Жәңгір хан атындағы Батыс Қазақстан аграрлық-техникалық университеті

МАННАПОВА Т. М.
ҚАСЫМОВА А.Х.

JAVA SCRIPT-ОБЪЕКТІГЕ БАҒЫТТАЛҒАН СКРИПТІК БАҒДАРЛАМАЛАУ ТІЛІ

ОҚУ ҚҰРАЛЫ

Орал2017

УДК: 004.434
ББК: 32.973.26 –018.1
J 11

Жәңгір хан атындағы Батыс Қазақстан аграрлық-техникалық университет оқу-әдістемелік кеңесінің отырысында басылымға мақұлданды. (№ 5 хаттама 30.05.2017 ж.)

«Ақпараттық жүйелер» кафедрасының кафедра отырысында талқыланды. 20.03.2017ж., №8 хаттама.

Политехникалық факультетінің оқу әдістемелік кеңесімен ұсынылды. 24.03.2017ж., №8 хаттама.

Сын пікір беруші: **Кажиакпарова Ж.С.,** пед.ғыл.канд., доцент,
Құсайынов Р.Қ., физ-мат.ғыл.канд., доцент.

Маннапова Т. М., Қасымова А.Х.

J 11 Java script-объектіге бағытталған скриптік бағдарламалау тілі: оқу құралы/Т. М.Маннапова, А.Х.Қасымова, **-** Орал: Жәңгір хан атын. Батыс Қазақст. аграр.–техн. ун-ті, 2017.- 108 б.

ISBN 978-601-319-055-6

JavaScript тілі HTML/CSS тілдерінің жалғасы ретінде web парақтардың интерактивтілігін арттыру мақсатында оқытылады. Курста теориямен қоса көптеген практикалық сабақтар өтізіледі.

Оқу құралы«Web-технологиялары» пәнінің типтік бағдарламасы талаптарына сәйкес құрастырылған және курстың барлық тақырыптары бойынша зертханалық жұмыстарды орындау үшін қажет мәліметтерді қамтиды.

Оқу құралы кредиттік технология негізінде оқитын 5В070300 – «Ақпараттық жүйелер» мамандығы студенттеріне арналған.

УДК: 004.434

ББК: 32.973.26 –018.1

ISBN 978-601-319-055-6

МАЗМҰНЫ

- КІРІСПЕ ……...……...……4
- ТЕРМИНОЛОГИЯ…….....….....….....6
- JAVASCRIPT-ТА ДЕРЕКТЕР ТИПІ.....13
- ҚОРЫТЫНДЫ СҰРАҚТАР.....45
- JAVASCRIPT-та ҚАРАСТЫРЫЛАТЫН ТАҚЫРЫПТАР.....46
  - Тақырып 1. Javascript-тің негізгі ұғымдары.....46
  - Тақырып 2. Javascript-кодын қайда орналастыруға болады.....50
  - Тақырып 3. Бірінші бағдарлама, оқиғаларды өңдеу.....51
  - Тақырып 4. Javascript-функцияларды құру.....54
  - Тақырып 5. Функция параметрлері.....57
  - Тақырып 6. Javascript-та Math объектісі.....61
  - Тақырып 7. Тармақталу бағдарламасы - if операторы.....64
  - Тақырып 8. Switch таңдау операторы.....66
  - Тақырып 9. Web-парақтардың объектілерін басқару.....71
  - Тақырып 10. For және while циклдар.....75
  - Тақырып 11. Күндер, ұсынулар және өңдеулер.....78
  - Тақырып 12. Массивтер.....81
  - Тақырып 13. Қатарлар.....84
  - Тақырып 14. Тұрақты өрнектер.....87
  - Тақырып 15. Терезелермен жұмыс.....92
- 1-15 ТАҚЫРЫПТАРЫН ҚОЛДАНЫП JAVASCRIPT-та БАҒДАРЛАМАЛАУ.....97
- 1-15 ТАҚЫРЫПТАРЫН ҚОРЫТЫНДЫЛАУ СҰРАҚТАРЫ.....103
- ҚОЛДАНЫЛҒАН ӘДЕБИЕТТЕР.....106

КІРІСПЕ

JavaScript әдемі әсерлер (және) немесе әдемі және бай вебқосымшалар жасау үшін пайдаланылады,HTML мазмұнын және пішімдеуді сақтау үшін пайдаланылады, ал CSS вебпарақта мазмұнды шарттар мен талаптар қалай көрінетінін айқындайды.Қалай болмасын кең терминде JavaScript деп 2 түрлі мағына түсіндіріледі.Олардың бірі – бұл өз негізі – бағдарламалау тілі (ECMAScript), және басқасы бұл DOM (Document Object Model).

Бұл ECMAScript ұғымына не кіреді?

ECMAScript анықтайды:

- Синтаксис тілі (парсирлеу ережесі, түйін сөздер, орындау тәртібі...)
- Қателерді өңдеу механизмі (ерекшелік, try/catch операторы, қателердің жаңа типтерін беретін мүмкіндіктер)
- Деректер типі (логикалық, сандар, жолдар, функциялар, объектілері...)
- Ауқымды объект. Браузер, ауқымды объекті арасында -бұл объекті window. Бірнеше функциялар осы объектіге бекітілген (parseInt, parseFloat, decodeURI, encodeURI...)
- Мұрагерлік Механизмі прототипке негізделген
- Объектілері және функциялары құрылымдары (JSON, Math, Array.prototype әдістері, интроспекция объектінің әдістері...)
- Қатаң режимі

Браузерлік қолдау

Қазіргі таңда кеңінен таралған браузерлер толығымен ECMAScript 5 қолдайды.

DOM (Document Object Model)

WebIDL

DOM ортасы

DOM негізі W3C стандартталған. Стандарт программалық интерфейс арасындағы синтаксистік құрылымдар тілдеріHTML және XML құжаттарымен объектілер мен абстракцияның бақылау тетіктері ретінде анықтайды. Сонымен қатар, DOM анықтайды:

- Структуралық құжат ағаш және DOM сәулеті оқиғаларының DOM негізі түрінде: Node (Узел), Element (Элемент), DocumentFragment (құжат фрагменті), Document (құжат), DOMImplementation, Event (оқиға), EventTarget, …
- DOM оқиғалардың сәулетін анықтау, сондай-ақ белгілі бір DOM оқиғаларды анықтау неғұрлым қатаң.
- Басқа мәліметтер DOM Traversal және DOM Range.

Негізінен ECMAScript, объектілерді орындау ортасында анықталған объектілер ерекшелігінде DOM деп аталады ("host objects").

Объектілі-бағытталған JavaScript

Объектілі-бағытталған құрылуына дейін, JavaScript қуатты және икемді ООР мүмкіндігін береді.

Объектілі-бағытталған бағдарламалау

Объектілі-бағытталған бағдарламалау (ОББ) — бұл бағдарламалау парадигмасы, ол абстракция жасау үшін моделін негізделген объектілерінде, нақты әлемде пайдаланады. ОББ бұрын танылған парадигмалармен бірнеше техник пайдаланады, модульдік, полиморфизм және инкапсуляцияны қоса алғанда. Бүгінгі күні көптеген танымал бағдарламалау тілдері (мысалы, Java, JavaScript, C) ОББ-ны қолдайды.

ОББ бағдарламалық қамтамасыз ететін өзара іс-қимыл жасайтын объектілердің жиынтығы ретінде ұсынады, функцияларды немесе қарапайым командаларының тізімінің жиынтығы емес (дәстүрлі ұсыну). ОББ әрбір объекті хабарламалар алуы мүмкін, деректерді өңдеп және басқа да объектілерге хабарлар жібереді. Әрбір объект кішкентай тәуелсіз машинаның жеке рөлімен немесе жауапкершілігі сияқты ұсынылуы мүмкін.

ОББ бағдарламалауда икемділікті және қолдауды ықпал етеді, және кең таралған кең ауқымды бағдарламалық инжинирингке бөлінген.Өйткені ОББ модульдікті қатаң сынға алады, объектілі-бағытталған кодын әзірлеу оңай және кейіннен оңай түсіну үшін.Кодтау және күрделі жағдайларды түсіну мен рәсімдеріне қарағанда, объектілі-бағытталған коды неғұрлым дәл талдауды ықпал етеді.

ТЕРМИНОЛОГИЯ

Атаулар кеңістігі - Контейнер әзірлеушілерге, специфич үшін қосымшаның атымен байланыстыруға бүкіл функционалмен бірегей мүмкіндік береді.

Класс- Объект сипаттамаларын анықтайды. Класс шаблонның қасиеттері сипатталған және обьект әдістері болып табылады.

Объект- Класс экземплярі.

Қасиет- Объектінің сипаттамасы, мысалы, түсі.

Әдісі- Объектінің мүмкіндіктері. Бұл класспен байланысты функциялар немесе бағдарламалар.

Конструктор- Әдіс, объект данасын құру кезінде шақырылады. Оның класс секілді аты-жөні бар.

Мұрагерлік-Класс басқа класстың сипаттамаларын иеленуге мүмкін береді.

Инкапсуляция- Деректерде пайдаланылатын деректерді жинақтау мен әдістерінің тәсілі.

Абстракция- Кешенді мұрагерлік жиынтығы, объектінің әдістері және қасиеттері моделі шындығын көрсетуі тиіс

Полиморфизм- Поли "көп"-ті, ал морфизм "нысандарды" білдіреді. Әр түрлі класстар бір әдіс немесе қасиетті жариялауы мүмкін.

Объектілі-бағытталған бағдарламалау JavaScript атаулар кеңістігі

Атаулар кеңістігі — бұл контейнер, әзірлеушілерге функционалдық бірегей атаумен қосымшаларды жинау мүмкіндігі. Атаулар кеңістігі JavaScript — бұл объект, әдістер, қасиеттер және басқа да объектілерді қамтитын. Кеңістік есімдердің JS жұмыс істеу принципі қарапайым: осы объектінің қасиеті ретінде бір жаһандық объект және барлық айнымалылар, әдістері мен функцияларды жариялау керек. Сондай-ақ, кеңістіктер аттарын пайдалану қосымша атаулары қақтығыстар ықтималдығын азайтады, өйткені әрбір объект жаһандық объект қасиеті қосымшалары болып табылады.

Жаһандық объект MYAPP жасайық:

```
// Глобальное пространство имён
var MYAPP = MYAPP || {};
```

Барлық фрагмент кодында алдымен объект MYAPP айқындалды ма, осыны тексереміз (ағымдағы файлда немесе басқа файлда). Егер, тексерілген болса, онда қазіргі жаһандық объект MYAPP пайдаланамыз, әйтпесе бос объект MYAPP жасаймыз, онда барлық әдістер, функциялар, айнымалылар және объектілерді инкапсулирлейміз. Сондай-ақ, құру кеңестік аттарын құрамыз (ескеріңіз алдымен жаһандық объектті жариялау керек):

```
// Подпространство имён
MYAPP.event = {};
```

Бұдан әрі кеңістігі атауларын құру синтаксисі және айнымалылар, функциялар мен әдістерін қосу керек:

```
// Создаём контейнер MYAPP.commonMethod для общих методов и свойств
MYAPP.commonMethod = {
  regExForName: "", // определяет регулярное выражение для валидации имени
  regExForPhone: "", // определяет регулярное выражение для валидации телефона
  validateName: function(name){
    // Сделать что-то с name, вы можете получить доступ к переменной regExForName
    // используя "this.regExForName"
  },
  validatePhoneNo: function(phoneNo){
    // Сделать что-то с номером телефона
  }
}
// Объект вместе с объявлением методов
MYAPP.event = {
    addListener: function(el, type, fn) {
    // код
    },
    removeListener: function(el, type, fn) {
    // код
    },
    getEvent: function(e) {
    // код
    }
    // Можно добавить другие свойства и методы
}
// Синтаксис использования метода addListener:
MYAPP.event.addListener("yourel", "type", callback);
```

Стандартты жапсарлас объектілер

JavaScript-те жапсарлас ядрода бірқатар нысандар бар, мысалы, Math, Object, Array және String. Мысалы, төменде Math объектісін қалай пайдалануын көрсететеді, кездейсоқ санды random() әдісін пайдалана отырып алу үшін:

console.log(Math.random());

JavaScript объектілердің JavaScript Reference: Standard built-in objects барлық жапсарлас тізімімен танысу үшін қарау керек.

JavaScript-тің әрбір объектісі Object объектінің бір данасы болып табылады, оның барлық қасиеттері мен әдістерін иеленеді.

Класс

JavaScript — бұл прототипті-бағытталған тіл, онда class операторы жоқ.Кейде бағдарламашылармен түсінбейтін жағдайлар болады, өйткені тілдерде class операторымен жұмыстанып үйренген. Оның орнына JavaScript класс конструкторлар ретінде функцияларды пайдаланады. Мысал ретінде төменде жаңа класс Person бос конструкторын жариялаймыз:

var Person = function () {};

Объект (класс экземпляры)

obj объектісінің жаңа экземплярын құру үшін new obj операторын қолданамыз. Жоғарыдағы мысалда Person классты анықтадық. Төмендегі мысалда оның екі данасын (person1 және person2) құрамыз.

var person1 = new Person();
var person2 = new Person();

Конструктор

Конструктор класс данасын құру кезінде шақырылады (объект құрылған сәтте). Конструктор класс әдісі болып табылады. JavaScript функциясы объекті конструкторы ретінде қызмет етеді, сондықтан анық конструктор әдісін анықтау қажеттілігі жоқ. Кез келген әрекет белгілі бір класста, класс данасын құру кезінде болады.

Конструктор қолдануға дайындалатын объектіні дайындауда объектінің қасиеттерін құру үшін немесе әдістерді пайдалану үшінпайдаланылады. Төмендегі мысалда, конструктор Person класс консоль хабарламасы Person жаңа данасын құру кезінде шығарылады.

```
var Person = function () {
  console.log('instance created');
};
var person1 = new Person();
var person2 = new Person();
```

Қасиеті (объектінің аттрибуты)

Қасиеттері — бұл айнымалылар қамтылған класс, объектінің әрбір данасында осы қасиеттерді иеленеді. Конструктор (функциялары) класста қасиеттер белгіленеді, осылайша олар әрбір данасы үшін құрылады. Кілт сөз this, ол ағымдағы обьектіге сілтеме, класс қасиеттерімен жұмыс істеуге мүмкіндік

береді. Класс қасиеттеріне сыртынан қолжетімділігі (оқу және жазу) InstanceName синтаксисімен жүзеге асырылады. Төменде мысал ретінде firstName қасиетіг Person класс үшін экземпляр данасын құру кезінде анықтаймыз:

```
var Person = function (firstName) {
  this.firstName = firstName;
  console.log('Person instantiated');
};
var person1 = new Person('Alice');
var person2 = new Person('Bob');
// Выводит свойство firstName в консоль
console.log('person1 is ' + person1.firstName); // выведет "person1 is Alice"
console.log('person2 is ' + person2.firstName); // выведет "person2 is Bob"
```

Әдістер

Әдістер — бұл функциялар (және функция ретінде анықталады), екінші жағынан, сол логика, және қасиеттері жүреді. Шақыру әдісі қасиетке қол жеткізуге ұқсайды, бірақ сіз әдіс атын соңында () қосасыз, мүмкін дәлелдерді.

