: 'Ninja'
};

alert(hero.breed);
var hero2 = hero;
hero2.breed = 'Cat';
console.log(hero2.breed);
console.log(hero.breed);
```



```
> console.log(hero2.breed);
  Cat
<· undefined
> console.log(hero.breed);
  Cat
```

Object

```
var obj1 = {
  a: 1,
  b: 2
};

var obj2 = Object.create(obj1);
obj2.a = 2;

console.log(obj2.a); // 2
console.log(obj2.b); // 2
console.log(obj2.c); // undefined
console.log(obj1.a);
```

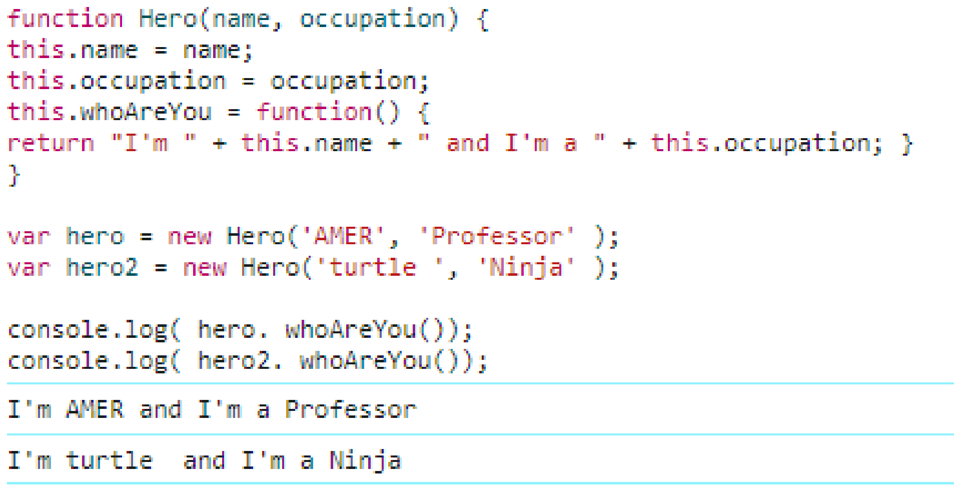


```
var obj1 = {
  a: 1,
  b: 2
};

var obj2 = Object.create(obj1);
obj2.a = 2;

console.log(obj2.a); // 2
console.log(obj2.b); // 2
console.log(obj2.c); // undefined
console.log(obj1.a);
```

```
2
2
undefined
1
```

Object

```
function Hero(name, occupation) {
this.name = name;
this.occupation = occupation;
this.whoAreYou = function() {
return "I'm " + this.name + " and I'm a " +
this.occupation; }
}

var hero = new Hero('AMER', 'Professor' );
var hero2 = new Hero('turtle ', 'Ninja' );

console.log( hero. whoAreYou());
console.log( hero2. whoAreYou());
```



```
function Hero(name, occupation) {
this.name = name;
this.occupation = occupation;
this.whoAreYou = function() {
return "I'm " + this.name + " and I'm a " + this.occupation; }
}

var hero = new Hero('AMER', 'Professor' );
var hero2 = new Hero('turtle ', 'Ninja' );

console.log( hero. whoAreYou());
console.log( hero2. whoAreYou());
```

```
I'm AMER and I'm a Professor
I'm turtle  and I'm a Ninja
```

instanceof

```
>>> function Hero(){}
>>> var h = new Hero();
>>> var o = {};
>>> h instanceof Hero;
```

true

