

7/12/2017

Variables and datatypes

- **Number**: Floating Point numbers for decimals and integers.

Var name = 'John';

- **String**: Sequence of characters, used for text.

Console.log (name);

- **Boolean**: Logical data type that can only be true or false.

Var lastName = 'smith';

- **undefined**: Data type of a variable which does not have a value yet.

Var age = 26;

- **null**: Also means 'non-existent'.

Console.log (age);

Var fullAge = true;

Console.log (fullAge);

Variable mutation and Type Casting

// single line comment
/* multiline
comment
*/

} it is useful to describe the content
(2) stop the code.

Concatenation

Var name = 'John'

Var age = 26;

Console.log (name + age); // John26

Console.log (age + age); // 52

Var job, isMarried;

Console.log (job); // undefined

job = 'teacher';

isMarried = false;

Console.log (name + ' is ' + age + ' years old ' + job + ' is ' + isMarried +
. . .);

age = 'Thirty six';

job = 'driver';

Var lastName = prompt('What is your lastname?');

Console.log (lastName);

Experience certainty.

IT Services
Business Solutions
Consultingalert (name + ' is ' + age + ' years old ' + job + ' is ' + isMarried +
. . .);

Operators

Var BirthYear = 2016 - 26;

birthYear = 2016 - 26 * 2;

Console.log (birthYear);

Var ageJohn = 30;

Var ageMark = 30;

ageJohn = (3 + 5) * 6;

ageJohn = ageMark = (3 + 5) * 4 - 6;

Console.log (ageJohn); // 26

Console.log (ageMark); // 26

ageJohn ++;

Console.log (ageJohn);

ageMark *= 2 // 52 (ageMark = ageMark * 2);

if/else statement

which allows us to code based on the decisions

Var name = 'John';

Var age = 26;

Var isMarried = 'No';

if (isMarried == 'Yes') {

 Console.log(name + ' is married!');

} else {

 Console.log(name + ' will hopefully marry soon.:)');

}

isMarried = false

if (isMarried) {

 Console.log('Yes');

} else {

 Console.log('No');

}

if (`23 == "23"`) {

`console.log('something to print...');` // print

{

if (`23 == "23"`)

`console.log('something to print...');` // won't print

{

Boolean logic & Switch Statement

`var age = 16;`

if (`age < 20`) {

`console.log('John is a teenager');` // printed

{ else {

`console.log('John is a man');`

{

- AND (`&&`) \Rightarrow true & & true

- OR (`||`) \Rightarrow true if one is true

- NOT (`!`) \Rightarrow inverts true / false value

Var age 16;

age >= 20; //=> false

age < 30; //=> true

! (age < 30); //=> false

age >= 20 && age < 30; //=> false

age >= 20 || age < 30; //=> true

if (age < 20) {

Console.log ('John is a teenager');

} else if (age > 20 && age < 30) {

Console.log ('John is a Youngman');

} else {

Console.log ('John is a man');

}

switch statement /

Var job = 'teacher';

job = prompt ("what does john do?");

switch (job) {

case 'teacher':

console.log ('john teaches kids.');

break;

case 'driver':

console.log ('John drives a cab in Lisbon.');

break;

case 'cop':

console.log ('John helps fight crime.');

break;

default:

console.log ('John does something else.');

}

Coding Challenge

John and a friend invented a simple game where the player with the highest value of his height (in centimeters) plus five times his age wins (what a silly game);

1. Create Variable for the height and ages of two friends and assign them some values
2. Calculate their scores
3. Describe who wins and print the winner to the console. Include the score in the string that you output to the console.
Don't forget that there can be draw (both players with the same score).
4. Extra: Add a third player and now describe who wins.
Hint: You will need the `<=` operator to take the decision.
~~If you can't solve this one~~

solution for code challenge

Var heightJohn = 176;

Var heightMark = 168;

Var heightJane = 156;

Var ageJohn = 27;

Var ageMark = 29;

Var ageJane = 30;

Var scoreJohn = heightJohn + ageJohn * 5;

Var scoreMark = heightMark + ageMark * 5;

Var scoreJane = heightJane + ageJane * 5;

if (scoreJohn > scoreMark) {
 if (scoreJohn > scoreJane) {
 console.log ('John won the game with ' + scoreJohn + ' points');
 } else if (scoreMark > scoreJohn) {
 console.log ('Mark won the game with ' + scoreMark + ' points');
 } else if (scoreMark == scoreJohn && scoreJohn == scoreJane && scoreJane == scoreMark) {
 console.log ('It's a draw try it again');
 }
}

elseif (scoreJane > scoreJohn && scoreJane > scoreMark) {
 console.log ("Jane won the game with " + scoreJane + " points");
}