Әдістерді жариялау үшін, функцияны аталған қасиетімен prototype класстың қасиеттерімен иелененсіз. Содан кейін сіз объекті әдісін сіз мұралаған функцияларын есімімен шақырасыз. Төмендегі мысалда Person класс үшін sayHello()әдісін анықтаймыз және пайдаланамыз.

```
var Person = function (firstName) {
  this.firstName = firstName;
};
Person.prototype.sayHello = function() {
  console.log("Hello, I'm " + this.firstName);
};
var person1 = new Person("Alice");
var person2 = new Person("Bob");
// вызываем метод sayHello() класса Person
person1.sayHello(); // выведет "Hello, I'm Alice"
person2.sayHello(); // выведет "Hello, I'm Bob"
```

JavaScript әдістері бұл —функциялардың объектіге байланысты қасиеті ретінде жай объектілері: бұл дегеніміз"мәннен тыс" әдістерді тудырасыз. Келесі мысалды қарастырайық:

```
var Person = function (firstName) {
  this.firstName = firstName;
};
Person.prototype.sayHello = function() {
  console.log("Hello, I'm " + this.firstName);
};
```

```
var person1 = new Person("Alice");
var person2 = new Person("Bob");
var helloFunction = person1.sayHello;
// выведет "Hello, I'm Alice"
person1.sayHello();
// выведет "Hello, I'm Bob"
person2.sayHello();
// выведет "Hello, I'm undefined" (or fails
// with a TypeError in strict mode)
helloFunction();
// выведет true
console.log(helloFunction === person1.sayHello);
// выведет true
console.log(helloFunction === Person.prototype.sayHello);
// выведет "Hello, I'm Alice"
helloFunction.call(person1);
```

Мұрагерлік

Мұрагерлік — бұл класс ретінде мамандандырылған нұсқасында бір немесе бірнеше класс құру тәсілі (JavaScript жалғыз мұрагерлікті ғана қолдайды). Мамандандырылған класс, әдетте, ұрпағы деп аталады, ал екіншісі-класс ата-анасы.Төмендегі мысалда Student класс ретінде Person класс мұрагерін анықтаймыз. Одан кейін sayHello()әдісін қосамыз addDoodBye() әдісін қайтадан анықтаймыз.

```
// Определяем конструктор Person
var Person = function(firstName) {
  this.firstName = firstName;
};
// Добавляем пару методов в Person.prototype
Person.prototype.walk = function(){
  console.log("I am walking!");
};
Person.prototype.sayHello = function(){
  console.log("Hello, I'm " + this.firstName);
};
// Определяем конструктор Student
function Student(firstName, subject) {
  // Вызываем конструктор родителя, убедившись (используя
Function#call)
  // что "this" в момент вызова установлен корректно
  Person.call(this, firstName);
  // Инициируем свойства класса Student
  this.subject = subject;
```

```
};
// Создаём объект Student.prototype, который наследуется от Person.prototype.
// Примечание: Рспространённая ошибка здесь, это использование "new Person()", чтобы создать
// Student.prototype. Это неверно по нескольким причинам, не в последнюю очередь
// потому, что нам нечего передать в Person в качестве аргумента "firstName"
// Правильное место для вызова Person показано выше, где мы вызываем
// его в конструкторе Student.
Student.prototype = Object.create(Person.prototype); // Смотрите примечание выше
// Устанавливаем свойство "constructor" для ссылки на класс Student
Student.prototype.constructor = Student;
// Заменяем метод "sayHello"
Student.prototype.sayHello = function(){
  console.log("Hello, I'm " + this.firstName + ". I'm studying "
            + this.subject + ".");
};
// Добавляем метод "sayGoodBye"
Student.prototype.sayGoodBye = function(){
  console.log("Goodbye!");
};
// Пример использования:
var student1 = new Student("Janet", "Applied Physics");
student1.sayHello();   // "Hello, I'm Janet. I'm studying Applied Physics."
student1.walk();       // "I am walking!"
student1.sayGoodBye(); // "Goodbye!"
// Проверяем, что instanceof работает корректно
console.log(student1 instanceof Person);  // true
console.log(student1 instanceof Student); // true
```

Student.prototype = Object.create(Person.prototype) қатарына қатысты мыналар: Ескі JavaScript-та Object.create жоқ болса, онда полифиллді қолдануға (ретінде белгілі "shim") немесе сол нәтижелерге жеткізетін функциялар сияқты, мыналарды пайдалануға болады:

```
function createObject(proto) {
    function ctor() { }
    ctor.prototype = proto;
    return new ctor();
}
// Пример использования:
Student.prototype = createObject(Person.prototype);
```

Инкапсуляция

Жоғарғы мысалда Student класы үшін walk() әдісін іске асыру туралы Person класс білу қажет емес, бірақ ол оны пайдалана алады; Student класс бұл әдісті анық анықтауы тиіс емес, әзірге оны өзгерте алмаймыз. Бұл инкапсуляция деп аталады, соның арқасында әр класс деректерді және әдістері бір блокта жинайды.

Ақпаратты жасырудың таралған ерекшелігі, басқа да бағдарламалау тілдері ретінде жиі жүзеге асырылып жатқан жабық және қорғалған әдістері/қасиеттері. Алайда, JavaScript тек имитирлейтін нәрсе сияқты, бұл объектілі-бағдарланған бағдарламалауға қажетті талап болып табылады.

Абстракция

Абстракция – бұл механизм, мұрагерлік (мамандандыру) немесе композиция көмегімен жұмыс мәселелерінің ағымды фрагментін моделирлеуге мүмкіндік беретін. JavaScript мамандандыру мұрагерлігіне қолжеткізеді, класс данасы композициясы басқа обьектілердің атрибуттар мәндерімен мүмкіндігі болады.

JavaScript Fucntion класс Object класстан (бұл мамандануын көрсетеді) мұраға қалдырылады, ал Function.prototype қасиеті бұл Object класс данасы (бұл композиция көрсетеді).

```
var foo = function () {};
// выведет "foo is a Function: true"
console.log('foo is a Function: ' + (foo instanceof Function));
// выведет "foo.prototype is an Object: true"
console.log('foo.prototype is an Object: ' + (foo.prototype instanceof Object));
```

Полиморфизм

Барлық әдістері мен қасиеттері prototype қасиеттері ішінде анықталады, әртүрлі класстар әдістерді бірдей атауларымен анықтайды;көріну класстары әдістері олар анықталған облыста тұр, әзірге байланыспайтын екі класс ата-мұра бар (мысалы, біреуі мұрагерлікті басқа тізбегіндегілерге мұраға қалдырылады).

Ескертпелер

Бұл, объектілі-бағытталған бағдарламалау JavaScript олардың жүзеге асыруға болатын барлық әдістері емес. Сондай-ақ, бұл қаралған тәсілдері JavaScript–тің барлық мүмкіндіктерін көрсетпейді және басқа тілдерде объектілер теориясын іске асырмайды.

JAVASCRIPT-та ДЕРЕКТЕР ТИПІ

JavaScript нашар типизирленген немесе динамикалық тілі болып табылады. Бұл сізге айнымалы түрін алдын-ала анықтау қажет емес. Түр автоматты түрде орындау уақытында анықталады. Сондай-ақ, бұл дегеніміз, сіз бір айнымалыны әр түрлі типтегі деректерді сақтау үшін қолдана аласыз:

var foo = 42; // сейчас foo типа Number
var foo = "bar"; // а теперь foo типа String
var foo = true; // foo становится типа Boolean

Кез-келген тіл негіздерімен бастайық: мәліметтер типі. JavaScript–та бағдарлама мәндерімен, және барлық осы мәндер белгілі бір түрге тиесілі. ECMAScript стандарты 7 типтегі деректерді айқындайды:

•_6 тип деректер примитивтер болып табылады:
o_Boolean (Булев, Логикалық тип)
o_Null (Null типі)
o_Undefined (Белгісіз түрі)
o_Number (Саны)
o_String (Жол)
o_Symbol (ECMAScript 6)
•_ Object (Объект)

Undefined және Null,объектілерінің ерекше түрі болып табылатын массивтер.

Сондай-ақ күні мен өрнектерде объектілері болып табылады. Және, егер техникалық жағынан дәлме-дәл мүмкін болса, бұл да функциялардың ерекше түрі:

- **Сан**
- **Қатар**
- **Логикалық тип**
- **Объекті**
- **Символдар**
  - **Функция**
  - **Массив**
  - **Күн**
  - **Тұрақты білдіру**
  - **Null**
- **Undefined**

Сан

JavaScript–те сан — бұл "64-битті маңызы бар екілік форматты дәлдік IEEE 754" техникалық ерекшелікке сәйкес. JavaScript ешқандай бүтін сан ретінде берілмейді, сондықтан арифметикалық болуы керек, егер

сіз C немесе Java тілдерінде есептеулерге үйреніп кетсеңіз. Мына мысалға қараңыз:

0.1 + 0.2 == 0.30000000000000004

Практикада бұл 32-биттік бүтін маңызы бар (сақталады осылайша, кейбір браузерных реализациях), ол биттік операциялар үшін маңызды болуы мүмкін.

Стандартты арифметикалық операторлар қолдауды қоса алғанда, қосу, азайту, қалдық бөлу және т.б.. Кірістірілген Math деп аталатын обьект бар, оның құрамында неғұрлым озық математикалық функциялар мен тұрақтылар бар:

Math.sin(3.5);

*var circumference = Math.PI * (r + r);*

Сіз parseInt() функциясын пайдалана отырып бүтін санды түрлендіру жолын ала аласыз. Оның міндетті емес екінші параметрі — негізі, санау жүйесі, ол әрқашан осылай ұсынылуы қажет:

parseInt("123", 10); // 123

parseInt("010", 10); // 10

Егер сізде негіз болмаса, онда сіз күтпеген нәтижелерді алуыңыз мүмкін:

parseInt("010"); // 8

parseInt("0x10"); // 16

Бұл parseInt() функция 0 – бастауышына қарай сегіздікпен, ал оналтылық - "0x" бастауышпенқатарды бағалайды.

Егер түрлендіруді екілік санау жүйесінде көргіңіз келсе, онда жай ғана негізді ауыстырыңыз:

parseInt("11", 2); // 3

Сіз parseFloat() функцияны пайдалана отырып, ұқсас бөлшектік санын бересіз, әрдайым 10 негізімен parseInt() функциясы айырмашылығымен өзгеше.

Сондай-ақ +унарлы операторын пайдалануға болады, түрлендіру үшін маңызы бар:

+ "42"; // 42

+ "0x10"; // 16

Арнайы мәні NaN ("Not a Number" қысқартылған) қайтарылады, егер қатар сан болып табылмаса:

parseInt("hello", 10); // NaN

NaN "залалды": кез-келген математикалық операция үстінде NaN NaN-ды қайтарады:

NaN + 5; // NaN

isNaN()кіріктірме функциясымен NaN мәнін тексеруге болады:

isNaN(NaN); // true

JavaScript сондай-ақ, арнайы маңызы бар Infinity (шексіздік)-Infinity бар:

1 / 0; // Infinity
-1 / 0; // -Infinity

Infinity тексеру мәні, -Infinity және NaN болады, және isFinite ()функция көмегімен кірістірілген:

isFinite(1/0); // false
isFinite(-Infinity); // false
isFinite(NaN); // false

Қатар

JavaScript - бұл Unicode символдар реті (кодтау UTF-16), олардың әрқайсысы 16-битті санымен ұсынылған. Жалғыз символ ұсыну үшін бірлік жолағының ұзындығын пайдаланыңыз. Қатардың ұзындығын анықтау үшін length қасиетін пайдаланыңыз:

"hello".length; // 5

Бұл объектілермен жұмыста бірінші қадам болады!

"hello".charAt(0); // h
"hello, world".replace("hello", "goodbye"); // goodbye, world
"hello".toUpperCase(); // HELLO

Басқа да түрлері

JavaScript қосымша типтерді ажыратады, null секілді, ол әдейі undefined мәні көрсететін инициализирленбеген мәнін көрсетеді, ол тіпті тағайындалмаған болуы мүмкін.

Тағы JavaScript- логикалық (булевый) деректер түрі бар, ол екі мүмкін болатын true немесе false мәндерді қабылдай алады (екеуі де кілттік сөздер болып табылады). Кез келген логикалық мән, келесі ережелерге сәйкес қайта құрылуы мүмкін:

1. false, 0, бос
2. Қатар (""), NaN, null және undefined, false болып өрнектеледі
3. Қалғандары true болып өрнектеледі.

Мәндерді түрлендіруді Boolean() функциясын пайдалана отырып, анық жүзеге асыруға болады:

Boolean(""); // false
Boolean(234); // true

Бұл әдіс сирек қолданылады, өйткені JavaScript-та автоматты түрде түрлендіру типтері, булевтік мәні жағдайында күтілуі мүмкін, мысалы, оператор if.

Кез келген деректер түрі булевтік мәні болып қайта құрылуы мүмкін, кейде, бұл "шынайы" немесе "жалған" деп айтады. Логикалық деректер операциялары үшін логикалық операторлар пайдаланылады:&&(логикалық ЖӘНЕ), || (логикалық НЕМЕСЕ), !(логикалық ТЕРІСТЕУ).

Ауыспалы

Жаңа айнымалыларды жариялау үшін JavaScript пайдаланылады, кілт сөз var:

var a;

var name = "simon";

Егер сіз айнымалыны жарияласаңыз, оның қандай да бір маңызы болса, онда оның түрі undefined ретінде айқындалатын болады.

Операторлар

JavaScript мынадай операторларды қолдайды,+, -, *, /,%, ол қалдық бөлуді қайтарады (модулімен шатастырмау керек). Маңызы барлар + = оператордың көмегімен, немесе += және -= құрамдас операторлардың көмегімен беріледі. Бұл қысқартылған жазу x = x оператор *y* білдіру.

x += 5

x = x + 5

Сонымен қатар инкремента (++)және декремента (--) операторлары пайдаланылады. Олар префикс және постфикс үлгісін жазуды қолданады.

Оператор **+** сонымен бірге қатарлардың конкатенациясын (бірлестігі) орындайды:

"hello" + " world"; // "hello world"

Қосу кезінде жолдық және сандық мәндерді автоматты түрде қайта құру жолына көшіріледі.

Бастапқыда шатасуы мүмкін:

"3" + 4 + 5; // "345"

3 + 4 + "5"; // "75"

Маңызы бар жолда келтіру үшін, тек оған бос қатарды қосу керек.

Салыстыру операторлар

Салыстыру үшін JavaScript мына операторларды пайдаланады: <, >, <= және >=. Санды ғана емес, жолдарды да салыстыруға болады. Тексеру теңдігі сәл қиынырақ. Тексеру үшін қос (==) пайдаланады немесе үш (===)меншіктеу операторы қолданылады. Қос оператор == автоматты түрде типтерді түрлендіруді жүзеге асырады, бұл қызықты нәтижелерге әкелуі мүмкін:

123 == "123"; // true

1 == true; // true

Егер қайта құру орынсыз болса, онда қатаң теңдік операторы пайдаланады:

1 === true; // false

123 === "123"; // false

true === true; // true

Тексеру үшін!= және !== теңсіздік операторлары пайдаланады.

JavaScript үш оператордың салыстыру шамаларын ұсынады:

• теңдік ("қостық") қолданады,==,

• қатаң теңдік (немесе "үштік қостық" немесе "бірдей") пайдаланады ===,

• Object.is (жаңалық-дан ECMAScript 6).

Таңдау операторы салыстыру түріне байланысты.

Жалпы алғанда, қостық бірдей салыстыру алдында шамаларды келтіру типтері жүргізіледі; үштік қостық шамаларды келтірусіз салыстырады (егер шамалары әр түрлі типтегі болса, онда false болады, тіпті салыстырмай);ал, Object.is өзін де, үштік қостық сияқты, бірақ арнайы өңдеу үшін NaN, -0 және +0, -0 және +0 салыстыру кезінде false, және Object.is (NaN, NaN) операция үшін true болады.

Айта кету керек, барлық осы салыстыру айырмашылықтары қарапайымдар үшін ғана қолданылады. Кез келген примитивті емес x және y объектілері, бірдей құрылымды болатын, бірақ екі бөлек объектіні білдіретін (x және y айнымалылар бір және сол обьектіге сілтелінбейді), барлық салыстыру операторлары false болады.

Салыстыру == пайдалана отырып құру

Салыстыру алдында теңдік операторы екі шаманы жалпы түрге айналдырады. Айналдырудан кейін (бір немесе екі операнд), түпкі салыстыру орындалады, сондай-ақ ===. Салыстыру операциясы симметрлі: A == B нақ сол мәнді қайтарады, және B == A үшін A және B кез-келген мәндері болады.

Төмендегі кестеде салыстыру операторының теңдік үшін әр түрлі мәндер нәтижелері келтірілген (1-кесте):

1-кесте

		Операнд B					
		Undefined	**Null**	**Number**	**String**	**Boolean**	**Object**
Операнд A	**Undefined**	true	true	false	false	false	false
	Null	true	true	false	false	false	false
	Number	false	false	A === B	A === ToNumber (B)	A === ToNumber (B)	A == ToPrimitive (B)
	String	false	false	ToNumber (A) === B	A === B	ToNumber (A) === ToNumber (B)	A == ToPrimitive (B)
	Boolean	false	false	ToNumber (A) === B	ToNumber (A) === ToNumber (B)	A === B	ToNumber (A) == ToPrimitive (B)
	Object	false	false	ToPrimitive (A) == B	ToPrimitive (A) == B	ToPrimitive (A) == ToNumber (B)	A === B

Берілген кестеде, ToNumber(A) салыстыру алдында өз аргументін санға келтіруге тырысады. Мұндай жағдай +A (унарлы оператор +) баламалы. Егер ToPrimitive(A) аргумент ретінде объект алынса, онда оны примитивтіге аудару A талпыныстары. toString және A. valueOf әдістерімен жүргізіледі.