```
>>> h instanceof Object;
```

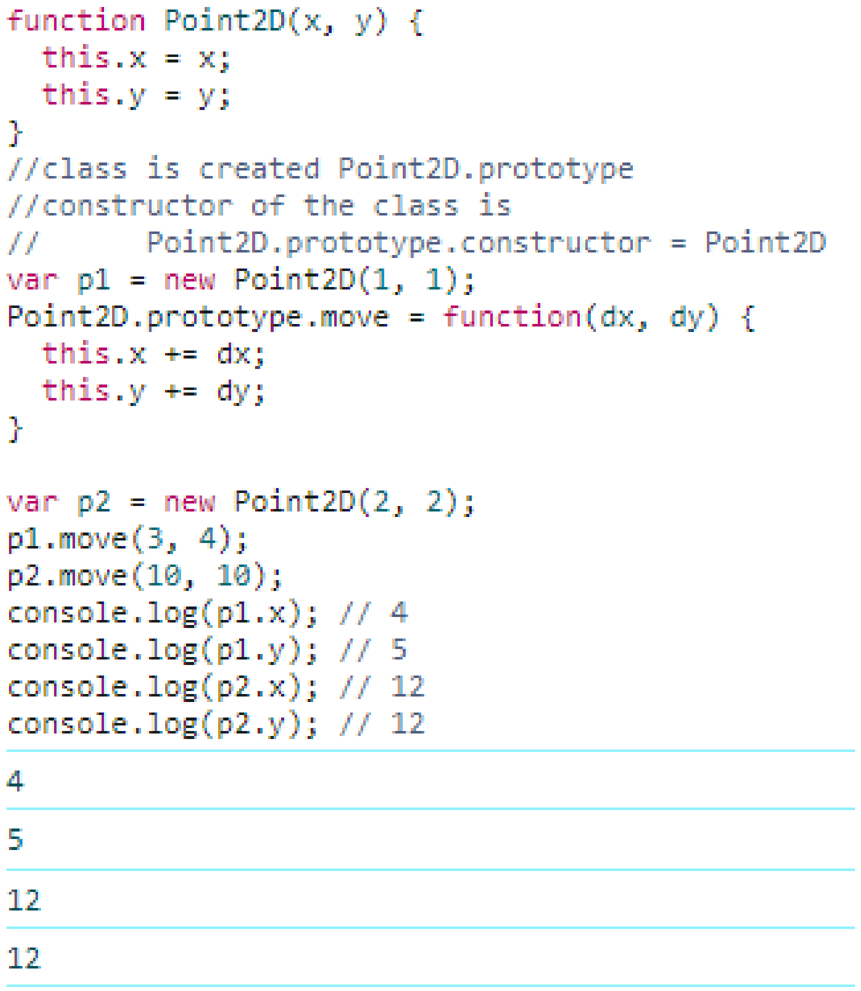
false

```
>>> o instanceof Object;
```

true

prototype

```
function Point2D(x, y) {
  this.x = x;
  this.y = y;
}
//class is created Point2D.prototype
//constructor of the class is
//      Point2D.prototype.constructor = Point2D
var p1 = new Point2D(1, 1);
Point2D.prototype.move = function(dx, dy) {
  this.x += dx;
  this.y += dy;
}

var p2 = new Point2D(2, 2);
p1.move(3, 4);
p2.move(10, 10);
console.log(p1.x); // 4
console.log(p1.y); // 5
console.log(p2.x); // 12
console.log(p2.y); // 12
```

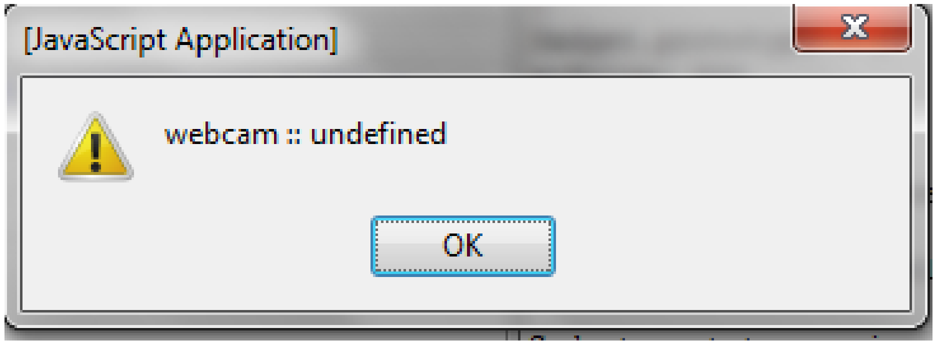


```
function Point2D(x, y) {
  this.x = x;
  this.y = y;
}
//class is created Point2D.prototype
//constructor of the class is
//      Point2D.prototype.constructor = Point2D
var p1 = new Point2D(1, 1);
Point2D.prototype.move = function(dx, dy) {
  this.x += dx;
  this.y += dy;
}

var p2 = new Point2D(2, 2);
p1.move(3, 4);
p2.move(10, 10);
console.log(p1.x); // 4
console.log(p1.y); // 5
console.log(p2.x); // 12
console.log(p2.y); // 12
```