DRY - Don't repeat yourself

Functions

Function calculateAge (~~name~~, yearOfBirth) {
 var age = 2017 - yearOfBirth;
 ~~console.log~~ return age;

var ageJohn = calculateAge (~~John~~, 1990);

function calculateAge (name, yearOfBirth) {

var age = 2017 - yearOfBirth;

console.log (name + ' is ' + age + ' years old');

}

calculateAge ('John', 1990);

calculateAge ('Mark', 1996);

calculateAge ('Jane', 1948);

```
function calculateAge(YearOfBirth) {  
    var age = 2016 - YearOfBirth;  
    return age;  
}
```

}

```
var age_john = calculateAge(1990);  
var age_mike = calculateAge(1969);  
var age_mary = calculateAge(1948);
```

```
function yearsUntilRetirement(name, year) {
```

```
    var age = calculateAge(year);
```

```
    var retirement = 65 - age;
```

```
    if (retirement >= 0) {
```

```
        console.log(name + ' retires in ' + retirement + ' years.');
```

```
} else {
```

```
    console.log(name + ' is already retired!');
```

}

```
yearsUntilRetirement('John', 1990);
```

```
yearsUntilRetirement('Mike', 1969);
```

```
yearsUntilRetirement('Mary', 1948);
```

080 - 25533995
8470024365
705784289
326646-

Statement & Expressions

// Function statement

function someFun (par) {

// code

3

if (x == 5) {

• // do something

3

// Function expressions

Var somFun = function (par) {

// code

3

3 + 4

Var x = 3;

Arrays

Var names = ['John', 'Jane', 'Mark'];

Var years = new Array(1990, 1969, 1948);

Console.log(names[0]);

names[1] = 'Ben';

Console.log(names);

```
var John = ['John', 'smith', 1999, 'teacher', false];
```

```
John.push('blue');
```

```
console.log(John);
```

```
John.unshift('mr.'), John.pop(); John.shift();  
console.log(John);
```

```
var a = John.indexOf('smith');  
console.log(a);
```

```
if(John.indexOf('teacher') == -1) {
```

```
    console.log('John is not a teacher')
```

3

Objects and Properties

Var arr = [1, 2, 3]

this is for array

arr[0];

For object we don't have particular order

Object literal

Var john = {

name: 'John',

lastName: 'Smith',

YearOfBirth: 1990,

job: 'Teacher'

isMarried: false

}

Object
literal

console.log(john);

(1)

console.log(john.lastName);

(2)

console.log(john['lastName']);

manipulating mutation

john.lastName = 'miller';

john['job'] = 'Programmer';

Var xyz = 'Job',

console.log(john[xyz]);

Console.log(john);

var Jane = new Object();

Jane.name = 'Jane';

Jane.lastName = 'Smith';

Jane['yearOfBirth'] = 1969;

Jane['job'] = 'retired';

Jane['isMarried'] = 'yes';

console.log(Jane);

Object & Methods

Var John = {

name: 'John',

lastName: 'Smith',

YearOfBirth: 1990,

job: 'Teacher',

isMarried: false,

family: ['Jane', 'Mark', 'Bob'],

CalculateAge: function (YearOfBirth) {

return 2016 - YearOfBirth;

}

{

Console.log(John.family[2]);

Console.log(John.CalculateAge(1990));

v2.

```
var john = {  
    name : 'John',  
    lastname : 'Smith',  
    yearOfBirth : 1990,  
    job : 'teacher',  
    isMarried : false,  
    family : ['Jane', 'Mark', 'Bob']  
}
```

```
calculateAge: function() {  
    this.age = 2016 - this.yearOfBirth;  
}  
};
```

```
john.calculateAge();  
console.log(john);
```

Var name = ['John', 'Jane', 'Marry', 'Mark', 'Bob'];

~~for~~

for (var i=0; i<name.length; i++) {

 console.log(name[i]);

}

for (var i=name.length-1; i>0; i--)

 {
 console.log(name[i]);

}

var i=0;
while (i<name.length) {

while loop

 console.log(name[i])

 i++;

}

 {
 console.log(i);

} (i++); if (i < name.length)

 loop & iteration

```
for (var i=0; i<5, i++) {
```

```
    console.log(i);
```

① if (i==3) {
 break;

}

}

```
for (var i=0; i<5; i++) {
```

```
    if (i==3) {
```

```
        continue;
```