Дәстүрлі (және ECMAScript сәйкес), бірде-бір обьект undefined немесе null тең емес. Бірақ көптеген браузерлер белгілі бір класс объектілері (атап айтқанда, document.all объектілерге кез келген парақ) undefined мәнін эмулирлеуге мүмкіндік береді.

Теңдік операторы null == A және undefined == A үшін true мәні қайтарады, сонда және тек сонда ғана, обьект undefined мәнін эмулирлегенде. Қалған барлық жағдайларда обьект undefined немесе null –ге тең болуы мүмкін емес.

```
var num = 0;
var obj = new String("0");
var str = "0";
var b = false;
console.log(num == num); // true
console.log(obj == obj); // true
console.log(str == str); // true
console.log(num == obj); // true
console.log(num == str); // true
console.log(obj == str); // true
console.log(null == undefined); // true
// оба false, кроме очень редких случаев
console.log(obj == null);
console.log(obj == undefined);
```

Кейбір қолданушылар әрқашан қатаң теңдік салыстыру операторын салыстыру типтерімен бірге қолдануды жақсы деп санайды. Қатаң теңдік нәтижесін болжау оңай, тіпті салыстыру мағынасында, олардың сәйкес келтіруінсіз ұтыс жылдамдығын алуға болады.

Қатаң теңдік === пайдалана отырып

Қатаң теңдік екі шаманың теңдігін тексереді, мұнда әрбір шаманың түрі салыстыру алдында өзгермейді. Егер мәні ажыратылатын түрлерді қабылдаса, онда олар тең болмайды. Екінші жағынан, барлық сандық айнымалы, тиесілі бір түр болып саналады, егер құрамында бірдей шамалар бір-бірімен тең болса.

Сандық айнымалылар тең болып есептеледі, егер олар бірдей мағыналы болады, біреуі 0, ал екіншісі -0болса. Егер ең болмағанда сандық айнымалының біреуінің құрамында мәні NaN болса, өрнек false-ны қайтарады.

```
var num = 0;
```

```
var obj = new String("0");
var str = "0";
var b = false;
console.log(num === num); // true
console.log(obj === obj); // true
console.log(str === str); // true
console.log(num === obj); // false
console.log(num === str); // false
console.log(obj === str); // false
console.log(null === undefined); // false
console.log(obj === null); // false
console.log(obj === undefined); // false
```

Іс-жүзінде әрқашан салыстыру үшін қатаң теңдік операторын пайдалану керек. Барлық мәндерін үшін, сандықты қоспағанда айқын семантика пайдаланылады: шамасы тек өз-өзіне тең. Жоғарыда айтылғандай сандық типтері үшін екі ерекше жағдайды бөлуге болады.

Біріншіден, салыстыру +0 және -0. Нөл үшін белгісі кейбір есептеулерді жеңілдету үшін құбылмалы үтірден бастап, алайда, математика тұрғысынан, +0 және -0 арасында айырмасы жоқ, сондықтан, қатаң теңдік оператор оларды тең деп санайды. Екіншіден, NaN шамаларды салыстыру. NaN (Not a number) белгілі бір шама мәні болып табылмайды, ол нақты белгілі бір математикалық тапсырмалар үшін қолданылмайды (мысалы ∞ -∞). NaN қатаң теңдік оператор үшін бірде-бір шама тең болмайды, оның ішінде өз-өзіне (жалғыз болған жағдайда (x!==x) true болады).

Бірдей шамалар теңдігі

Бірдей шамалар теңдігі барлық контексттер салыстырылатын шамаларда *функционалдық ұқсастықты* анықтайды. (Бұл тәсіл салыстыру принципіне негізделеді). Мысал, өзгертудің өзгертілмейтін (immutable) қасиеттері:

```
// Добавление неизменяемого свойства NEGATIVE_ZERO
(отрицательный ноль) в конструктор Number.
Object.defineProperty(Number, "NEGATIVE_ZERO",
{ value: -0, writable: false, configurable: false, enumerable: false });
function attemptMutation(v)
{
  Object.defineProperty(Number, "NEGATIVE_ZERO", { value: v });
}
```

Өзгермейтін қасиеттерді өзгерту кезінде, Object.defineProperty шақыруды алып тастау, егер жаңа қасиет ескі өзгерістермен тең болса, өзгерістер болмайды және ерекшелік алып тасталынбайды. Егер v құрамында -0 болса, өзгерістер болмайды, демек, код ерекшеліктерді

шығарылмай пысықтайды. Алайда, егер v құрамында +0 болса, Number.NEGATIVE_ZERO өз өзгермейтін шамасын жояды. Дәл салыстыру үшін Object.is әдіспен берілген, жаңа және ағымдағы өзгермейтін қасиеттерін салыстыруда бірдей шамалар пайдаланылады (2-кесте).

2-кесте

Салыстыру операторларының салыстырмалы кестесі				
x	**y**	==	===	**Object.is**
undefined	undefined	true	true	true
null	null	true	true	true
true	true	true	true	true
false	false	true	true	true
"foo"	"foo"	true	true	true
{ foo: "bar" }	x	true	true	true
0	0	true	true	true
+0	-0	true	true	false
0	false	true	false	false
""	false	true	false	false
""	0	true	false	false
"0"	0	true	false	false
"17"	17	true	false	false
[1,2]	"1,2"	true	false	false
new String("foo")	"foo"	true	false	false
null	undefined	true	false	false
null	false	false	false	false
undefined	false	false	false	false
{ foo: "bar" }	{ foo: "bar" }	false	false	false
new String("foo")	new String("foo")	false	false	false
0	null	false	false	false
0	NaN	false	false	false

Салыстыру операторларының салыстырмалы кестесі				
x	**y**	==	===	**Object.is**
"foo"	NaN	false	false	false
NaN	NaN	false	false	true

Object.is қашан пайдалану керек?

Object.is нөлдердің өңдеу ерекшелігі метабағдарламалауда пайдалы болады, қажет болғанда қарама-қарсы мәні қасиеті Object.defineProperty дескриптор арқылы берілсін. Егер сіздің міндеттеріңіз осыларды талап етпесе, онда === артықшылығын беріп Object.is –ті пайдаланудан қалыс қалған жақсы. Тіпті кодыда екі NaN салыстыру қажет болса, әдетте, әр түрлі белгілермен нөлдерді салыстыруға ықпал ететін келесі есептеулер үшін, isNaN қолданыстағы әдісін пайдалану оңай.

Операторлар мен әдістер мысалдары, -0 жән +0 арасындағы айырмашылықтарды айқын жасай алатын, бұл міндетті түрде сіздің кодыңызға әсер етеді:

- *(унарлы минус)*

Унарлы минусты қолдану нөлді -0 береді. Бірақ, кейде, бұл елеусіз білінбей. Мысалы:

*let stoppingForce = obj.mass * -obj.velocity*

Егер obj.velocity мәні 0 болса, онда нәтижесі -0 болады, соңғы нәтижесінде stoppingForce айнымалыға оң өзгерістер әсер етеді:

Math.atan2
Math.ceil
Math.pow
Math.round

Мүмкіндік бар, егер оны анық параметрлердің бірі жасамасада,бұл әдістер -0 қайтаруы мүмкін.Мысалы, егер Math.pow әдісімен -Infinity кез келген теріс дәрежесіне салу. Деректер әдістерімен танысыңыз.

Math.floor
Math.max
Math.min
Math.sin
Math.sqrt
Math.tan

Кейбір жағдайларда нәтижеге қол жеткізуге болады, -0 тең, егер жоғарыда аталған әдістеріне бірі параметрлерді берсе -0. Мысалы, Math.min(-0, 0) -0 қайтарады.

~
<<
>>

Бұл операторлар ToInt32 ішкі алгоритмін пайдаланады. Онда теріс нөлге орын жоқ, сондықтан мәні -0 ұқсас операциялар аман қалмайды. Яғни Object.is(~~(-0), -0), және Object.is(-0 > 2, -0) түпнұсқаны false қайтарып береді.

Алынынған мәліметтен, Object.is пайдалану анық,кейде мәселелі болуы мүмкін. Әрине, егер сіз -0 және +0 анық ажырата алатын болсаңыз, онда дәл осы әдіс керек.

Басқарушы құрылымдар

JavaScript –те басқарушы құрылымдар өте ұқсас, C тектес тілдеріне. Шартты операторлар if және else негізгі сөздермен болатын, көрсетілген тізбекті құрайтын:

```
var name = "kittens";
if (name == "puppies") {
  name += "!";
} else if (name == "kittens") {
  name += "!!";
} else {
  name = "!" + name;
}
name == "kittens!!"
```

JavaScript-үш типті цикл бар: while, do-while және for. While қарапайым цикл тапсырмалар үшін пайдаланылады, do-while орынды қолдануға, егер сіз циклдің бір рет қана орындалғанын қаласаңыз:

```
while (true) {
  // бесконечный цикл!
}
var input;
do {
  input = get_input();
} while (inputIsNotValid(input))
```

Цикл for C және Java тілдеріне ұқсас: ол деректерді беру үшін циклдің орындалуын бақылауға мүмкіндік береді:

```
for (var i = 0; i < 5; i++) {
  // Выполнится 5 раз
}
```

Логикалық операторлар && және || "қысқа циклді есептеулер" пайдаланады, бұл есептеуде әрбір келесі оператор алдыңғысына байланысты. Мысалы, объектіні тексеру пайдалы болама немесе жоқпа, алдымен оның қасиеттеріне қол жеткізуге тырысу керек:

```
var name = o && o.getName();
```

Осындай тәсілмен үнсіз беру ыңғайлы:

```
var name = otherName || "default";
```

Сондай-ақ JavaScript -та шартты операторларына "?" тернарлы операторы тиесілі:

```
var allowed = (age > 18) ? "yes" : "no";
```

Оператор switch бірнеше салыстыру қажет болған жағдайда пайдаланылады:

```
switch(action) {
  case 'draw':
    drawit();
    break;
  case 'eat':
    eatit();
    break;
  default:
    donothing();
}
```

Егер case инструкция соңында тоқтату break инструкциясын қоспаса, онда нұсқаулық келесі case инструкциясын орындауға көшеді. Әдетте мұндай жағдай орынсыз, бірақ егер сіз оны кенеттен пайдалансаңыз, табандылықпен түсініктеме жазуға тиісті қателерді жеңілдетуді іздеуді ұсынамыз:

```
switch(a) {
  case 1: // fallthrough
  case 2:
    eatit();
    break;
  default:
    donothing();
}
```

default опционалды нұсқа. Өрнектерде switch қалай болған жағдайда да пайдалануға жол беріледі, сондай-ақ cases-тіде. Тексеру кезінде қатаң теңдік === теңдік операторы пайдаланылады:

```
switch(1 + 3) {
  case 2 + 2:
    yay();
    break;
  default:
    neverhappens();
}
```

Объекті

JavaScript объектілер жұптық атау-мәні (кілт-мәні) коллекцияларын білдіреді. Олар ұқсас:

- Python словары.
- Perl және Ruby хештер.
- C және C++ хеш кестелері.
- Java-да HashMaps.
- PHP-да ассоциативті массив.

JavaScript объект қасиеттері атымен жол құрылады, ал мәні JavaScript кез келген деректер түрі болуы мүмкін, тіпті басқа да объектілер. Бұл кез-келген қиындықтағы деректер құрылымын жасауға мүмкіндік береді.

Объекті жасауға екі негізгі тәсілі бар:

var obj = new Object();

Сонымен қатар:

var obj = {};

Осы екі жазба бір нәрсені жасайды. Екінші жазба литерал объекті деп аталады және өте ыңғайлы. Мұндай тәсіл JSON форматы негізі болып табылады, және кодты жазу кезінде оны пайдалану ең жақсы шешім болып табылады.

Литерал объектінің көмегімен тек бос объектілерді құрмай, жәнеде деректермен объектілерді құруға болады:

```
var obj = {
  name: "Carrot",
  "for": "Max",
  details: {
    color: "orange",
    size: 12
  } }
```

Объектінің қасиетіне қол жеткізудің екі тәсілін алуға болады:

```
obj.name = "Simon";
var name = obj.name;
```

Немесе...

```
obj["name"] = "Simon";
var name = obj["name"];
```

Бұл екі әдіс сондай-ақ, салалас. Бірінші әдісті, егер қандай әдіс қажет екенін анық білгенде пайдаланылады. Екінші әдіс қатар қасиеттері атаулары ретінде қабылданады, және есептеу процесінде есептеу атын анықтауға мүмкіндік береді.

Айта кету керек, соңғы әдіс кейбір қатарларға және кодты минимизаторлы оңтайландыруға кедергі келтіреді. Егер қажеттігі болып объект қасиеттері аттары ретінде резервтелген сөздер тағайындалса, онда бұл әдісте пайдалы болуы мүмкін:

```
obj.for = "Simon";
// Вызовет Syntax error, ведь 'for' это зарезервированное слово
obj["for"] = "Simon"; // А тут всё номрально
obj.details.color; // orange
obj["details"]["size"]; // 12
```

Массивтер JavaScript объектілердің жеке жағдайы. Олар іс жүзінде бірдей жұмыс жасайды (егер қасиеттер атымен сан болып табылса, онда оған қол жеткізу тек [] жақша ішінде шақыру арқылы болады), тек бір

ғажайып қасиеті 'length' (ұзындығы) бар. Ол санды қайтарады, ең үлкен массив индексі +1 тең.

Ескіше массив құру:

```
var a = new Array();
a[0] = "dog";
a[1] = "cat";
a[2] = "hen";
a.length; // 3
```

Бірақ әлдеқайда литерал массив пайдалануға ыңғайлы:

```
var a = ["dog", "cat", "hen"];
a.length; // 3
```

Есте сақтаңыз, array.length қасиеті массивте элементтердің санын көрсету міндетті емес. Үлгісін көріңіз:

```
var a = ["dog", "cat", "hen"];
a[100] = "fox";
a.length; // 101
```

Есте сақтаңыз — массив ұзындығы бұл оның ең үлкен индексі плюс бір. Элементі жоқ массивке қол жеткізу үшін, онда undefined алыңыз:

```
typeof a[90]; // undefined
```

Массив элементтерін жинақтау үшін мынадай тәсілді пайдаланыңыз:

```
for (var i = 0; i < a.length; i++) {
   // Сделать что-нибудь с элементом a[i]
}
```

Бұл кодты жақсартуға болады, ол үшін length мәні әрбір итерацияда қайтадан мәнді сұрамау қажет:

```
for (var i = 0, len = a.length; i < len; i++) {
 // Сделать что-нибудь с элементом a[i]
}
```

Тағы бір тәсілі бар жазбалар, ол жаман көрінбейді, бірақ шын мәнінде бір жетіспеушілігі бар:

```
for (var i = 0, item; item = a[i++];) {
 // Сделать что-нибудь с элементом item
}
```

for циклдың бірінші бөлігінде екі айнымалыны жариялаймыз. Екінші бөлімінде әр итерациямен жаңа біліктілік жүргізіледі және бірден тексеріледі — ол дұрыс немесе жоқ. Егер дұрыс болса, онда цикл жұмысын жалғастырада. Бұл циклдың кемшілігі массивтің элементі жалған мән (мысалы, undefined) кездестірсе жұмысын тоқтатады.

Мынадай әдісті қолдануға болады, өте сенімді бола отырып, бұл массивтің бірде-бір элементтері жалған мәнді қайтармайтын болады (мысалы, массивтер, объектілер немесе коллекциялар тораптары DOM). Егер сіздің массив құрамында сандық немесе жолдық параметрлер болса,

жолдық мәліметтер, яғни ықтималдығы, бұл цикл қасиеттері нөл немесе бос жолға жолығады. Сондықтан жоғарыда көрсетілген жазбаның екінші жолын пайдаланған жақсы.

Массив элементтерін, сондай-ақ for...in цикл көмегімен жинақтауға болады. Бірақ, егер кенеттен Array.prototype қасиетінде қандай да бір өзгертілу болса, онда ол да таңдауға қатысатын болады. Бұл әдісті пайдаланбаңыз.

Ең жаңа тәсілі массивтің қасиеттері — бұл ECMAScript5–ке қосылған forEach() әдісі:

```
["dog", "cat", "hen"].forEach(function(currentValue, index, array) {
// Сделать что-нибудь с currentValue или array[index]
});
```

Массивке деректер қосу үшін, push() әдісін пайдаланыңыз:

```
a.push(item);
```

Массивтердің тағыда көптеген пайдалы әдістері бар (3-кесте).