```
4
5
12
12
```

prototype

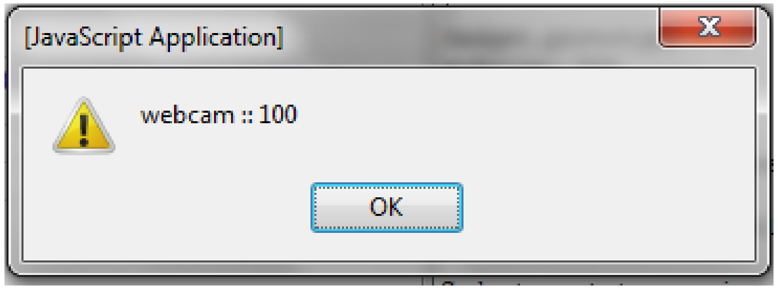
```
function Gadget(name, color) {
this.name = name;
this.color = color;
this.whatAreYou = function(){
return 'I am a ' + this.color + ' ' + this.name;
}
}
Gadget.prototype = {
myPrice: 200
};

var t1 = new Gadget('webcam', 'black');
t1.myPrice = 45;
alert(t1.name+" :: "+t1.price);
```



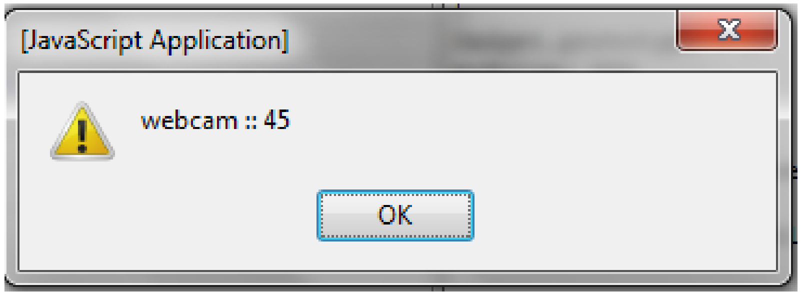
```
webcam :: undefined
```

prototype

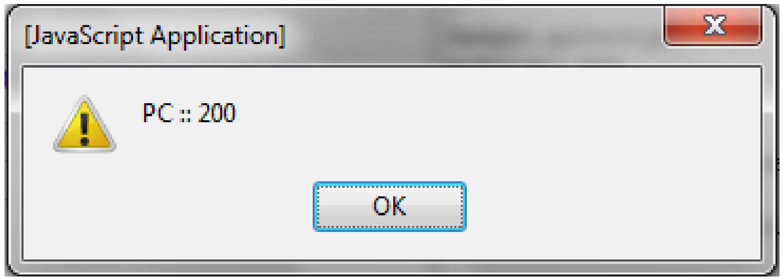
```
Gadget.prototype.price = 100;
Gadget.prototype.rating = 3;
Gadget.prototype.getInfo = function() {
return 'Rating: ' + this.rating +
   ', price: ' + this.price;
};