}

```
    console.log(i);
```

}

Coding Challenge - 2

- ① Create an array with some years where person were born
2. Create an empty array (just {})
3. Use a loop to fill the array with the ages of the persons
4. Use another loop to log into the console whether each person is of full age (18 or older), as well as their age
5. Finally, Create a function called printFullAge which receives the vector of years as an argument, executes the step 2., 3. and 4. and returns a vector of true/false boolean values:
true if the person is of full age (≥ 18 years) and false if not (< 18 years)
6. Call the function with two different vectors and store the results in two variables: full_1 and full_2

Solution for code challenge

```
var years = [1990, 2001, 1995, 1994, 2014];  
var ages = [ ];  
var fullAges = [ ];  
for (let i=0; i < years.length; i++) {
```

~~age[i] = 2016 - years[i];~~

```
}  
for (let i=0; i < ages.length; i++) {
```

```
    if (ages[i] >= 18) {  
        console.log('person is ' + ages[i] + ' years old, and  
        is of full age.);  
        fullAges.push(true);
```

```
    } else {  
        console.log('person is ' + ages[i] + ' years old, and  
        is not of full age.);  
        fullAges.push(false);
```

```
}  
}
```

```
function printFullAge(years) {
```

```
    return fullAges;
```

```
var full1 = printFullAge(years);  
var full2 = printFullAge([2012, 1915, 1999]);
```

Object and function

Everything is an object : Inheritance and the Prototype chain

Primitives

- numbers
- strings
- Booleans
- undefined
- null

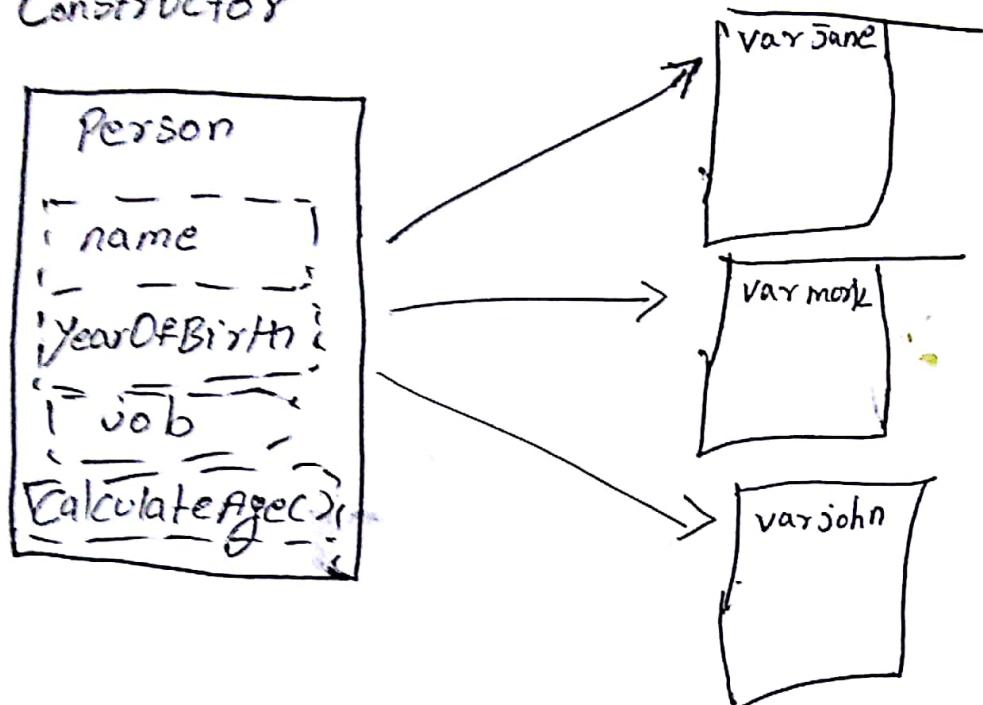
Everything else

- Array
- functions
- Objects
- Dates
- Wrappers for numbers, strings, Booleans
- .. is an object