3-кесте

Әдіс	**Анықтама**
a.toString()	Массивтің жолдық ұсынуын қайтарады, барлық элементтер үтірмен бөлінеді
a.toLocaleString()	Массивтің жолдық ұсынуын сәйкес таңдалған локальмен қайтарады
a.concat(item1[, item2[, ...[, itemN]]])	Аталған элементтер қосылған жаңа массивті қайтарады.
a.join(sep)	Массивті sep параметрді бөлгіш ретінде пайдаланылған жолға түрлендіреді
a.pop()	Массивтің соңғы элементін жояды және оны қайтарады.
a.push(item1, ..., itemN)	Массивтің соғына бір немесе одан да көп элементтерді қосу
a.reverse()	Массив элементтерінің тәртібін керіге өзгертеді.
a.shift()	Массивтің бірінші элементін жою және оны қайтару
a.slice(start[, end])	Жаңа массивті қайтарады
a.sort([cmpfn])	Массивте деректерді сұрыптайды
a.splice(start, delcount[, item1[, ...[, itemN]]])	Массивтен оның бір бөлігін қиып алу және бұл орынға жаңа элементтер қосуға мүмкіндік береді
a.unshift(item1[, item2[, ...[, itemN]]])	Массивтің басына элементтерді қосу

Функциялар

Функциялары, сондай-ақ JavaScript тілінің түйінді элементтері болып табылады. Негізгі функциялары өте қарапайым:

```
function add(x, y) {
   var total = x + y;
   return total;
}
```

Бұл мысалда функциялары туралы іс жүзінде білу керектердің барлығы көрсетілгендей. JavaScript функцияларда нөл немесе одан да көп параметрлерді қабылдай алады. Функциялар кез келгенді білдіруі және өз айнымалыларын анықтауды қамтуы мүмкін, олар осы функциялар үшін жергілікті. return нұсқауы маңызы бар мәнді қайтару үшін пайдаланылады және тоқтату функциясын орындайды. Егер return нұсқаулық функцияда жоқ (немесе бар, бірақ көрсетілген қайтарылатын мәні) болса, онда JavaScript undefined қайтарады.

Функцияны оған параметрлерін мүлдем бермей шақыртуға болады. Мұндай жағдайда, олардың мәндері undefined тең:

```
add(); // NaN
// Нельзя проводить сложение undefined и undefined
```

Көбірек аргументтер беруге болады функцияны күтуге қарағанда:

```
add(2, 3, 4); // 5
// используются только первые два аргумента, "4" игнорируется
```

Бұл қажетсіз болып көрінуі мүмкін, бірақ шын мәнінде функциялары "артық" аргументпен arguments псевдомассивті көмегімен қол жеткізе алады, онда маңызы бар функциялардың барлық аргументтері берілген. Шексіз аргументтер қабылдайтын функциялар жазайық:

```
function add() {
   var sum = 0;
   for (var i = 0, j = arguments.length; i < j; i++) {
      sum += arguments[i];
   }
   return sum;
}
add(2, 3, 4, 5); // 14
```

Немесе орташа мәндерді есептеу үшін функция құрамыз:

```
function avg() {
   var sum = 0;
   for (var i = 0, j = arguments.length; i < j; i++) {
      sum += arguments[i];
   }
   return sum / arguments.length;
}
avg(2, 3, 4, 5); // 3.5
```

Өте ыңғайлы, бірақ шағын проблема бар. Бұл функция үтірмен бөлінген цифрлардың тізімін ғана түсінеді. Ал егер массивке сандар беру керек болса? Функция құрайық:

```
function avgArray(arr) {
   var sum = 0;
   for (var i = 0, j = arr.length; i < j; i++) {
      sum += arr[i];
   }
   return sum / arr.length;
}
avgArray([2, 3, 4, 5]); // 3.5
```

Бұл жағдайда, егер сіз функциялардың бірінші нұсқасын пайдаланғыңыз келсе, оны қайтадан жазбай-ақ, онда JavaScript–та аргументтер массивімен функцияны шақыру мүмкіндігі бар. Бұл үшін apply() әдісі пайдаланылады:

```
avg.apply(null, [2, 3, 4, 5]); // 3.5
```

Екінші аргумент apply() әдісімен массив беріледі, функциялар аргументтер ретінде берілетін. Бірінші аргумент туралы кейінірек айтамыз. Функциялардың әдістерінің болуы, сондай-ақ, бұл-шын мәнінде, олар объектілері болып табылады.

JavaScript –та анонимді функциялар жасауға болады:

```
var avg = function() {
   var sum = 0;
   for (var i = 0, j = arguments.length; i < j; i++) {
      sum += arguments[i];
   }
   return sum / arguments.length;
}
```

Бұл жазба function avg() семантикалық тең жазу. Бұл әр түрлі қызықты фокустарды пайдалануға мүмкіндік береді. Міне, қараңызшы, жергілікті айнымалыларды "жасыру" функциясы қалай болады:

```
var a = 1;
var b = 2;
(function() {
   var b = 3;
   a += b;
})();
a; // 4
b; // 2
```

JavaScript -та рекурсивті функцияларды шақыру мүмкіндігі бар. Бұл иерархиалық құрылымдар деректерімен (мысалы, DOM жұмыс жасау кезінде кездесетін) жұмыс істеу кезінде пайдалы болуы мүмкін.

```
function countChars(elm) {
   if (elm.nodeType == 3) { // TEXT_NODE
      return elm.nodeValue.length;
   }
   var count = 0;
   for (var i = 0, child; child = elm.childNodes[i]; i++) {
      count += countChars(child);
   }
   return count;
}
```

Мұнда мәселелермен бетпе-бет келеміз: функцияны рекурсивті қалай шақыруға болады, егер онда атау жоқ болса? Бұл үшін JavaScript–та аталған функционалдық өрнектеу бар. Мысал, функцияларды пайдалану:

```
var charsInBody = (function counter(elm) {
   if (elm.nodeType == 3) { // TEXT_NODE
      return elm.nodeValue.length;
   }
   var count = 0;
   for (var i = 0, child; child = elm.childNodes[i]; i++) {
      count += counter(child);
   }
   return count;
})(document.body);
```

Мысалда функциялардың аты ғана функциялардың ішіндегі ең қолжетімділік. Бұл кодты оңтайландырады және оқылуды жақсартады.

Меншікті объектілер

Классикалық объектно-ориентирленген бағдарламалау (ООБ) объектілері — бұл деректер мен әдістердің коллекциясы, осы деректермен жүзеге асатын. JavaScript-бұл тіл, типтік негізделген, және класс сияқты ұғымдар C немесе Java тілдерінде анықтауы жоқ.

Класс орнына JavaScript-та функциялар пайдаланады. Объектінің жеке деректерін қамтитын өрістің аты және тегі. Аттарды белгілеудің екі түрі бар: "Аты Тегі" немесе "Тегі, Аты". Объектілер мен функциялардың көмегімен мыналарды жасауға болады:

```
function makePerson(first, last) {
   return {
      first: first,
      last: last
   }  }
function personFullName(person) {
   return person.first + ' ' + person.last;
}
```

```
function personFullNameReversed(person) {
    return person.last + ', ' + person.first
}
s = makePerson("Simon", "Willison");
personFullName(s); // Simon Willison
personFullNameReversed(s); // Willison, Simon
```

Жұмыс істейді, бірақ бұл код жарамсыз. Мұндай тәсілмен жаһандық объектпен ондаған функциялар болады. Бұларды объектіге функцияны тіркеп түзетуге болады. Бұл барлық функциялары олар объектілер:

```
function makePerson(first, last) {
    return {
        first: first,
        last: last,
        fullName: function() {
            return this.first + ' ' + this.last;
        },
        fullNameReversed: function() {
            return this.last + ', ' + this.first;
        }   }   }
s = makePerson("Simon", "Willison")
s.fullName(); // Simon Willison
s.fullNameReversed(); // Willison, Simon
```

Ал мыналар жаңалар: 'this' кілт сөзі. Егер 'this' функциялар ішінде пайдаланылса, ол ағымдағы объектте сілтеме жасайды. Кілт сөздер мәні функцияларды шақыру әдістеріне байланысты болады. Егер функцияны шақыру, объектіге үндеу жолдау, нүкте арқылы немесе тік төртбұрышты жақшалар болса, онда 'this' осы объектіге тең болады. Өзге жағдайда 'this' жаһандық объектіге сілтеме болады. Бұл жиі қателіктерге әкеледі. Мысалы:

```
s = makePerson("Simon", "Willison")
var fullName = s.fullName;
fullName(); // undefined undefined
```

fullName() шақыру кезінде, 'this' жаһандық объектіге сілтеме алады. Ал жаһандық объектіде айнымалылар first және last анықталған жоқ, онда екі undefined иеленеміз.

'this' кілт сөздер ерекшелігін пайдаланып makePerson функция кодын жақсартуға болады:

```
function Person(first, last) {
    this.first = first;
    this.last = last;
    this.fullName = function() {
        return this.first + ' ' + this.last;
```

```
    };
    this.fullNameReversed = function() {
        return this.last + ', ' + this.first;
    };
}
var s = new Person("Simon", "Willison");
```

Мысалда қолданған жаңа кілт сөз: 'new'. Ол 'this' тығыз байланысты. Бұл кілт сөз, жаңа бос объект жасайды, одан кейін аталған функцияны шақырады, athis бұл жаңа объектіге сілтеме алады. Функцияларды шақыруға арналған 'new' конструкторлар деп аталады. Келісім бар, оған сәйкес барлық функциялары-конструкторлар бас әріптен басталып жазылады.

Әр қашан конструктор көмегімен жаңа обьект құрылады, қайтадан құрамыз және екі жаңа функцияларды жасаймыз. Бұл функцияларды әлдеқайда бөлек-бөлек құру және конструкторға қол жеткізу қызметтері ыңғайлы:

```
function personFullName() {
    return this.first + ' ' + this.last;
}
function personFullNameReversed() {
    return this.last + ', ' + this.first;
}
function Person(first, last) {
    this.first = first;
    this.last = last;
    this.fullName = personFullName;
    this.fullNameReversed = personFullNameReversed;
}
```

Қазірдің өзінде код құрылымы жақсы: функциялар-әдістерін тек бір рет құрдық, ал жаңа шақыру арқылы функциялар-конструкторды жай сілтемелейміз. Одан да жақсы жасауға болады? Әрине:

```
function Person(first, last) {
    this.first = first;
    this.last = last;
}
Person.prototype.fullName = function fullName() {
    return this.first + ' ' + this.last;
}
Person.prototype.fullNameReversed = function fullNameReversed() {
    return this.last + ', ' + this.first;
}
```

Person.prototype бұл объект, оған кіру үшін барлық Person класс даналарының мүмкіндігі бар. Ол ерекше прототиптердің тізбегін

жасайды. Әр кезде, Person объектінің қолданыста жоқ қасиеттеріне қол жеткізуге тырысасыз, онда JavaScript Person.prototype қасиетінің бар жоғын тексереді. Нәтижесінде, Person.prototype объектінің барлық қасиеттеріне және this арқылы осы класстың барлық экземплярларына қол жетімді болады.

Бұл өте қуатты құрал. JavaScript прототипі кез келген уақытта өзгертуге мүмкіндік береді, бұл қолданыстағы объектілерді орындау кезінде жаңа әдістерді қосуға болатынын көрсетеді:

```
s = new Person("Simon", "Willison");
s.firstNameCaps();
// TypeError on line 1: s.firstNameCaps is not a function
Person.prototype.firstNameCaps = function() {
    return this.first.toUpperCase()
}
s.firstNameCaps(); // "SIMON"
```

JavaScript –тің жапсарлас салынған объектілеріне қызықты нәрсені қосудың прототиптер қасиеті болады. to String жаңа әдіс қосайық, ол мына қатарды соңынан алдына қайтарады:

```
var s = "Simon";
s.reversed(); // TypeError on line 1: s.reversed is not a function
String.prototype.reversed = function reversed() {
    var r = "";
    for (var i = this.length - 1; i >= 0; i--) {
        r += this[i];
    }
    return r;
}
s.reversed(); // "nomiS"
```

Осы әдіс тіпті литералды жолдан жұмыс істейтін болады!

```
"This can now be reversed".reversed();
// desrever eb won nac sihT
```

Айтылғандай, prototype бұл тізбек бөлігі. Осы тізбектің прототиптерінің түпкі объектісі Object.prototype әдістер болып табылады, әдістер toString()–діде қамтиды - бұл әдіс объектінің қатарлық бейнеленуін алу керек болғанда шақырылатын болады. Міне, енді Person объектілерімен нелер жасауға болады:

```
var s = new Person("Simon", "Willison");
s.toString(); // [object Object]
Person.prototype.toString = function() {
    return '<Person: ' + this.fullName() + '>';
}
s.toString(); // "<Person: Simon Willison>"
```

avg.apply()-ті бірінші аргументі null тең шақырдық. Енді былай жасаймыз: apply() әдісімен берілген объекттің бірінші аргументі, 'this' мәнін қабылдайтын. Міне, мысалы 'new' оңайлатып іске асыру:

```
function trivialNew(constructor, ...args) {
   var o = {}; // Создаём новый объект
   constructor.apply(o, arguments);
   return o;
}
```

Бұл new-тің дәл көшірмесі емес, өйткені ол прототиптердің тізбегін белгілемейді. Apply() әдісі өте жиі қолданылмайды, бірақ оны білу маңызды. Жоғарыгы мысалда, ...args (көпнүктеніқоса алғанда) жазбасы "қалған аргументтер" деп аталады — ол қалған барлық аргументтерді қамтиды.

JavaScript –те apply() әдісіне ұқсас call()әдісі бар, ол да 'this' белгілеуге мүмкіндік береді, бірақ массив аргументтерін емес, тізімді қабылдайды.

```
function lastNameCaps() {
   return this.last.toUpperCase();
}
var s = new Person("Simon", "Willison");
lastNameCaps.call(s);
// Аналогично записи:
s.lastNameCaps = lastNameCaps;
s.lastNameCaps();
```

Салынған функциялар

Жаңа функцияларды және басқа да ішкі функциялар жариялауға болады. Бұл қабылдауды жоғарыдан қолдана отырып, makePerson() функциясын жасағанбыз. Салынған функциялардың басты ерекшелігі олардың жарияланған басты -функцияларында айнымалыға қол жеткізе алады:

```
function betterExampleNeeded() {
   var a = 1;
   function oneMoreThanA() {
      return a + 1;
   }
   return oneMoreThanA();
}
```

Егер сіздің функцияңыз өзінің жұмысында басқа еш жерде қолданылмайтын функцияларды пайдаланса, онда қосалқы функцияларды негізгіге салуға болады. Бұл жаһандық объектіде функциялар санын қысқартады, бұл өте жақсы.

Жаһандық айнымалы санын қысқартудың тағы бір тамаша тәсілі. Осылай код жазғанда жаһандық айнымалылар жиі туындайтын болады, олар әр түрлі функцияларға қол жетімді болады. Бұл барлық кодты қиындатады, оны оқылатын етіп құрайды. Салынған функциялар өзінің ауыспалы басты функцияларына қол жеткізе алады, және оларды топтастыру үшін көптеген функцияларды бірге пайдалана аламыз (әрине, ақылға қонымды шекте), жаһандық объектті тазалықта және тәртіпте сақтауға мүмкіндік береді.

Тұйықталу

JavaScript құралдарының ең бір қуатты және түсініксізіне келдік. Талдап көрейікші.

```
function makeAdder(a) {
   return function(b) {
      return a + b;
   };
}
var x = makeAdder(5);
var y = makeAdder(20);
x(6); // ?
y(7); // ?
```

makeAdder функциясы жаңа функция жасайды, ол функцияларды құру кезінде алынған мәнді мәнге қосады.