alert(t1.name+" :: "+t1.price);
var t2 = new Gadget('webcam', 'black');
t2.myPrice = 45;
alert(t2.name+" :: "+t2.myPrice);
var t3 = new Gadget('PC', 'red');
alert(t3.name+" :: "+t3.myPrice);
```


[JavaScript Application]

webcam :: 100


[JavaScript Application]

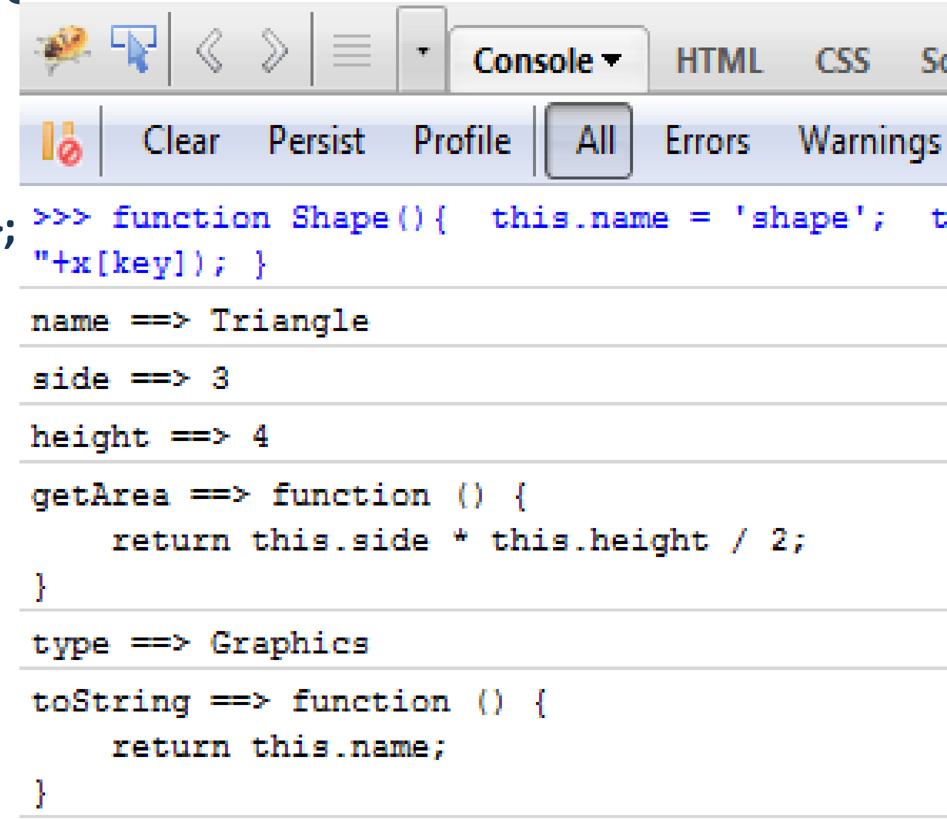
webcam :: 45


[JavaScript Application]

PC :: 200

inheritance

```
function Shape(){
this.name = 'shape';
this.type='Graphics';
this.toString = function() {return this.name;};
}
function TwoDShape(){
this.name = '2D shape';
}
TwoDShape.prototype = new Shape();

function Triangle(side, height) {
this.name = 'Triangle';
this.side = side;
this.height = height;
this.getArea = function(){return this.side * this.height / 2;};
}
Triangle.prototype = new TwoDShape();

var x = new Triangle(3, 4);
for (var key in x)
{  console.log(key+" ==> "+x[key]);   }
```



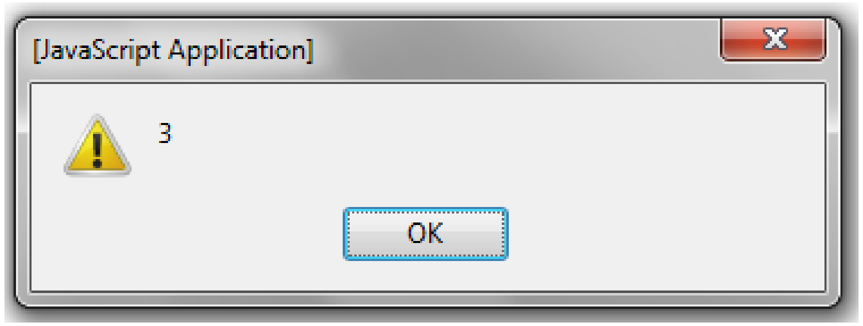
```
>>> function Shape(){  this.name = 'shape';  t
"+x[key]); }
name ==> Triangle
side ==> 3
height ==> 4
getArea ==> function () {
    return this.side * this.height / 2;
}
type ==> Graphics
toString ==> function () {
    return this.name;
}
```

AJAX and fetch

Callback function

Because a function is just like any other data assigned to a variable, it can be defined, deleted, copied, and why not also *passed as an argument to other functions?*