Object oriented Programming

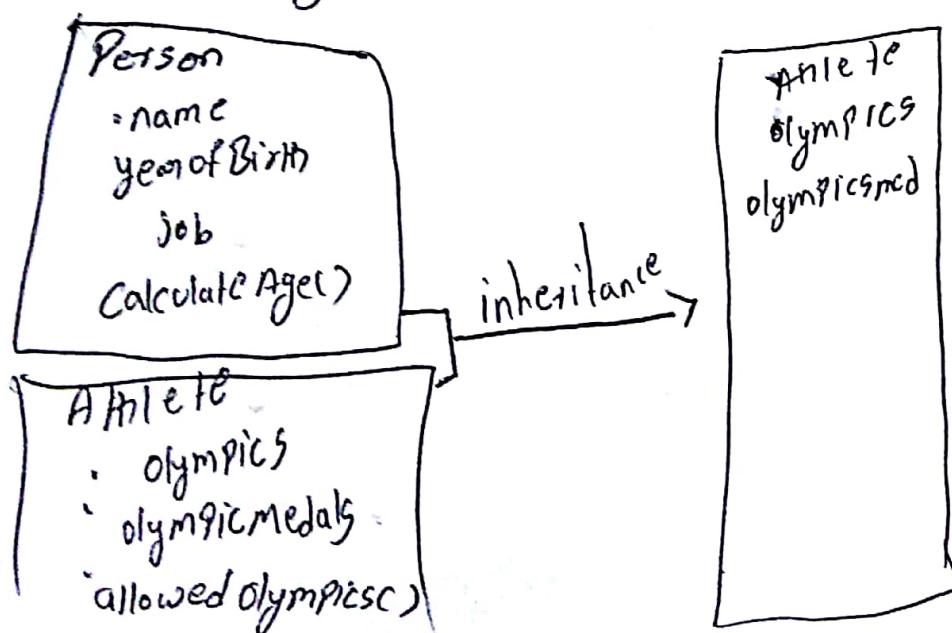
- Objects interacting with one another through methods and properties;
- Used to store data, structure applications into modules and keeping code clean.

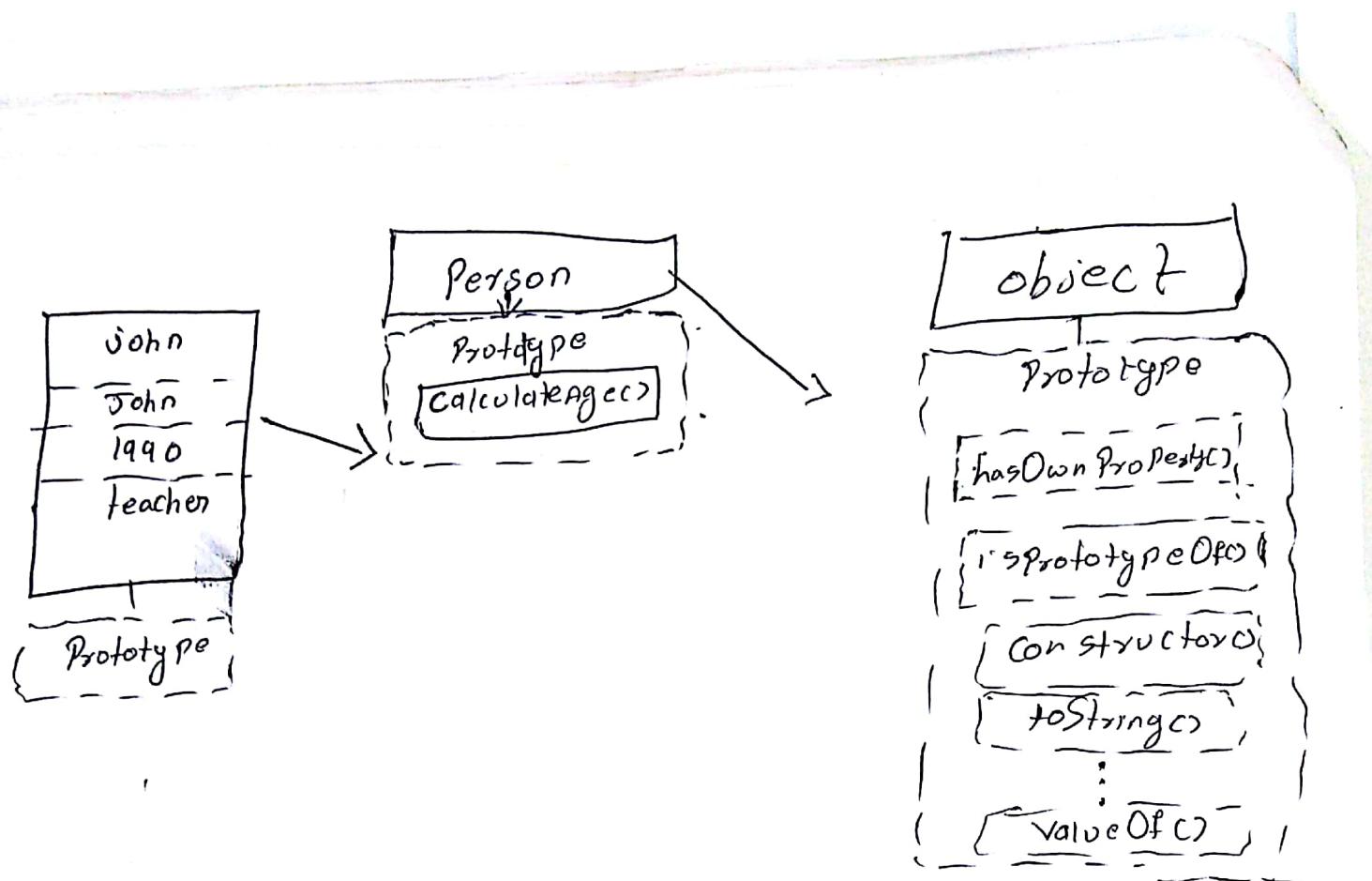
Constructor



inheritance.