Осындайды алдыңғы мысалда көрдік, ішкі функциялар басқа функцияларға ауыспалы алып кіруші болып жарияланған. Тек мысалда, негізгі функциясы салынғанды қайтарады. Алғашқыда, бұл локальды айнымалылар жойылып кетуі мүмкін көрінуі мүмкін. Бірақ олар жойылмайды — әйтпесе код мүлде орындалмауы мүмкін еді. Оған қоса барлығына " makeAdder функцияларының екі түрлі "көшірмелерінің әр түрлі айнымалылары берілген (бір көшірмесі 5 қосады, екіншісі 20). Міне, нәтижесінде:

```
x(6); // возвратит 11
y(7); // возвратит 27
```

Міне, не болды: JavaScript функциясын орындағанда 'scope' объектісі құрылады, ол осы функциялардың ішінде жарияланған барлық жергілікті айнымалыларды қамтиды. Ол кез келген мәнмен инициализациялайды, берілген функциялардың параметрі ретінде. 'Scope' жаһандық объектіге ұқсас, ол құрамында барлық жаһандық айнымалылар және функцияларды ұстайды, бірнеше маңызды ерекшеліктерінен басқа: әрбір функцияларды шақыру кезінде жаңа 'scope' объектісі құрылады, жаһандық объектісіне айырмашылығы сіздің кодыңыздан 'scope' объектісіне тікелей қол жеткізу мүмкін болмайды. Және берілген объект қасиеттері бойынша өтіп кету тәсілі жоқ.

Сондықтан makeAdder функциялары шақыру кезінде жаңа 'scope' объектісі жалғыз қасиетпен құрылады: оған мәні беріледі, берілген функциялар ретінде сапасын қайта қарастыруды сұрайды. Содан кейін makeAdder жаңа анонимді функцияны қайтарады. Кез келген басқа жағдайда 'құрастырушы қоқыс' scope объектісін жойуы мүмкін, бірақ қайтарылатын функция бұл обьектіге сілтемеленеді. Нәтижесінде scope объектісі оған ең болмағанда бір сілтемеде жоқ болғанша жойылмайды.

Барлық scope объектілері көріну облыстарының тізбегіне қосылады, ол JavaScript'а объектілік жүйесіне ұқсас тізбекті прототиптер.

Тұйықталу

Лексикалық көру облысы. Келесі мысалды қарастырайық:

```
function init() {
   var name = "Mozilla"; // name - локальная переменная, созданная в
init
   function displayName() { // displayName() - внутренняя функция,
замыкание
       alert (name); // displayName() использует переменную,
объявленную в родительской функции
   }
   displayName();
}
init();
```

init() функциясы name айнымалы локальын жасайды, содан кейін displayName() фукнциясы туындайды. displayName() — бұл ішкі функция, ол иinit() ішінде және тек осы функциялар денелерінің ішінде анықталған. init(), displayName() –ның локальды айнымалылар айырмашылығы жоқ және оның орнына name өзгерістері, белгілі бір басты функциясында пайдаланады.

Бұл кодты орындаңыз және ол қалай жұмыс істейтінін қараңыз. Бұл лексикалық көріну облысы (lexical scoping) деп аталатын үлгі: JavaScript-айнымалының қолданылу саласы, оның кодта орналасуы (бұл анық лексически) бойынша анықталады, және салынған функциялар сырттай жарияланған айнымалыларға қол жеткізе алады. Бұл механизм Lexical scoping (қолданылу саласы, шектелген лексика) деп аталады.

Келесі мысалды қарастырайық:

```
function makeFunc() {
  var name = "Mozilla";
  function displayName() {
    alert(name);
  }
  return displayName;
};
```

var myFunc = makeFunc();
myFunc();

Егер орындасақ, оның алдыңғы мысалдағы init() әсеріндей болады: "Mozilla" қатары JavaScript alert диалогында көрсетілген. Айырмашылығы - displayName() ішкі функциясы орындалғанға дейін сырттан қайтарылады.

Қарапайым жағдайда, функцияда локальды айнымалылар оны орындау кезінде ғана бар. makeFunc() шақырғаннан кейін, енді бұл name айнымалысы қол жетімсіз болатынын күтуге болады. Бұл, әлбетте, тұйықталу жағдайында емес.

Бұл басқатырғыштардың шешімі, myFunc *тұйықталуға* айналуы. Тұйықталу — бұл объектінің ерекше түрі, онда екі нәрсені біріктіреді: функцияны және онда функция құрылған оның ортасын. Орталар жаңа іс-қимыл саласындағы функциялардың тұйықталуының құрылуындағы кез келген локальдық айнымалыдан тұрады. Бұл жағдайда, myFunc — бұл тұйықталу, ол displayName функциясын, және "Mozilla"қатарын, тұйықталууақытындаболған құруды қамтиды.

Мұнда қызықты мысал —makeAdder функциясы:

```
function makeAdder(x) {
  return function(y) {
    return x + y;
  };
};
var add5 = makeAdder(5);
var add10 = makeAdder(10);
console.log(add5(2));  // 7
console.log(add10(2)); // 12
```

Мұнда makeAdder(x) функциясын анықтадық, ол x таңдайды және жаңа функцияны қайтарады. Бұл функция y –ті алады, және x және y қосындысын қайтарады.

Бұл мәні бойынша, makeAdder фабрикалық функциясы — ол нақты мәндері үшін өз аргументтерін қосуы мүмкін функцияларды жасайды. Жоғарғы мысалда фабрикалық функцияны екі жаңа функция құру үшін қолданамыз – бірінші оның аргументі ретінде 5 мәнін анықтайды, екіншісі - 10.

add5 және add10 - тұйықталу мысалдары болып табылады. Осы функциялардың денесі бірдей, бірақ олар түрлі ортаны сақтайды. add5 x функциясын қоршалған – 5, ал бұл уақытта add10 x функциясы қоршалған – 10.

Тұйықталу тәжірибесі

Тұйықталу қаншалықты пайдалы? Олармен бірге не жасауға болады, қарайық. Жалпы, тұйықталу қандай да бір деректерді (нақты ортасы) осы деректермен жұмыс істейтін функцияны байланыстыруға мүмкіндік береді. Объектілі-бағытталған бағдарламалаумен параллельдік, онда объект деректер жиынтығын (объектінің қасиеттері) бір немесе бірнеше әдістермен байланыстырулар жүргізуге мүмкіндік береді.

Яғни, тұйықталуды барлық жерде пайдалануға болады, онда бір жалғыз тәсілмен объектіні қалыпты және дұрыс пайдалану болса. Мұндай жағдайлар web -әзірлеудің барлық жерінде кездеседі. Web үшін JavaScript кодының үлкен санын жазу оқиғалар қатары болып табылады - яғни, қандай да бір сипаттауды, содан кейін оны қандайда бір оқиғаға тіркейміз, ол пайдаланушы арқылы іске қосылады (мысалы, тышқанмен шертуді немесе пернелерді басқанда). Бұл ретте код әдетте кері шақыру ретінде тіркеледі (жауапты): осы оқиғаның басталуынажауапты, жүзеге асыратын жеке функция.

Практикалық мысал қарастырайық: параққа мәтін көлемін ауыстыратын бірнеше батырмалар қосу. Нұсқа ретінде, font-size қасиетін body элементі үшін пикселде көрсетеміз, содан кейін парақтар элементтерінің басқа размерін орнату (мысалы, тақырыптар, мысалы) үшін em салыстырмалы бірлігін пайдалану:

```
body {
 font-family: Helvetica, Arial, sans-serif;
 font-size: 12px;
}
h1 {
 font-size: 1.5em;
}
h2 {
 font-size: 1.2em;
}
```

Сонда батырмалар body элементінің font-size қасиетін өзгертетін болады, ал парақтың басқа қалған элементтері бұл жаңа мәнді тіркейді және салыстырмалы бірлікті қолданудың арқасында мәтін көлемін масштабтайды:

```
function makeSizer(size) {
  return function() {
    document.body.style.fontSize = size + 'px';
  };            };
var size12 = makeSizer(12);
var size14 = makeSizer(14);
var size16 = makeSizer(16);
```

Енді size12, size14 және size16 - бұл функциялар body элементінде мәтін мөлшерін, тиісінше 12, 14, 16 пикселдік мәніне өзгертеді. Содан кейін бұл функцияларды батырмаларға шамамен осылайша тіркейміз:

```
document.getElementById('size-12').onclick = size12;
document.getElementById('size-14').onclick = size14;
document.getElementById('size-16').onclick = size16;
<a href="#" id="size-12">12</a>
<a href="#" id="size-14">14</a>
<a href="#" id="size-16">16</a>
```

Тұйықталу көмегімен жекеше әдістерінің эмуляциясы

Java сияқты тілдері жекеше әдістерін жариялауға мүмкіндік береді. Бұл кезде тек сол класстағы басқа әдістер оларды тудыруы мүмкін, ал олар сыртында қол жетімді емес.

JavaScript мұндай тікелей жасау мүмкін болмайды, бірақ мұндай сипаттауды тұйықталу арқылы эмулирлеуге болады. Кодқа қол жеткізуді шектеу үшін, кейбір әдістердің жекешелеу мағынасын жасау керек емес. Бұл өзінің дербес жаһандық атаулар кеңістігінің жалпы кеңістіктігін бітемеу үшін құрылатын жақсы амал, яғни жалпы кеңістікте жекешелеу мүлдем қажет емес.

Міне, жекешеленетін әдістерге және айнымалыларға қол жеткізетін, тұйықталу көмегімен бірнеше көпшілік жария әдістерін сипаттау. Мұндай бағдарламалау мәнері module pattern деп аталады. "Модуль", "шаблон" және "Javacript"деген іздеу сөздерін пайдалануға болады:

```
var Counter = (function() {
  var privateCounter = 0;
  function changeBy(val) {
    privateCounter += val;
  }
  return {
    increment: function() {
      changeBy(1);
    },
    decrement: function() {
      changeBy(-1);
    },
    value: function() {
      return privateCounter;
    }
  };
})();
alert(Counter.value()); /* Alerts 0 */
Counter.increment();
```

```
Counter.increment();
alert(Counter.value()); /* Alerts 2 */
Counter.decrement();
alert(Counter.value()); /* Alerts 1 */
```

Мұнда көп нәрсе өзгерді. Алдыңғы мысалда әрбір тұйықталуда орындаудың (ортасы алған болса) өз контексті болды. Мұнда үш функция үшін біртұтас ортасын құрамыз: Counter.increment, Counter.decrement және Counter.value.

Жасырын функциялар құрамында бірыңғай орта құрылады, ол сипаттау сәтті орындалады. Бұл орта құрамында екі жекешелену элементі болады: privateCounter айнымалысы және changeBy фукциясы. Бірде-бір осы элементтердің бірі қол жетімді емес, тікелей ең анонимді функциялардан тыс болады. Оның орнына олар қолжетімді және үш жария функциялары пайдаланылуы тиіс, олар сол анонимді функцияларда орындалады, анонимді блок кодын қайтарады (anonymous wrapper).

Бұл жалпы контекстті орындауда (ортасы) пайдаланылатын,үш жария функциялары тұйықталу болып табылады. lexical scoping Javascript механизімі көмегімен, олар privateCounter айнымалысы мен changeBy функцияларына қол жеткізеді. Назар аударыңыз, есептегіш жасайтын анонимді фунцияны суреттейміз, және Counter айнымалының орындау нәтижесін тағайындап,оны бірден іске қосамыз. Бірақ бұл функцияны бірден іске қоспауымызға болады, оны жеке айнымалыны пайдалану үшін сақтап, одан әрі осылай бірнеше есептегіштерді құра аламыз:

```
var makeCounter = function() {
  var privateCounter = 0;
  function changeBy(val) {
    privateCounter += val;
  }
  return {
    increment: function() {
      changeBy(1);
    },
    decrement: function() {
      changeBy(-1);
    },
    value: function() {
      return privateCounter;
    }
  }
};
var Counter1 = makeCounter();
var Counter2 = makeCounter();
alert(Counter1.value()); /* Alerts 0 */
```

```
Counter1.increment();
Counter1.increment();
alert(Counter1.value()); /* Alerts 2 */
Counter1.decrement();
alert(Counter1.value()); /* Alerts 1 */
alert(Counter2.value()); /* Alerts 0 */
```

Назар аударыңыз, бұл есептегіштер бір бірінен тәуелсіз жұмыс істейді. Мұның себебі, makeCounter() функциясымен олардың әрқайсысын құру кезінде, өзінің жеке контекстін орындау (ортасы) құрылған. Бұл шын мәнінде жеке өзіндік айнымалы,яғни әрбір есептегіштердің жекеше privateCounter айнымалысы.

Тұйықталуды осылайша пайдаланып, әдетте объектілі-бағытталған бағдарламалаумен байланысты оқшаулау және инкапсуляция сияқты, сіз бірнеше артықшылықтарды аласыз.

Тұйықталуды құру циклі: Өте жиі кездесетін қате.

ECMAScript 6 нұсқасы let кілт сөзін енгізді. Келесі мысалды қарастырайық:

```
<p id="help">Helpful notes will appear here</p>
<p>E-mail: <input type="text" id="email" name="email"></p>
<p>Name: <input type="text" id="name" name="name"></p>
<p>Age: <input type="text" id="age" name="age"></p>
function showHelp(help) {
  document.getElementById('help').innerHTML = help;
}
function setupHelp() {
  var helpText = [
      {'id': 'email', 'help': 'Ваш адрес e-mail'},
      {'id': 'name', 'help': 'Ваше полное имя'},
      {'id': 'age', 'help': 'Ваш возраст (Вам должно быть больше 16)'}
    ];
  for (var i = 0; i < helpText.length; i++) {
    var item = helpText[i];
    document.getElementById(item.id).onfocus = function() {
      showHelp(item.help);
    }
  }
}
setupHelp();
```

Массив helpText үш кеңес үшін үш алаңды енгізуді сипаттайды. Цикл-бұл сипаттаулар кезек бойынша жүріп өтеді және әрбір енгізу өрісін анықтайды, бұл кезде onfocus оқиғалары үшін, бұл элементке керек функция туындайды, көрсететін тиісті ақпарат жазылып тұрады.

Егер сіз бұл кодты іске қоссаңыз, онда оның біз қалағандай жұмыс істемейтінін көруге болады. Сіз қандайда өрісті таңдасаңызда, кеңестер ретінде әрқашан құқы туралы хабарлама көрсетіледі.

Функциялар onfocus оқиғаларды өңдеуіштер ретінде берілген тұйықталу болып табылады. Олар setupHelp функциясынан мұраға қалдырылған функцияларды сипаттаудан және орындау (ортасынан) контексттен тұрады,.Үш тұйықталу құрылды, бірақ олардың барлығы бір контекстті орындаудан (ортада) құрылған. onfocus оқиғалар сәтінде туындаған цикл бұрыннан жұмыс істеді, ал item (үш тұйықталу) айнымалы массивтің соңғы элементін көрсетеді, ол өріске байланысты.

Бұл жағдайда шешім ретінде функцияларды, фабрикалық функцияларды (function factory), жоғарыда мысалда сипатталып айтылған пайдалануды ұсынуға болады:

```
function showHelp(help) {
  document.getElementById('help').innerHTML = help;
}
function makeHelpCallback(help) {
  return function() {
    showHelp(help);
  };
}
function setupHelp() {
  var helpText = [
      {'id': 'email', 'help': 'Ваш адрес e-mail'},
      {'id': 'name', 'help': 'Ваше полное имя'},
      {'id': 'age', 'help': 'Ваш возраст (Вам должно быть больше 16)'}
    ];
  for (var i = 0; i < helpText.length; i++) {
    var item = helpText[i];
    document.getElementById(item.id).onfocus =
makeHelpCallback(item.help);
  }
}
setupHelp();
```

Міне, бұл жөнді жұмыс істейді. Барлығына бір ортадан бөлудің орнына makeHelpCallback функциясы әрбір тұйықталуға өзінің жекелігін жасайды, онда item айнымалы helpText массивтің дұрыс элементін көрсетеді.

Пайымдаулар бойынша өнімділік

Функциялар ішіндегі қажеттілікте тұйықталу қажет емес жағдайларда функцияларды жасау қажеті жоқ. Бұл техниканы пайдалану,сондай-ақ жадты тұтыну бөлігінде қойылатын талаптардың жылдамдығы тұрғысынан өнімділік артады.

Мысалы, бұл объектінің прототипіне жаңа объект/класс құруда баяндау мәтінінде конструктор барлық әдістерді енгізу мәні бар. Себебі, мұнды басқаша болса, оларды прототиптан мұраға қалдыру орнына әрбір объектіні құру үшін оған әрбір әдістің данасы құрылады.