```
function invoke_and_add(a, b){
return a() + b();
}
function one() {
return 1;
}
function two() {
return 2;
}
var x = invoke_and_add(one, two);
alert(x);
```



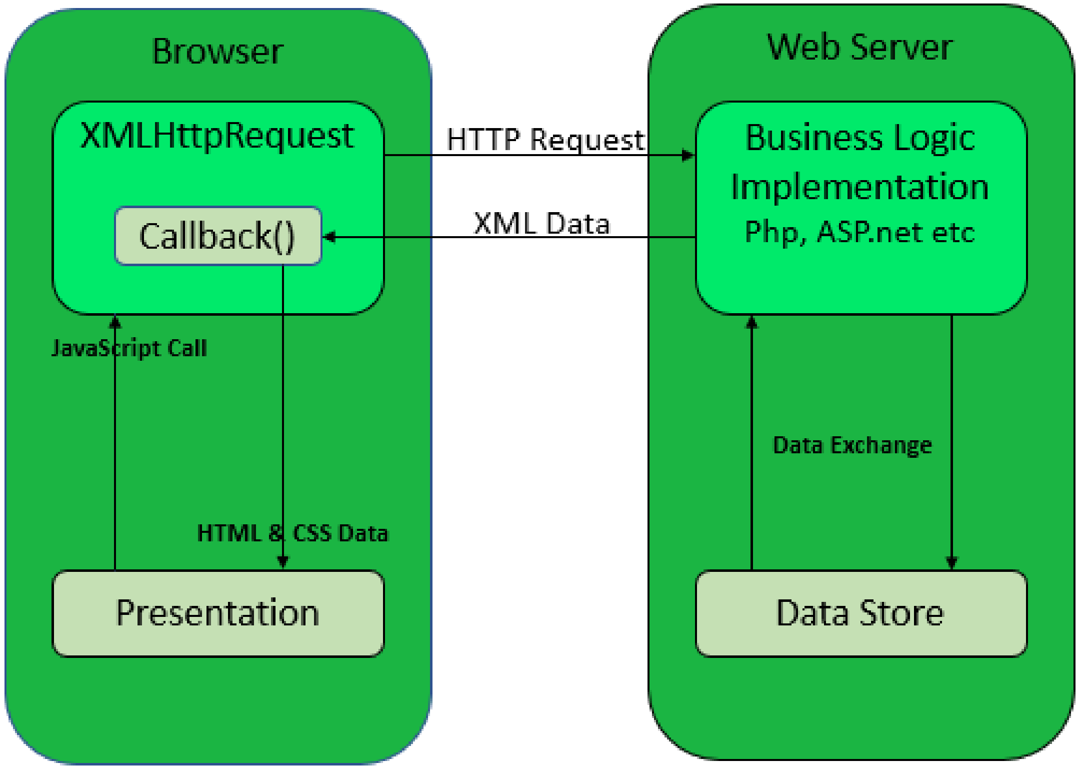
3

Callback function

```
function Hero(name, method) {
this.name = name;
this.fn = method;
this.callBack = function() {
this.fn(); }
}