It is nothing but one object based on the second one.





Summary

- Every JavaScript object has a `prototype` property, which is inheritance possible in JavaScript;
- The `prototype` property of an object is where we put methods and properties that we want other objects to inherit;
- The constructor's `prototype` is not the `prototype` of the constructor itself it's the `prototype` of all instances that are created through it;
- When a certain method (or property) is called, the search starts in the object itself, and if it cannot be found, the search moves on to the object's `prototype` this continues until the method is found;
prototype Chain

Creating object using function constructor

Var john {

 name: 'John',
 yearOfBirth: 1990;
 job = 'teacher'

} ;
object literal

Var person = function (name, yearb, job)

{

 this.name = name,
 this.yearOfBirth = yearOfBirth,
 this.job = job

} ;

Var john = New person ('John', 1990, 'teacher')

Var

 calculateAge = function () {

 console.log(2017 - this.yearOfBirth);

} ;

john.calculateAge();

person.prototype.lastName = 'Smith';

The Prototype chain in the console

-proto- : object

Creating objects: Objects.Create

var personProto = {

 calculateAge: function() {

 console.log(2017 - this.yearOfBirth);

}

var john = Object.create(personProto);

john.name = 'John';

john.yearOfBirth = 1990;

john.job = 'teacher';

var Jane = Object.create(personProto, {

 name: { value: 'Jane' },

 yearOfBirth: { value: 1969 },

 job: { value: 'designer' }

Primitives vs Objects

```
var a = 23;
```

```
var b = a;
```

```
a = 46;
```

```
console.log(a); // 46
```

```
console.log(b); // 23
```

```
var obj1 = {
```

```
name: 'John',
```

```
age: 26
```

```
};
```

```
var obj2 = obj1
```

```
obj1.age = 40
```

```
console.log(obj1.age); // 40
```

```
console.log(obj2.age); // 40
```

* Then storing an object in a variable, doesn't mean storing value, It means we pointing it to that particular function.

```
var age = 27;
```

```
var obj = {
```

```
name: 'Jonas',
```

```
city: 'Lisbon'
```

```
};
```

```
function Change(a, b) {
```

```
a = 30
```

```
b.city = 'San Francisco';
```

```
}
```

```
Change(age, obj);
```

```
}, console.log(age);  
}, console.log(obj.city);
```

Functions are also objects in JavaScript

Passing function as a argument

- A function is an instance of the object type;
- A function behaves like any other object;
- We can store functions in a variable;
- We can pass a function as an argument to another function;
- We can return a function from a function.

First-Class Functions

```
var years = [1990, 1965, 1937, 2005, 1998];
function arrayCalc(arr, fn) {
    var arrRes = [];
    for (var i = 0; i < arr.length; i++) {
        arrRes.push(fn(arr[i]));
    }
    return arrRes;
}

function calculateAge(c1) {
    return 2017 - c1;
}

var Result = arrayCalc(years, calculateAge);
console.log(Result);
```

```
function isFullAge(c1) {
    return 2017 - c1;
    if (c1 >= 18) {
        var fullAges = arrayCalc(ages,
            isFullAge);
        console.log(fullAges);
        console.log(isFullAge);
    }
}

function maxHeartRate(c1) {
    if (c1 >= 1988 & c1 <= 81) {
        return math.round(206.9 - (0.67 * c1));
    } else {
        return -1;
    }
}

var HeartRate = arrayCalc(ages, maxHeartRate);
console.log(HeartRate);
```

Function returning function

```
function interviewQuestion(job) {
    if (job === 'designer') {
        return function(name) {
            console.log(`name + ', can you please explain what is UX
                        designer?'`);
        }
    } else if (job === 'teacher') {
        return function(name) {
            console.log(`what subject do you teach, ${name}?`);
        }
    } else {
        return function(name) {
            console.log(`Hello ${name}, what do you do?`);
        }
    }
}

var teacherQuestion = interviewQuestion('teacher');
teacherQuestion('John'); // what subject do you teach, John?

interviewQuestion('designer')(mark);
```

immediately invoke function expressions (IIFE);

```
function game() {  
    var score = Math.random() * 10;  
    console.log(score >= 5);  
}  
game();
```

```
C function () {  
    var score = Math.random() * 10;  
    console.log(score >= 5);  
}
```

```
C function (goodLuck) {  
    var score = Math.random() * 10;  
    console.log(score >= 5 - goodLuck);  
}  
()
```