Парактикалық емес, бірақ көрсетілімді мысал қарастырайық:

```
function MyObject(name, message) {
  this.name = name.toString();
  this.message = message.toString();
  this.getName = function() {
    return this.name;
  };
  this.getMessage = function() {
    return this.message;
  };
}
```

Жоғарыда көрсетілген код тұйықталудың артықшылықтарында пайдаланылмайды, оны төмендегідей қайта жазуға болады:

```
function MyObject(name, message) {
  this.name = name.toString();
  this.message = message.toString();
}
MyObject.prototype = {
  getName: function() {
    return this.name;
  },
  getMessage: function() {
    return this.message;
  }
};
```

Прототипке әдістер ұсынылады. Дегенмен, прототипті қайтааныктау — қажет емес, сондықтан жаңа әдістерді қолданыстағы прототипке қосуға болатындай, барлығын қайта жазайық:

```
function MyObject(name, message) {
  this.name = name.toString();
  this.message = message.toString();
}
MyObject.prototype.getName = function() {
  return this.name;
};
MyObject.prototype.getMessage = function() {
  return this.message;
};
```

Жадының жылыстауы

Тұйықталуда бір жағымсыз жанама әсер бар — олармен өте қарапайым Internet Explorer–та жадының жылыстауын алуға болады. JavaScript-та жадны босату "қоқыс жинаушының" көмегі арқылы жүреді — әрбір объектіні құру кезінде физикалық жад көлемі бөлінеді және бұл жад браузермен объектіге бірде-бір сілтеме қалмаған кезде босатылады. Объектілер орындау ортасында құрылғандар, осы ортамен өңделеді.

Браузердің HTML парақтарды тудыратын көптеген үлкен объектілермен ісімен бар - DOM объектілері ретінде. Браузер міндеті жүктелген жадты бөлу және босату.

Осы үшін Internet Explorer браузер өз қоқыс құрастырушысын пайдаланады, ол JavaScript құрастырушы емес. Сәйкес келмейтін екі құрастырушылардың жұмысында жадтың жылыстауын туындатуы мүмкін.

Қашанда JavaScript объектісі арасындағы циклдық сілтеме және меншікті браузер объектісі пайда болған сайын,IE-да жадының жылыстауы пайда болады. Мысалы:

```
function leakMemory() {
   var el = document.getElementById('el');
   var o = { 'el': el };
   el.o = o;
}
```

Мысалда циклдік сілтеме жадының жылыстауына әкеледі. Браузер әзірге толық жүктелгенше, IE el және o-мен бөлінген жадыны босатпайды.

Жоғарыдағы мысалдың кодын, тіпті нақты жобада пайдаланылуға болады. Өйткені, жадының жылыстауы мәселесі қосымшалардың орындалуының үлкен мерзімімен және қосымшалардың жұмыс істейтін үлкен құрылымдар деректері немесе циклдарымен неғұрлым өзекті.

Әдетте, жылыстау соншалықты анық емес. Тұйықталу мүмкін жад жылыстауын байқаусызда тудыруы мүмкін:

```
function addHandler() {
   var el = document.getElementById('el');
   el.onclick = function() {
      this.style.backgroundColor = 'red';
   }
}
```

Бұл код клике кезінде элементтің түсін қызылға өзгертеді. Және жадының жылыстауын жасайды. Неге? el сілтемелері, байқамай жасырын ішкі функциямен құрылған тұйықталуға түскен. Бұл JavaScript (функциясы) объектісі және браузер объектісі (el) арасында циклдік сілтемені жасайды.

Осы қатені айналып өтудің бірнеше жолдары бар. Ең қарапайым бұл el айнымалыны мүлдем пайдаланбау:

```
function addHandler(){
    document.getElementById('el').onclick = function(){
        this.style.backgroundColor = 'red';
    }
}
```

Мәселені шешу тәсілдерінің бірі, тұйықталудың басқа тұйықталуда болуы:

```
function addHandler() {
    var clickHandler = function() {
        this.style.backgroundColor = 'red';
    };
    (function() {
        var el = document.getElementById('el');
        el.onclick = clickHandler;
    })();
}
```

Ішкі функция бірден орындалады және clickHandler функциясының тұйықталуынан өз контексін жасырады.

ҚОРЫТЫНДЫ СҰРАҚТАР

1. Бұл ECMAScript ұғымына не кіреді?
2. DOM нені анықтайды?
3. Объектілі-бағытталған JavaScript дегеніміз не?
4. Терминология дегеніміз не?
5. Объектілі-бағытталған бағдарламалауда JavaScript атаулар кеңістігі қандай?
6. Стандартты жапсарлас объектілерге нелер кіреді?
7. JavaScript класс конструкторлары ретінде қандай функцияларды пайдаланады?
8. Объект (класс экземпляры) дегеніміз не?
9. Конструктор дегеніміз не?
10. Объектінің аттрибуты, қасиеті қандай?
11. Класспен байланысты функциялар немесе бағдарламаларда қандай әдістер қолданылады?
12. Мұрагерлік дегеніміз не?
13. Инкапсуляция дегеніміз не?
14. Абстракция дегеніміз не?
15. Полиморфизм дегеніміз не?
16. ECMAScript стандарты неше және қандай типтегі деректерді айқындайды?
17. JavaScript–те сандар қандай?
18. Қатардың ұзындығын анықтау үшін қандай қасиет пайдаланамыз?
19. JavaScript - логикалық (булевый) деректер түрі қандай?
20. JavaScript қандай операторларды қолдайды?
21. Салыстыру операторларының кестелері қандай?
22. Бірдей шамалар теңдігі қандай ұқсастықты анықтайды?
23. Object.is қашан пайдалану керек?
24. JavaScript –те басқарушы құрылымдары қандай?
25. JavaScript-та неше типті цикл бар?
26. Массив helpText кеңес үшін қанша алаңды енгізуді сипаттайды?
27. Объектінің қасиетіне қол жеткізудің неше тәсілін алуға болады және қандай?
28. JavaScript тілінің түйінді элементтері нелер болып табылады және анықтама беріңіз?
29. Классикалық объектно-ориентирленген бағдарламалау (ООБ) объектілері не?
30. Салынған функциялардың басты ерекшелігі не?
31. Тұйықталуда қанша және қандай нәрсені біріктіреді?
32. Тұйықталуды құру циклі
33. Жадының жылыстауы не?
34. Циклдік сілтемені қатені айналып өтудің қандай жолдары бар?
35. Тұйықталу көмегімен жекеше әдістерінің эмуляциясы қандай?

JAVASCRIPT-та ҚАРАСТЫРЫЛАТЫН ТАҚЫРЫПТАР

- Тақырып 1. Javascript-тің негізгі ұғымдары.
- Тақырып 2. Javascript-кодын қайда орналастыруға болады.
- Тақырып 3. Бірінші бағдарлама, оқиғаларды өңдеу.
- Тақырып 4. Javascript-функцияларды құру.
- Тақырып 5. Функция параметрлері.
- Тақырып 6. Javascript-та Math объектісі.
- Тақырып 7. Тармақталу бағдарламасы - if операторы.
- Тақырып 8. switch таңдау операторы.
- Тақырып 9. web-парақтардың объектілерін басқару.
- Тақырып 10. for және while циклдар.
- Тақырып 11. Күндер, ұсынулар және өңдеулер.
- Тақырып 12. Массивтер.
- Тақырып 13. Қатарлар.
- Тақырып 14. Тұрақты өрнектер.
- Тақырып 15. Терезелермен жұмыс.

ТАҚЫРЫП 1. JAVASCRIPT-ТІҢ НЕГІЗГІ ҰҒЫМДАРЫ

JavaScript бағдарламалау тілі Sun Microsystems ынтымақтастықта Netscape фирмасымен әзірленді және 1995 жылы қолданысқа енді. JavaScript html-құжаттарды интерактивті құруға арналған. Негізгі пайдалану саласындағы JavaScript:

- динамикалық парақтарды құру,парақтар, олардың мазмұны жүктегеннен кейін өзгеруі мүмкін.
- пайдаланушы нысандарын толтыру дұрыстығын тексеру.
- сценарийлер көмегімен "локальды" міндеттерді шешу.
- JavaScript-код көптеген Ajax-қосымшалардың негізі.

JavaScript клиент жағында орындалатын қосымшаны құруға мүмкіндік береді, яғни осы қосымшалар браузермен пайдаланушының компьютерінде орындалады.

Бағдарламалар (сценарилер) осы тілде браузер ішіне орнатылған интерпретатормен өңделеді. Өкінішке орай, барлық сценарийлер барлық браузерлерде дұрыс орындалмайды, сондықтан да өз javascript-бағдарламаңызды әр түрлі браузерлерде пайдаланып көріңіз.

JavaScript тілі тіркеулерді ескереді, яғни бас және әліпбидің бас әріптері әртүрлі символдар болып саналады. Сценарийлерді жазуға кіріспес бұрын, литералы, айнымалылар және өрнектер сияқты негізгі ұғымдармен танысу қажет.

Литерал

Литерал - бұл қарапайым деректер, олармен бағдарлама жұмыс істей алады.

- **Литерал бүтін типті** - бүтін сандарды ұсыну:
- ондық, мысалы: 15, 5, -174.
- оналтылық, мысалы: 0x25, 0xff. Оналтылық санын 0 - 9 цифрлары және a, b, c, d, e, f грек әріптері қамтиды. Олар символдармен 0x саны алдында жазылады.
- сегіздік, мысалы: 011, 0543. Сегіздік санын тек 0 – 7 цифрлары қамтиды.
- **Нақты литерал** - бөлшек нөмірлерді анықтау. Бөлшектің бүтін бөлігі нүктемен ажыратылады, мысалы: 99.15, -32.45. Экспоненциалды жазу үлгісінің мүмкіндігі, мысалы: 2.73 e^{-7}. Қарапайым түрде бұл $2.73X10^{-7}$, бірақ Javascript көбейту белгісі және 10 саны-e- символынаауыстырылады
- **Логикалық маңыз** - екі: ақиқат (true) және жалған (false).
- **Қатарлы литерал** - таңбалар тізбегі, бірлік немесе қос тырнақшаға алынатындар. Мысалы: "сіздің атыңыз", 'аты'.

Айнымалылар

Айнымалылар деректерін сақтау үшін пайдаланылады. Айнымалылар **var** операторының көмегімен анықталады, одан кейін айнымалылар атауы керек. Айнымалылар атауы латын әліпбиінің әріптерімен немесе асты сызылған символмен басталуы тиіс. Атаулар латын әліпбиінің әріптерін, цифрлар және астын сызу белгісін қамтуы мүмкін.Мысалы:

var test
var _test
var _my_test1

Әрбір айнымалымен оны баптандыруға (жариялау), немесе сол бағдарламаның кодында мәнін тағайындауға болады. Меншіктеуоператоры тең (=) белгісімен белгіленеді, бірақмұнда теңдік белгісінің өзіндік тікелей мағынасыжоқ. Бұл жағдайда, олтек, яғни осы айнымалығатағайындалған мәнді көрсетеді.Мысалы:

var a=15
var b=23.15
var c='выполнено'
var s=true

Әрбір айнымалының айнымалы мәнімен анықталатын түрі бар. Осылайша мысалда: a және b айнымалы number түрін, c айнымалы string түрін, ал s айнымалы - логикалық түрін қабылдайды.

Өрнектер

Өрнектер литералдар, айнымалылар, операциялар белгілері және жақша арқылы құрылады. Өрнектерді есептеулер нәтижесінде жалғыз мән, ол сан болуы мүмкін, қатарлар немесе логикалық мән болады.

a*b өрнегі, a және b **операнд** деп аталады, ал * - операциялар белгісі. Javascript- келесі операциялар анықталған (4-кесте):

4-кесте

Операция	Атаулары
+	қосу
-	азайту
*	көбейту
/	бөлу
%	Бөлуден қалған бүтін сандар
++	Операнд мәнін бірлікке ұлғайту
--	Операнд мәнін бірлікке азайту

Операторлар солдан оңға қарай арифметикалық операциялардың басымдықтарына сәйкес мәнде есептеледі. Есептеулер тәртібін жақша көмегімен өзгертуге болады. Әрбір арифметикалық операторлар үшін форма бар, онда бір мезгілде, берілген операция біліктілігі орындалады(5-кесте). Бұл жағдайда алдымен оң операндесептелінеді, содан кейін алынған мән сол жақ операндаға беріледі.:

5-кесте

Оператор	Эквивалентті операторды меншіктеу
X+=Y	X=X+Y
X-=Y	X=X-Y
X*=Y	X=X*Y
X/=Y	X=X/Y
X%=Y	X=X%Y

Javascript-та салыстыруға мүмкіндік үшін екі маңызы бар салыстыру операциялары анықталды, соның нәтижесінде ғана логикалық мәні бар болуы мүмкін: *true* немесе *false* (6-кесте):

6-кесте

Операция	Атауы
<	Кіші
<=	Кіші немесе тең
==	Тең
!=	Тең емес
>=	Үлкен немесе тең
>	Үлкен

Javascript-та логикалық операциялар анықталған:

- *&&* - логикалық ЖӘНЕ (AND),
- || - логикалықНЕМЕСЕ (OR),
- *!* - логикалықТЕРІСТЕУ (NOT).

Логикалық операторлардың әсер ету нәтижесі операнд мәндерінің түрлі комбинацияларымен кестеде көрсетілген (7-кесте):

7-кесте

A	B	A&&B	A||B	!A
true	true	true	true	false
true	false	false	true	false
false	true	false	true	true
false	false	false	false	true

Басқаша айтқанда, A&&Bөрнектің мәні ақиқат болады, егер екі оператор ақиқат, жәнекарсы жағдайда жалған болса. A||B өрнектің мәні ақиқат болады, егер ең болмағанда бір операнда мәні ақиқат, және кері жағдайда жалған болса. Егер A операнда мәні ақиқат болса, онда !A - жалған, және керісінше.

Литерал қатары үшін конкатенация қатары операциясы анықталады, яғни олардың бірлестігі. Бұл операция плюс (+) белгісімен белгіленеді. Операциялардың орындалу нәтижесі қатар болып табылады. Мысалы:

var st1="Привет";
var st2="Вася";
var st3=st1+st2;

st3 айнымалы нәтижесінде мәні "Привет Вася" болады.

Операциялардың басым бағыттарының кестесін олардың кему тәртібімен келтірейік. Операциялардың басымдығы операциялар тұрғысынан орындалатын тәртібті анықтайды (8-кесте):

8-кесте

Атаулары	Белгілеулері
Инкремент	++
Декремент	--
Терістеу	!
Унарлы минус	-
Көбейту	*
Бөлу, бөлінді қалдығы	/,%
Қосу	+
Азайту	-
Салыстыру	<, >, <=, >=
Атаулары	**Белгілеулері**
Теңестіру	==
Теңдік емес	!=
Логикалық ЖӘНЕ	&&
Логикалық НЕМЕСЕ	||
Біліктіліктер	=, +=, -=, *=, /=, %=, !=

ТАҚЫРЫП 2. JAVASCRIPT-КОДЫН ҚАЙДА ОРНАЛАСТЫРУҒА БОЛАДЫ

Javascript-коды парақ ретінде, сондай-ақ сыртқы файлда орналастырылуы мүмкін.

Javascript-кодының парақта орналасуы

Javascript тілінде жазылған сценарийлер, html-парақта ***<script></script>*** тегтары арасында орналастырылуы мүмкін, олар өз кезегінде әдетте ***<head></head>***тегта орналасқан. ***<script>*** тегінде **language** параметрі көрсетілуі тиіс, ол скрипт жазу тілін көрсетеді:

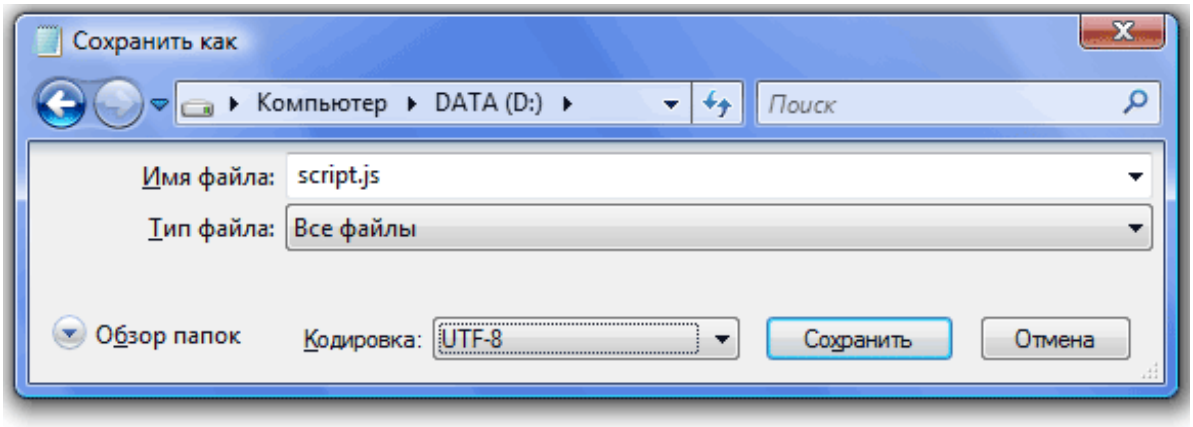
```
<html>
<head>
<title>Заголовок документа</title>
<link rel="stylesheet" type="text/css" href="style.css">
<script language="javascript"></script>
```

</head>
<body>
</body>
</html>

<script></script> теориялық тег құжаттың құрамында (*<body></body>* тег арасында) орналасуы мүмкін. Мұндай сценарий орындалады, егер браузер парақты осы орынға "есептесе" (*<script></script>* тегке дейін). Мұндай тәсілді пайдалану қазір қабылданған жоқ.