var sayHello = function ()
{  alert("hello"); }

var hero = new Hero('AMER', sayHello );
hero.callBack();
```

AJAX overview

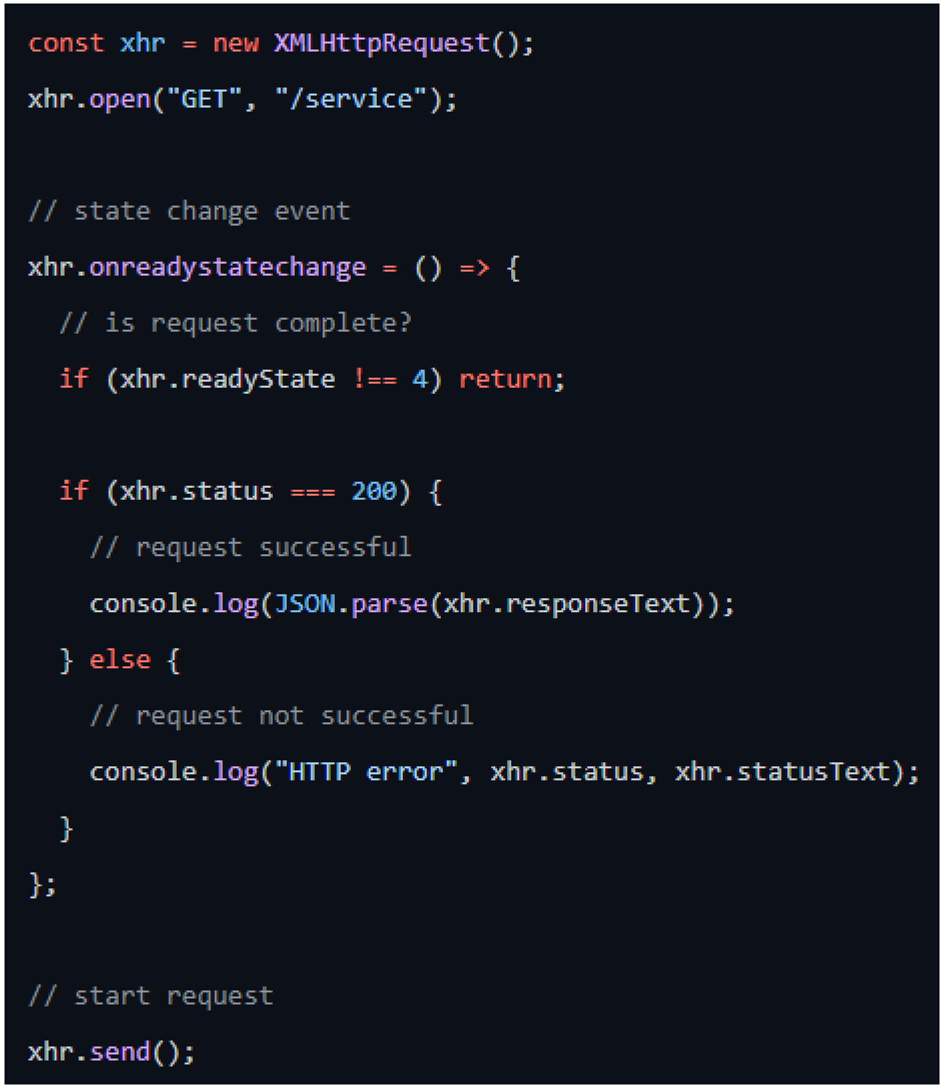
Asynchronous ***J***avaScript ***A***nd ***X***ML



AJAX overview

```
<!DOCTYPE html>
<html>
<body>

<div id="demo">
<h2>The XMLHttpRequest Object</h2>
<button type="button" onclick="loadDoc()">Change Content</button>
</div>

<script>
function loadDoc() {
  const xhttp = new XMLHttpRequest();
  xhttp.onload = function() {
    document.getElementById("demo").innerHTML =
    this.responseText;
  }
  xhttp.open("GET", "ajax_info.txt");
  xhttp.send();
}
</script>

</body>
</html>
```

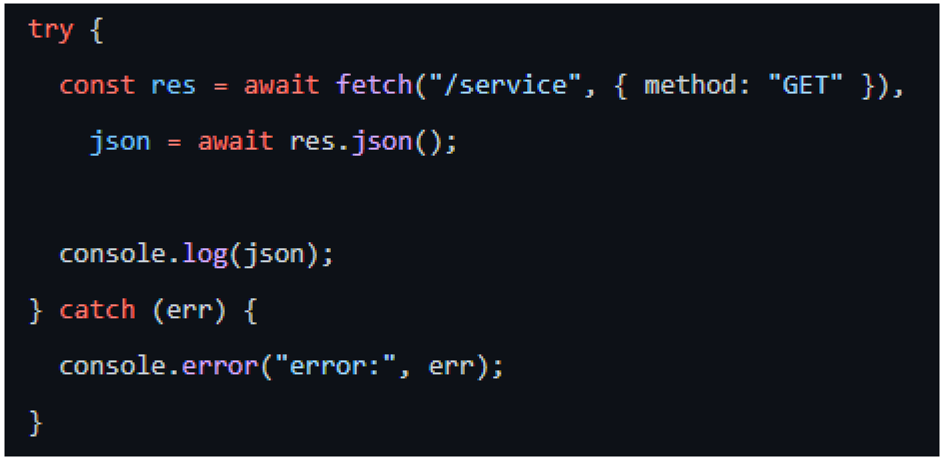
AJAX overview



```
const xhr = new XMLHttpRequest();
xhr.open("GET", "/service");

// state change event
xhr.onreadystatechange = () => {
  // is request complete?
  if (xhr.readyState !== 4) return;

  if (xhr.status === 200) {
    // request successful
    console.log(JSON.parse(xhr.responseText));
  } else {
    // request not successful
    console.log("HTTP error", xhr.status, xhr.statusText);
  }
};