Closures

```
function retirement(retirementAge) {
    var a = 'years left until retirement';
    return function(yearOfBirth) {
        var age = 2016 - yearOfBirth;
    }
}
```

```
console.log((retirementAge - age) + a);
}
```

```
var retirementUS = retirement(65);
retirementUS(1990); // 40 years left until retirement.
```

```
var retirementGermany = retirement(65);
var retirementIceland = retirement(69);
```

```
retirementGermany(1990);
retirementIceland(1990);
```

An inner function has always access to the variables and parameters of its outer function, even after the outer function has returned.

```
function interviewQuestion(job) {
    return function(name) {
        if (job === 'designer') {
            console.log(name + ' can you explain uxdesign?');
        } else if (job === 'teacher') {
            console.log('what subject do you teach ' + name + '?');
        } else {
            console.log('Hello ' + name + ' what do you do?');
        }
    }
}
```

```
interviewQuestion('designer')( 'marie');
```

Bind, Call, apply

```
Var john = {
```

```
    name: 'John',
```

```
    age: 26,
```

```
    job: 'teacher'
```

```
Presentation: function (style, timeOfDay) {
```

```
    if (style === 'formal') {
```

```
        console.log('Good' + timeOfDay); Ladies and gentlemen! I'm  
        this.name + ', I'm a' + this.job + 'and I'm ' + this.age + 'years old.'
```

```
    } else if (style === 'friendly') {
```

```
        console.log('Hey! what's up? I'm ' + this.name + ',
```

```
I'm a' + this.job + 'and I'm ' + this.age + 'years old.'
```

```
        Have a nice' + timeOfDay + '.');
```

```
Var emily = {
```

```
    name: 'Emily',
```

```
    age: 35,
```

```
    job: 'designer',
```

```
};
```

bind will return a function, so
we need to store it in a variable

```
john.Presentation('formal', 'morning');
```

```
john.Presentation.Call(emily, 'friendly', 'afternoon');
```

```
john.Presentation.apply(emily, [ 'friendly', 'afternoon' ]);
```

```
Var johnFriendly = john.Presentation.bind(john, 'friendly')
```

```
johnFriendly('morning');
```

loops

```
//ES5
forC var i=0; i < boxesArr5.length; i++) {
    if (boxesArr5[i].className == 'box blue') {
        continue; // skips the iteration & continues (break it stops like);
    }
    boxesArr5[i].textContent = "I Changed to blue!";
}
```

```
//ES6
for (const cur of boxesArr6) {
    if (boxesArr6.includes('blue')) {
        continue;
    }
    cur.textContent = "I Changed to blue!";
}

```

Array methods to find elements

```
//ES5 var ages = [12, 17, 8, 21, 14, 11];
```

map method

```
var full = ages.map(function (cur) {
    return cur + 18;
});
```

});

```
Console.log (full);
```

```
console.log (full.indexOf(true));
```

```
console.log (ages [full.indexOf(true)]);
```

// E96

```
console.log(ages.findIndex((cur => cur >= 18));
```

~~Output~~

```
Console.log(ages.findIndex((cur => cur >= 18));
```

The spread operator

To expand an elements of an array.

```
function addFourAges (a, b, c, d) {  
    return a+b+c+d;
```

3

```
var sum1 = addFourAges(18, 30, 12, 21);  
console.log(sum1);
```

```
// E95  
var ages = [18, 30, 12, 21];
```

```
var sum2 = addFourAges.apply(null, ages);
```

```
Console.log(sum2);
```

expanding

// E96

```
const sum3 = addFourAges(...ages);
```

```
Console.log(sum3 sum3);
```

ES5

- Fully supported in all modern browsers.
- Ready to be used today (2016)

ES6 / ES2015

- Partial support in modern browsers, not support in older browser;
- Can't use it in production today (2016)

ES2016 / ES2017

- Almost no support in modern browsers;
- Can't use it in production today (2016)

- Variable declarations with let and const
- Blocks and IIFEs
- Strings
- Arrow Functions
- Destructuring
- Array
- The spread operator
- Rest and Default Parameters
- Maps
- Classes and subclasses
- How to use ES2015 / ES6 today