Javascript-коды сыртқы файлда орналасуы

Блокнот парағын (әзірге бос) құрыңыз, оны script.js ретінде сақтаңыз (1-сурет) және html-парақтар папкасына қойыңыз:



1-сурет. script.js файлын сақтау

Бұл *javascript*-коды бар парағымыз. Енді *script.js* – парағын html-парағына қосу керек. Бұл үшін, естеріңізде болса, html *<script>* тег бар. Айта кетейік, бұл тег-html-парақта :

<html>
<head>
<title>Заголовок документа</title>
<link rel="stylesheet" type="text/css" href="style.css">
<script type="text/javascript" src="script.js"></script>
</head>
<body>
</body>
</html>

ТАҚЫРЫП 3. БІРІНШІ БАҒДАРЛАМА, ОҚИҒАЛАРДЫ ӨҢДЕУ

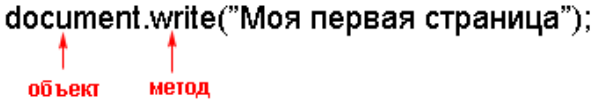
Бірінші бағдарламаны жазайық. Бұл үшін алып тастауды жасаймыз және кез келген javascript-кодтарын html-параққа енгіземіз. Сонымен, (немесе жасаңыз) html-парағын ашыңыз, оған келесі кодты салыңыз:

```
<html>
<head>
<title>Заголовок документа</title>
<link rel="stylesheet" type="text/css" href="style.css">
<script language="javascript">
    document.write("Моя первая страница.");
</script>
</head>
<body>
</body>
</html>
```

Осы парақты браузерде ашыңыз, ол былайша көрінеді:

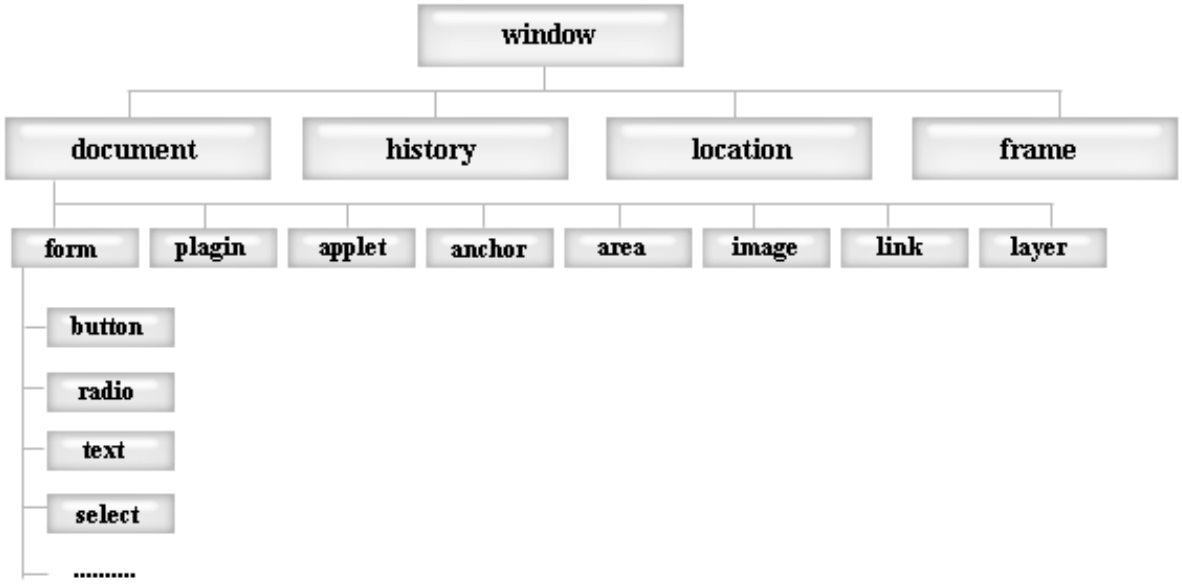
Моя первая страница.

Бұл қалай жұмыс істейтінін талдап көрейікші. Браузер html-парақты оқиды, *document.write("Моя первая страница.")* орындау үшін операторды көреді; және оны орындайды. Оператор (нұсқаулық) өзі неден тұратынын қарастырайық (3-сурет):



2-сурет.*document.write*

html-парақты түсіндіру кезінде браузер javascript объектілерін жасайды. Олар иерархиялық құрылым түрінде сақталады, құжат құрылымынкөрсетейік (1-схема), мысалы:Ең жоғарғы деңгейде, браузердің терезесін білдіретін және қалған барлық объектілерге **"нақты"** болып табылатын*window*объектісі орналасқан. Төменде өзіне бағынышты объектілер орналасқан болуы мүмкін. Сонымен *document* (ағымдағы парақ) объектісіне *form* (форма) объектісі еншілес болуы мүмкін және т.б.



1-схема. Иерархиялық құрылым

Барлық объектілердің әдістері бар (объектіден нүктемен бөлектенеді), мысалы: *document.write* ағымдағы парақта мәтін жазуға мүмкіндік береді, *window.open* браузердің жаңа терезесін ашады. Сондай-ақ обьектілердің қасиеттері бар, мысалы: ағымдағы парақтың *document.bgcolor* фондық түсінің құрамындағы мәні, *document.title* құрамында парақ тақырыптары.

Барлық бағдарламалық кодтыңнұсқаулары нүктелі үтірмен аяқталады. Бұл туралы есте сақтау керек.

Біз жазған сценарилер, парақты жүктеуден кейін дереу орындалды. Алайда, көптеген жағдайларда бір пайдаланушының қандай да іс-шараларынанкейін сценаридің орындалғаны керек: батырманы басу немесе мәтін енгізу. Яғни, қолданбалы оқиғаға жауап. Бұл үшін парақ элементтері тегтерінде оқиғаларды өңдеу параметрлеріне, элементпен байланысты оқиғалар туындаған кезде орындалатын, іс-әрекеттер көрсетіле енгізілген. Мысалы:

* *

Мұнда *Click* - оқиға (*div-y* түймешігін басыңыз), *onClic*k – өңдеуші оқиғалар, *addText()* - функциялар атауы, осы оқиғалар іске қосылған кезде (*div-y* түймешігін басыңыз) жұмыс жасайды.

Функциялардыкелесі Тақырыпта қарастыратын боламыз, ал қазір *javascript* қолдайтын оқиғалардытізімдейміз. Оларды есте сақтау керек емес, әрі қарай жай ғана 9-кестеге жүгіне аласыз.

9-кесте

Оқиғалар	Ол қай кезде туындайды	Оқиғаларды өңдеуші
Blur	фокусты жоғалту объектісі	onBlur
Change	қолданушы элементтің мәнін өзгертеді	onChange
Click	қолданушы тышқан бойынша объектіні шертеді	onClick
DblClick	қолданушы тышқанныңқос түймешігі бойынша объектіні шертіңіз	onDblClick
DragDrop	қолданушы объектіні тышқанмен тінтеді	onDragDrop
Error	javascript-қателердің пайда болуы	onError
Focus	терезе немесе форма элементін фокус алады	onFocus
KeyDown	қолданушы пернетақта пернесін басады	onKeyDown
KeyPress	қолданушы пернетақта пернесін басып ұстайды	onKeyPress
KeyUp	қолданушы пернетақта пернесін юосатады	onKeyUp
Load	құжат браузерге жүктеледі	onLoad
MouseDown	қолданушы тінтуір түймесін басады	onMouseDown
MouseOut	тышқан тінтуірі элемент шегіне орналастырылады	onMouseOut
MouseOver	тышқан тінтуірі элемент үстіне орналастырылады	onMouseOver
MouseUp	қолданушы тінтуір түймесін босатады	onMouseUp
Move	қолданушы терезені ауыстырады	onMove
Reset	қолданушы форманың "reset" батырмасын басады	onReset
Resize	қолданушы терезе немесе элемент мөлшерін өзгертеді	onResize
Select	қолданушы форманың элементін таңдайды	onSelect
Submit	қолданушы форманың "submit"батырмасын басады	onSubmit
Unload	қолданушы құжатты жабады	onUnload

ТАҚЫРЫП4. JAVASCRIPT-ФУНКЦИЯЛАРДЫ ҚҰРУ

Функция - бұл операторлардың (нұсқаулардың) атындағы реті. Кез келген функция келесі синтаксисті иеленеді:

```
function имя (){
оператор;
.....
оператор;
}
```

Алдымен *function* кілт сөзі өтеді, содан кейін функциялардың атауы, содан соң дөңгелек жақша ішінде параметрі (егер олар болса) аударылады, одан кейін фигуралы жақшаның ішінде операторлары аударылады, яғни дәйектілігі орындалатын іс-әрекеттер. Әрбір оператор нүктелі үтірмен аяқталады.

Мысалда қарастырайық. Екінші сабақта html-парақты, функциялар үшін *script.js* парағына біріне бірін қостық. Біз бұл парақтарды пайдаланатын боламыз. Кез келген сайттың html-парағына келесі кодты жазамыз:

```
<html>
<head>
<title>Заголовок документа</title>
<link rel="stylesheet" type="text/css" href="style.css">
<script type="text/javascript" src="script.js"></script>
</head>
<body>
<div onClick="showMessage();">Щелкни меня</div>
</body>
</html>
```

Пайдаланушының div-у ("Щелкни меня" сөзімен) басқан кезде "*showMessage*" атаулы функциясы туындауы мүмкін болуы тиіс. Енді script.js парағында "*showMessage*" функциясының өзін жазу керек. *script.js* парағын ашамыз және жазамыз:

```
function showMessage(){
  alert ("Вы щелкнули по div-y");
}
```

"*showMessage*" функциясы бір әрекетті орындайтын болады - мәтінмен ескерту терезесін көрсетеді: "Вы щелкнули по *div-y*". *alert* - бұл javascript стандартты функциясы, және ескертулер терезесін шығарады, біз тек осы терезе үшін мәтін қоямыз.

Javascript-та стандартты функциялар жиынтығы бар, оларды пайдалануға болады (мысалы, *alert*) және қолданылатын функциялар деп аталады, яғни, өзіміз жасаймыз. Кейбір *javascript* функцияларын қажеттілігіне қарай зерттеп, кейбірі жекелеген сол сабақтарға арналады.

html-парағын браузер көмегімен ашыңыз және div-у бойынша шертіңіз.

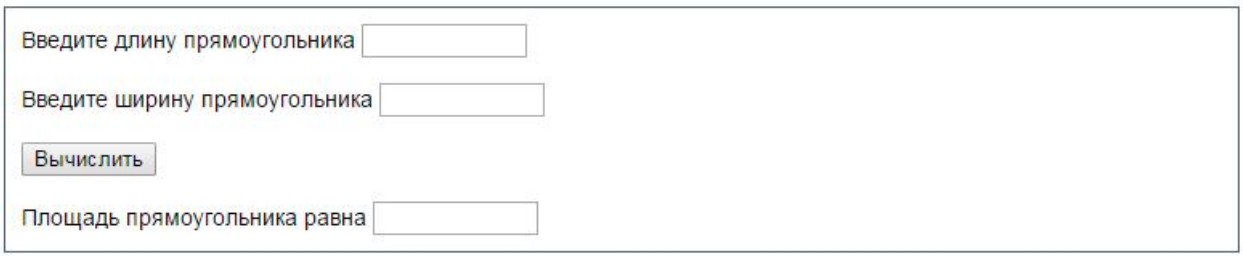
Осылайша, оқиғаның өңдеу жұмыс механизмін қарастырамыз:

1. Html-парақта қажетті элементке өңдеуші оқиғаларды (*onClick, onMouseUp* және т.б.) байланыстырамыз, мәні ретінде функциялардың атауын көрсетеміз, оқиға болатын жағдайда онда жұмыс істеуге тиістіні.
2. .js кеңейтуімен парақта, осы функциялардың кодын жазамыз, яғни нұсқаулар орындалуы тиіс, егер функция инициализацияланса (оған жүгінеді).

Тапсырманы қиындатып және сценарий жазайық, пайдаланушы енгізген ұзындығы және ені бойынша төртбұрыштың ауданын есептеу. Бұл үшін алдымен, html-парағында қажетті форманың элементтерін жариялаймыз:

```
<html>
<head>
<title>Расчет площади прямоугольника</title>
<link rel="stylesheet" type="text/css" href="style.css">
<script type="text/javascript" src="script.js"></script>
</head>
<body>
<form name="forma1">
   Введите длину прямоугольника <input type="text" name="t1"
size="10"><br><br>
   Введите ширину прямоугольника <input type="text" name="t2"
size="10"><br><br>
<input type="button" name="button" value="Вычислить"><br><br>
   Площадь прямоугольника равна <input type="text" name="res"
size="10">
</form>
</body>
</html>
```

Браузерде парақ осылай көрінеді:



Сонымен, қолданушы ені мен ұзындығын енгізеді, және "Вычислить" батырмасын басады. Содан кейін, алаңда көлем нәтижесі пайда болуы тиіс. Осылайша, " Вычислить"бастырмасына басқан кезде оқиға басталады, яғни дәл оған өңдеуші оқиғаларды байланыстырамыз. Көлемді есептеу функциясын"areaRectangle" деп атайық:

```
.....
<input type="button" name="button" value="Вычислить"
                              onClick="areaRectangle();"><br><br>
.....
```

Енді "areaRectangle" функциясын жазу керек. Ол үшін script.js парағын ашамыз және функцияларды дайындау үшін жазамыз:

```
function areaRectangle(){
}
```

Енді функциялардың денесін жазу керек. Ол үшін үш айнымалыны жария етеміз: a - мәні төртбұрыштың ұзындығы, b - мәні төртбұрыштың ені, s - тік төртбұрыш көлемі:

```
function areaRectangle(){
  var a;
  var b;
  var s;
}
```

a мәні *(value)*ағымдағы парақтан (*document*), "*forma1*" атаулы формадан, "*t1*" атаулы мәтіндік өрістен алынуы тиіс. Бұл *document.forma1.t1.value* жазылады, яғни нақтыдан қалаған объектілерімізді аттарын нүкте арқылы аударамыз (алдыңғы сабақта иерархиялық құрылым объектілерін талқыладық). Соңында қажетті объектінің қасиеті (*value*) көрсетіледі.

Сол сияқты*b* - *document.forma1.t2.value*.–ға мәндері үшін маңызы бар.Ал *s* айнымалының *a*-дан *b*-ға туындысы бар. Бұларды дене функцияларында жазамыз:

```
function areaRectangle(){
  var a=document.forma1.t1.value;
  var b=document.forma1.t2.value;
  var s=a*b;
```

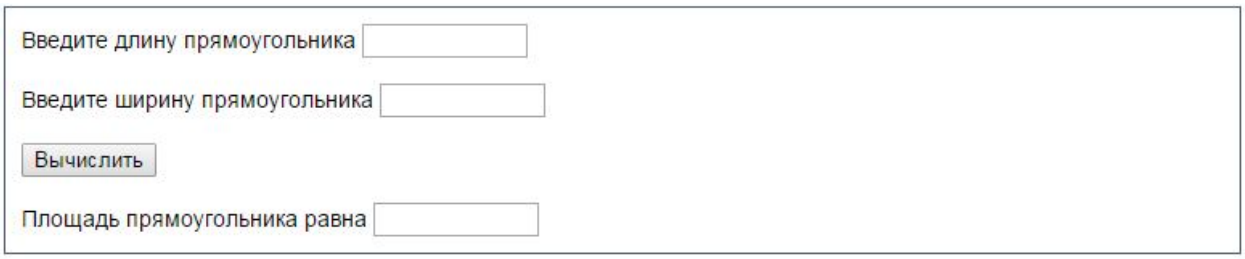
}

Формаларға көлемді есептеуді мәтіндік өрісте "*res*" атымен жазудың жазу нұсқаулығы қалды. Яғни, ағымдағы параққа, форманың "forma1" атауымен, мәтіндік өрістің "res" атауымен, мәні (*value*) ретінде s мәні берілді. Сонымен жазамыз:

```
function areaRectangle(){
  var a=document.forma1.t1.value;
  var b=document.forma1.t2.value;
  var s=a*b;
  document.forma1.res.value=s;
}
```

Алдымен a және b айнымалы мәндерін формалар ішінен бердік, содан кейін қажетті есептеулер жасалды, одан кейін белгілі бір форма элементіне алынған s мәні берілді.