// start request
xhr.send();
```

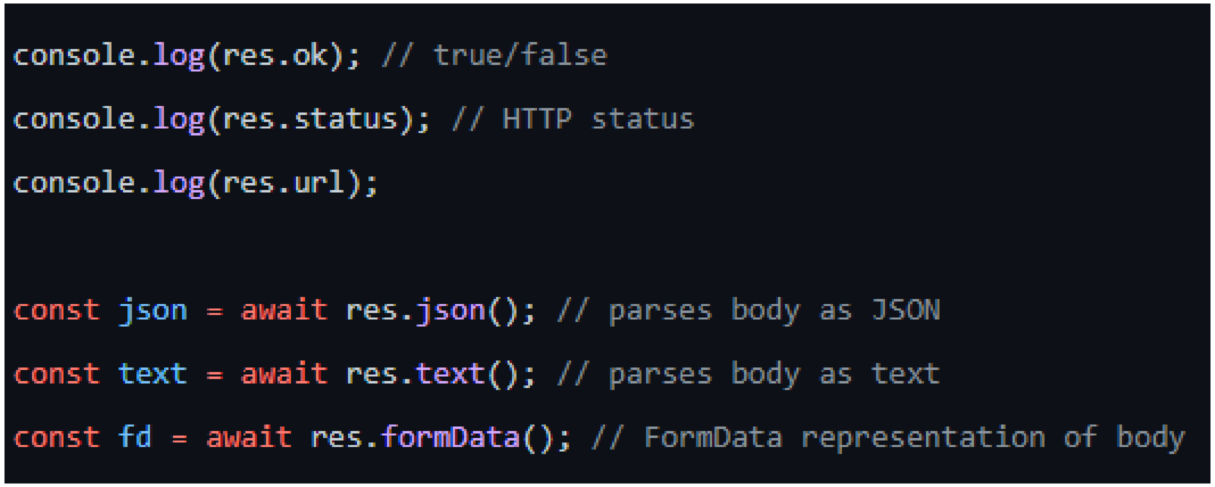
•0 (uninitialized) - request not initialized
•1 (loading) - server connection established
•2 (loaded) - request received
•3 (interactive) - processing request
•4 (complete) - request complete, response is ready

fetch



```
try {
  const res = await fetch("/service", { method: "GET" }),
    json = await res.json();

  console.log(json);
} catch (err) {
  console.error("error:", err);
}
```

```
fetch('url', {
  Method: 'POST',
  Headers: {
    Accept: 'application.json',
    'Content-Type': 'application/json'
  },
  Body: body,
  Cache: 'default'
})
```

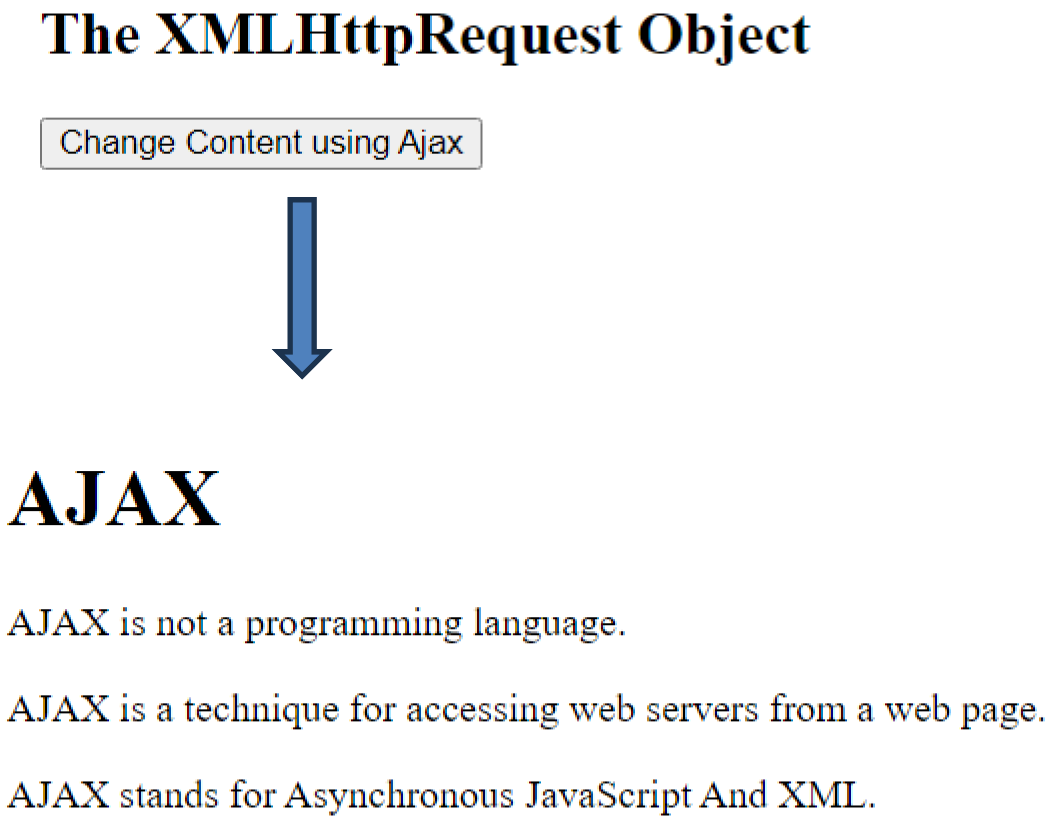