Variable declarations with let and const

ES5

Var name5 = 'Jane Smith';

Var age5 = 23;

~~Var~~

name5 = 'Jane Miller';

Console.log(name5);

→ Function scope



ES6

Const name6 = 'Janessmith';

Let age6 = 23;

name6 = 'Jane miller';

↓ block scope

Console.log(name6);

error

if you want to change the variable value in future we use 'LET'

if you want to keep the variable value constant we use 'CONST'

Driver licence function

function DriverLicence(PassedTest) {

if (PassedTest) {

Var FirstName = 'John';

Var YearBirth = 1990;

console.log(FirstName + ', born in ' + YearofBirth + ' is now

officially allowed to drive a car!);

}

Driveslicence5 (true);

//ES6

function Driveslicence6 (PassTest) {

if (PassTest) {

let FirstName = 'John';

const YearOfBirth = 1990;

Console.log (FirstName + ', born in ' + YearOfBirth + ', is
now officially allowed to drive a car.');

If you write the console outside ^{if statement}, you will get an error
because let, const are block scope / not a function scope

Driveslicence6 (true);

But in some cases we need to use those variables outside the
block scope

Let & const you need to declare them inside the function
not inside the if statements

Also if you ~~defin~~ console log a variable before declaring, what output you will get?

" undefined "

But if you ~~do~~ do the in the ES6 with let or const it will through an error on your face.

let i = 23;

for (let i=0; i<5; i++) {

//
 i

 console.log(i);

2

3

3

4

 console.log(i);

23

Assigning value to i outside the for loop and inside for loop is completely different variables like inside function outside the ~~both are~~ function.

Blocks and IFES

- Block is not just restricted to for loop, if statement or functions
- we can just create a block just by writing curly braces

1

1) This is a block

Const a=1; let b=2; var c=3;

2

console.log(a+b); // Error we will get, because block scope

console.log(c); // 3 because it's a var

// End

Function C {

Var C=3;

3) C;

Console.log(C); // Error

Strings in ES6 / ES2015 (There is very big improvement in strings)

let firstName = 'john';

let lastName = 'smith';

const yearOfBirth = 1990;

writing

new feature

template literals

We use backtick to write our entire string also the variable inside ~~it~~.

function calcAge(year) {

return 2016 - year;

3.

//ES5

Console.log('This is ' + firstName + ' ' + lastName + '. He was born in ' + yearOfBirth + '. Today, he is ' + calcAge(yearOfBirth) + ' years old.');

? This is quite a work

//ES6

Console.log(`This is \${firstName} \${lastName}. He was born in \${yearOfBirth}. Today, he is \${calcAge(yearOfBirth)} years old;`);

Other methods

```
const n = `${firstname} ${lastname}`;
```

```
n.startsWith('J');  
console.log(n.startsWith('J'));
```

```
console.log(n.endsWith('a'));
```

```
console.log(n.includes(' '));
```

```
console.log(firstname.repeat(5));
```

Arrow functions

Const years = [1990, 1965, 1982, 1937];

// ES5

Var ages5 = years.map(function (el) {

return 2016 - el;

});

Console.log (ages5);

// ES6

Let ~~const~~ ages6 = years.map((el) => 2016 - el);
operator

Console.log (ages6);

ages6 = years.map((el, index) => `Age element \${index + 1} is: \${2016 - el}`);

Console.log (ages6);

`ages6 = years.map((el, index) => {`

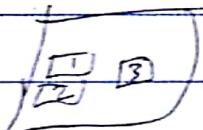
`const now = new Date().getFullYear();`

`const age = now - el;`

`return `age element ${index+1} : ${age}.``
});

`console.log(ages6);`

Arrow Functions: lexical 'This' Keyword



11 ESS

`var box5 = {`

`color: 'green',`

`Position: 1,`

`clickMe: function() {`

`document.querySelector('green').addEventListener('click',`
`function() {`

`var str = 'This is box number ' +`

`+ this.position + ' and ' +`
`this.color';`

`}, 3);`

`alert(str);`

De structuring

It is easy to extract data from an array or object and store it in a single variable

ES5

```
var john = ['John', 26];
```

```
var name = john[0];
```

```
var age = john[1];
```

ES6

```
const [name, age] = ['John', 26];
```

```
const obj = {
```

```
  firstName: 'John',
```

```
  lastName: 'Smith'
```

```
}
```

```
const {firstName, lastName} = obj;
```

```
console.log(firstName);
```

```
console.log(lastName);
```

```
Const {firstName: a, lastName: b} = obj;
```

```
Console.log(a);
```

```
Console.log(b);
```

```
function calcAgeRetirement(year) {
```

```
Const age = new Date().getFullYear() - year;  
return [age, 65 - age];
```

3

```
Const [age2, retirement] = calcAgeRetirement(1990);
```

```
Console.log(age2);
```

```
Console.log(retirement);
```

Arrays

```
Const boxes = document.querySelectorAll('.box');
```

//ES5

```
Var boxesArr5 = Array.prototype.slice.call(boxes);
```

```
boxesArr5.forEach(function (cur) {
```

```
    cur.style.backgroundColor = 'dodgerblue';
```

```
});
```

//ES6

```
Const boxesArr6 = Array.from(boxes);
```

```
boxesArr6.forEach((cur => cur.style.backgroundColor =  
    'dodgerblue'));
```

```
const FamilySmith = ['John', 'Jane', 'Marie'];
const FamilyMiller = ['Margo', 'Bob', 'Ann'];
const bigFamily = {...FamilySmith, ...FamilyMiller};
```

```
console.log(bigFamily);
```

```
const h = document.querySelectorAll('h1');
```

```
const boxes = document.querySelectorAll('.box');
```

```
const all = [h, ...boxes];
```

```
Array.from(all).forEach(cur => cur.style.color = 'purple');
```

Rest Parameters

```
// ES5
```

```
function isFullAge5() {
    console.log(arguments);
    var args = Array.prototype.slice.call(arguments);
    arrayArgs.forEach(function(cur) {
        console.log((2016 - cur) >= 15);
    });
}
```

```
isFullAge5(1990, 1999, 1965);
```

```
// ES6
```

```
function isFullAge6(...years) {
    console.log(years);
    years.forEach(cur => console.log((2017 - cur) >= 15));
}
isFullAge6(1990, 1999, 1965);
```

Default Parameters

1/ES5

```
function SmithPerson(firstname, yearOfBirth, lastname, nationality) {  
    this.firstname = firstname; → lastname === undefined ? lastname = 'smith';  
    this.yearOfBirth = yearOfBirth; → nationality === undefined ? nationality = 'american';  
    this.lastname = lastname; → nationality = nationality;  
    this.nationality = nationality;  
}
```

↳ var john = new SmithPerson('John', 1990);

↳ var emily = new SmithPerson('Emily', 1993, 'Diaz', 'spanish');

1/ES6

```
function SmithPerson(firstname, yearOfBirth, lastname='smith', nationality=  
    'american') {  
    this.firstname = firstname;  
    this.lastname = lastname;  
    this.yearOfBirth = yearOfBirth;  
    this.nationality = nationality;  
}
```

Maps [data structure]

```
Const question = new Map();
```

```
question.set('question', 'what is the official name of the  
latest major Java Script version?');
```

```
question.set(1, 'ES5');
```

```
question.set(2, 'ES6');
```

```
question.set(3, 'ES2015');
```

```
question.set(4, 'ES7');
```

```
question.set('correct', 3);
```

```
question.set(true, correctAnswer:D);
```

```
question.set(false, 'wrong, please try again');
```

In console

```
question → [enter]
```

```
console.log  
(question.get('question'));
```

```
console.log (question.size);
```

```
// if (question.has(4)) {
```

```
// question.delete(4);
```

```
// 3
```

```
// question.clear();
```

we can iterate over the map not as like objects

question.forEach((value, key) => console.log(`this is \${key}, and it's
set to \${value}`))

```
for (let [key, value] of question.entries()) {
    if (typeof(key) === 'number') {
        console.log(`Answer ${key}: ${value}`);
    }
}
```

```
const ans = prompt("write the correct answer")
console.log(
    question.get(ans === question.get('correct')))
```

Classes

1/ ESS

```
var Person5 = function(name, yearOfBirth, job) {
```

```
    this.name = name;
```

```
    this.yearOfBirth = yearOfBirth;
```

```
    this.job = job;
```

}

```
Person5.prototype.calculateAge = function() {
```

```
    var age = new Date().getFullYear() - this.yearOfBirth;
```

```
    console.log(age);
```

g

```
var john5 = new Person5('John', 1990, 'teacher');
```

1/ ESS → They are hosted

```
class Person6 {
```

```
    constructor(name, yearOfBirth, job) {
```

```
        this.name = name;
```

```
        this.yearOfBirth = yearOfBirth;
```

```
        this.job = job;
```

g

```
    calculateAge() {
```

```
        var age = new Date().getFullYear() - this.yearOfBirth;
```

```
        console.log(age);
```

g static greeting()

```
    static greeting() {
```

```
        console.log('Hey there!');
```

```
const john6 = new Person6('John', 1990, 'teacher');
```

```
Person6.greeting();
```