```
console.log(res.ok); // true/false
console.log(res.status); // HTTP status
console.log(res.url);

const json = await res.json(); // parses body as JSON
const text = await res.text(); // parses body as text
const fd = await res.formData(); // FormData representation of body
```

Ajax Vs fetch

```
<!DOCTYPE html>
<html>
<body>

<div id="demo">
<h2>The XMLHttpRequest Object</h2>
<button type="button" onclick="loadDoc()">Change Content using Ajax</button>
</div>

<script>
function loadDoc() {
  const xhttp = new XMLHttpRequest();
  xhttp.onload = function() {
    document.getElementById("demo").innerHTML =
    this.responseText;
  }
  xhttp.open("GET", "ajax_info.txt");
  xhttp.send();
}
</script>

</body>
</html>
```

Ajax Vs fetch



The XMLHttpRequest Object

AJAX

AJAX is not a programming language.

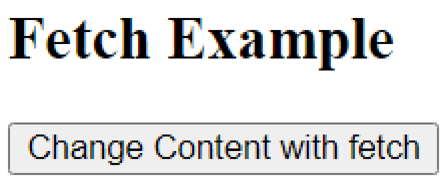
AJAX is a technique for accessing web servers from a web page.

AJAX stands for Asynchronous JavaScript And XML.

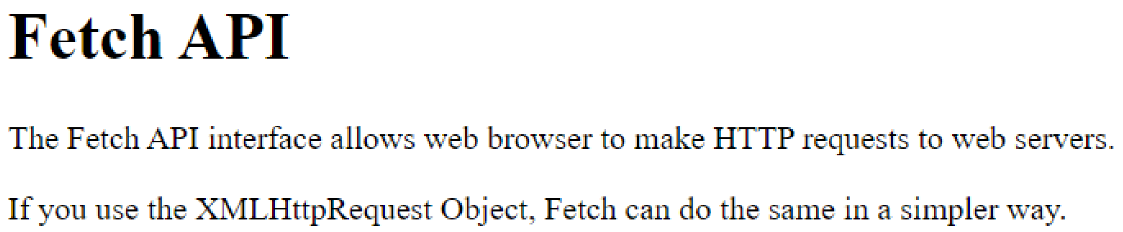
Ajax Vs fetch

```
<!DOCTYPE html>
<html>
<body>
<div id="demo">
<h2>Fetch Example</h2>
<button type="button" onclick="loadDoc()">Change Content with fetch</button>
</div>

<script>
async function loadDoc() {
  let x = await fetch("fetch_info.txt");
  let y = await x.text();
  document.getElementById("demo").innerHTML = y;
}
</script>

</body>
</html>
```

Ajax Vs fetch



Fetch Example



Fetch API

The Fetch API interface allows web browser to make HTTP requests to web servers.

If you use the XMLHttpRequest Object, Fetch can do the same in a simpler way.