Браузердің өзінде html-парақтың жұмысын тексеріңіз. Егер сіз бәрін дұрыс жасасаңыз, онда жұмыс істеуі тиіс, төмендегі мысалдай:



Міне, бірінші сценариімізді жаздық. Мүмкін, сізге қазір, бұлардың барлығы тым қиын көрінер. Бірақ бұл тек бірінші көзқарас, жай ғана көптеген жаңа терминдер мен ұғымдар берілген. Бірнеше сабақтан кейін сіз объектілер және функциялар санаттарын ойлауды үйренесіз.

ТАҚЫРЫП 5. ФУНКЦИЯ ПАРАМЕТРЛЕРІ

Өткен сабақта біз сценарий жазып, пайдаланушы төртбұрыштың ауданын есептеу бойынша ұзындығы мен енінің мәндері енгіздік. Сондай-ақ, функция жаздық, ол жеке және есептеу жүргізеді. Егер бірнеше html-парақ болса, төртбұрыштың ауданын есептеуге керек болады. Бізге әрбіреуіне өз функциясын жазуға тура келеді? Егер сценаридіқазіргі түріндеқалдыратын болса, онда - ИЯ. Бірақ, сіз түсінесіз бе, өте ыңғайсыз.

Функцияны ақылға қонымдыбір рет жазу және одан әрі барлық html-парақтарында пайдалануға арналған түрдежазу. Ол үшін html-парақта қандай функциялар болуы тиіс,есептеулерге арналған қандай мән бар (қай парақтан) екенін көрсету. Міне осы жерде параметрлер қажет болады (дөңгелек жақша ішінде көрсетіледі).Форманың коды:

```
<html>
<head>
<title>javascript параметры</title>
<link rel="stylesheet" type="text/css" href="style.css">
<script type="text/javascript" src="script.js"></script>
</head>
<body>
<form name="forma1">
   Введите длину прямоугольника <input type="text" name="t1"
size="10"><br><br>
   Введите ширину прямоугольника <input type="text" name="t2"
size="10"><br><br>
<input type="button" name="button" value="Вычислить"
onClick="areaRectangle();"><br><br>
   Площадь прямоугольника равна <input type="text" name="res"
size="10">
</form>
</body>
</html>
```

Ал функция келесілер болады:

```
function areaRectangle(){
  var a=document.forma1.t1.value;
  var b=document.forma1.t2.value;
  var s=a*b;
  document.forma1.res.value=s;
}
```

Функцияларда форма атын қолданамыз - *forma*1, оның біз параметр жасаймыз. Яғни, енді функцияны келесі түрде жазамыз:

```
function areaRectangle(obj){
  var a=obj.t1.value;
  var b=obj.t2.value;
  var s=a*b;
  obj.res.value=s;
}
```

Яғни, бұл функция параметрді қандай да бір объект (*obj*) ретінде қабылдауы тиіс және онымен барлық іс-әрекеттерді орындайды. html-парақта объекті атын көрсету:

```
.....
<input type="button" name="button" value="Вычислить"
                          onClick="areaRectangle(forma1);"><br><br>
.....
```

Енді қандай да бір басқа парақта ауданды есептеу қажет болса, онда areaRectangle() функциясын шақыру жеткілікті болады, ал жақшаның

ішінде форма атауларын көрсету керек. Сценарий жұмысын браузерде тексеріңіз.

Функция параметрлері ретінде кез келген мәнді беруге болады, әрі олар бірнеше болуы мүмкін. Осындай міндетті қарастырайық, мысалы: үш квадрат бар, әрбіреуін басқан кезде квадрат түсі туралы хабарламалар терезесін көрсетіп отырады, шертілген бойынша (төмендегі квадрат).

Сонымен, алдымен script.js параққа функция кодын жазамыз, ол хабарламалартерезесін ашады, оны "message"деп атайық. Естеріңізде болса, хабарламаларды шақыру терезесінjavascript-тің стандартты функциясыжүзеге асырады - *alert*, ал мәтін хабарларын параметршеберуге белгілейміз, оны "*m*"ретінде таңбалаймыз:

```
function message(m){
  alert (m);
  }
```

Енді html-парақта квадраттардың өздерінің кодын жазамыз:

```
<html>
<head>
<title>javascript параметры</title>
<link rel="stylesheet" type="text/css" href="style.css">
<script type="text/javascript" src="script.js"></script>
</head>
<body>
<table><tr>
<td><div id="red" onClick="message('Вы щелкнули по красному
        квадратику');"></div></td>
<td><div id="green" onClick="message('Вы щелкнули по зеленому
        квадратику');"></div></td>
<td><div id="blue" onClick="message('Вы щелкнули по синему
        квадратику');"></div></td>
</tr></table>
</body>
</html>
```

Біз квадраттарды басқан кезде "*message*" функциясы жұмыс істей бастауы тиіс, ал параметрлер ретінде қажетті хабарлардың мәтіні көрсетілген (олар функцияларына жүгінген кезде "m" параметр орнына қойылады).

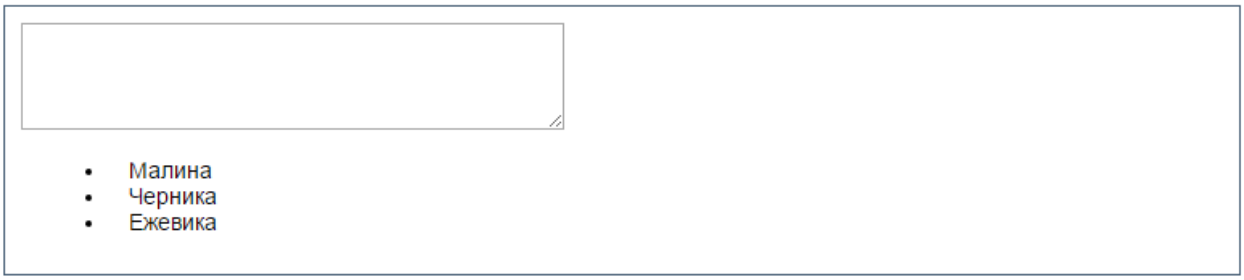
Енді квадраттарды әсемдеу қалды. Бұл үшін style.css парағында олар үшін стил жазамыз:

```
#red, #green, #blue{
```

```
 width:100px;
 height:100px;
 margin:10px;
}
#red{
 background:red;
}
#green{
 background:green;
}
#blue{
 background:blue;
}
```

Енді сценарий жұмысын браузерде тексеру қажет.

Алдыңғыларды біріктіретін тағы бір мысал қарастырайық. Бізде жидектер тізімі болсын, ал сіз жеміс-жидектер атауына тінтуірді апарғанда, оның сипаттамасы мәтіндік өрісте пайда болады.



Мұндай конструкцияның бір парақта пайдалы болуы мүмкін емес, сондықтан екі параметр керек: бірі объекті атымен (форма), екіншісі - жидектерді сипаттауға. Мұндай жағдайда параметрлері үтір арқылы жазылады. Сонымен, функция былайша жазылады:

```
function showDesc(obj, n){
  obj.desc.value=n;
}
```

desc - мәтіндік өріс аты сипаттамаларын шығару үшін. Бұл функция жұмысы мынадай жағдайларда іске қосылатын болады, егер ол кезде жидектер атауына меңзер апарылса, бірақ тағы бір функция мәтіндік өрісін тазалау үшін қажет, қашан меңзер атаулары шегінен шыққанда. Оны "delete" деп атайық:

```
function delet(obj){
  obj.desc.value=' ';
}
```

онда ‘’ дара тырнақша бос орын, бос қатарды білдіреді.

Енді нақты парақ кодын жазамыз. Яғни, өңдеуші оқиғалар, тінтуір меңзері элементтің үстіне орналастырылады, онда *onMouseOver*деп

аталады, ал оқиғаларды өңдеуші, тінтуір меңзері элементтен тыс кезде - *onMouseOu*t.

```
<html>
<head>
<title>javascript параметры</title>
<link rel="stylesheet" type="text/css" href="style1.css">
<script type="text/javascript" src="script.js"></script>
</head>
<body>
<form name="forma2">
<textarea name="desc" cols=45 rows=4></textarea>
</form>
<ul>
<li onMouseOver="showDesc(forma2,'Малина обыкновенная — кустарник с многолетним корневищем,'+
        ' из которого развиваются двухгодичные надземные стебли высотой до полутора метров.');"
        onMouseOut="delet(forma2);">Малина</li>
<li onMouseOver="showDesc(forma2,'Черника — кустарничек высотой 15—30 см. Ветви отходят'+
        ' от главного стволика под острыми углами.');"
        onMouseOut="delet(forma2);">Черника</li>
<li onMouseOver="showDesc(forma2,'Ежевика — название нескольких видов растений из рода Rubus'+
        ' семейства Розовые.');"
        onMouseOut="delet(forma2);">Ежевика</li>
</ul>
</body>
</html>
```

Назар аударыңыз, мәтін параметрі бірдей бағамдарда жасалады, ал қатарларды ауыстыруғақатарлардың конкатенация операциясы пайдаланылады, яғни әрбір бөлігі бірдей бағамдарда болып табылады және бұл бөлігі +операторы арқылы қосылады. Сіз өзіңізде конкатенация қатарын пайдаланбауыңызға болады, жай ғана мәтінді бір қатарға жазыңыз.

ТАҚЫРЫП6. JAVASCRIPT-ТА MATH ОБЪЕКТІСІ

Жоғарыда атап өткеніміздей, javascript-та кейбір стандартты объектілер және функциялар анықталған. math объектісін және оның әдістерін қарастырамыз.

math объектісі математикалық функциялармен жұмыс істейді, ал оның әдістеріносы функцияларды жүктеу үшін қолдануға болады. Төменде math объектінің кейбір әдістері көрсетілген (10-кесте):

10-кесте

Әдіс	Анықтама
abs	абсолюттік мәні
sin, cos, tan	тригонометриялық функциялар
log	натуральды логарифм
exp	экспонента
pow	көрсеткіш функция
sqrt	квадрат түбірі
min	ең аз мән
max	ең үлкен мән

Сіз кейбір әдістерді жиі, кейбірін сирек пайдаланасыз, ал кейбіреуі сізге мүлдем қажет болмайды. *math* объектісін пайдалану мысалын қарастырайық. Егер үшбұрыштың ауданын үш қабарғасымен есептеуге сценарий жазу қажет болса, ол үшін Герон формуласын пайдалану керек:

$S=\sqrt{p(p-a)(p-b)(p-c)}$, мұндағы $p=\frac{a+b+c}{2}$

яғни: S – алаңы; a, b, c - үшбұрыштың қабырғасының ұзындығы.
html-парақта форма кодын жазамыз:

```
<html>
<head>
<title>math javascript</title>
<link rel="stylesheet" type="text/css" href="style1.css">
<script type="text/javascript" src="script.js"></script>
</head>
<body>
<form name="forma3">
        Сторона 1 <input type="text" size="8" maxlength="8"
          name="st1">
        Сторона 2 <input type="text" size="8" maxlength="8"
          name="st2">
```

```
        Сторона 3 <input type="text" size="8" maxlength="8"
          name="st3"><br><br>
<input type="button" value="Вычислить"
          onClick="areaOfTriangle(forma3);">
<input type="reset" value="Отменить">
        Результат <input type="text" size="8" maxlength="8"
          name="res">
</form>
</body>
</html>
```

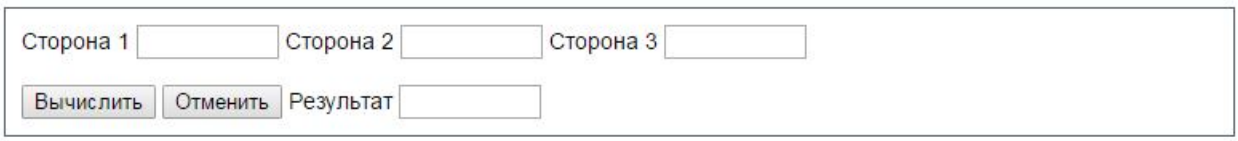
Мұнда ешқандай жаңалық жоқ. Енді script.js парағында функциялар кодын жазамыз:

```
function areaOfTriangle(obj){
  var a=1*obj.st1.value;
  var b=1*obj.st2.value;
  var c=1*obj.st3.value;
  var p=(a+b+c)/2;
  var s=Math.sqrt(p*(p-a)*(p-b)*(p-c));
  obj.res.value=s;
}
```

Бұл жерде *Math* объектісін пайдаландық және оның *sqrt* әдісі квадрат түбірді алу. Түбірден алынған өрнек, жақшаға алынуы тиіс.

Сондай-ақ, алғашқы функцияның үш қатарына назар аударыңыз, олар 1* басталады, a, b, c айнымалыларды бірлікке көбейттік. Не үшін? Есіңізде болсын, бірінші сабақта -әр айнымалының түрі автоматты түрде анықталады. Өйткені айнымалылар мәтіндік өрістен келеді, онда олар *string* болады, яғни функцияғасандар әріптер ретінде қабылданады.

Егер бұл айнымалыларды көбейтетін болсақ, онда олардың типін автоматты түрде *number* анықтар еді, бірақ оларды қосамыз, ал бұл жағдайда+ белгісі операциялар, конкатенация қатарларымен бағаланады. Осы болмауы үшін, айнымалыларды бірлікке көбейттік, оларды осылайша *number*түріне айырбастадық.Эксперимент үшін көбейту бірлігін осы үш қатарда алып тастаңыз және қараңыздаршы, бұл "санау" сценариіболады. Содан кейін дұрыс нұсқасын беріңіз және барлығы дұрыс жұмыс істейтінін көріңіз:



Бәрі жақсы, бірақ, бір ескеретін жайт: егер алынатын түбір бөлшекті сан болып табылатын болса, онда бөлшек бөлігі шексіз ұзын болуы

мүмкін. Дәл есептеулер үшін бұл қажет, бірақ көп жағдайларда үтірден кейін екі белгі жеткілікті.

Нәтижені үтірден кейін N-ге дейінгі белгімен дөңгелектеу үшін *Number* объектінің *toFixed* әдісін пайдалануға болады.

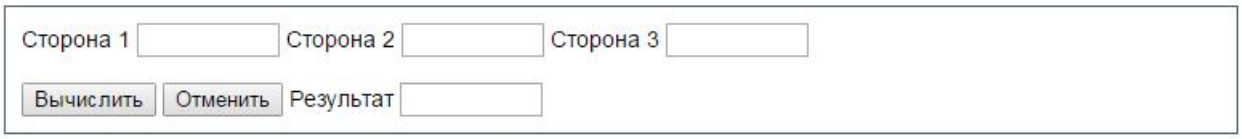
Синтаксис жазбалар:

(x).toFixed(N)

мұнда x - дөңгелектеуге қажет сан, ал N - үтірден кейін алынатын белгі саны.

Енді функцияларды, нәтижесі үтірден кейін 2 белгіге дейін дөңгелектейміз:

```
function areaOfTriangle(obj){
  var a=1*obj.st1.value;
  var b=1*obj.st2.value;
  var c=1*obj.st3.value;
  var p=(a+b+c)/2;
  var s=Math.sqrt(p*(p-a)*(p-b)*(p-c));
  s=s.toFixed(2);
  obj.res.value=s;
}
```



ТАҚЫРЫП 7. ТАРМАҚТАЛУ БАҒДАРЛАМАСЫ - IF ОПЕРАТОРЫ.

Қашанда қандай да бір шарттарды бір әрекет түрінде орындау өте жиі кездеседі. Мысалы, киімдер интернет-дүкені. Біз пайдаланушыдан оның кім екенін сұрадық (ер адам немесе әйел адам) және алынған жауапқа байланысты тиісті тауарлардың (ерлер немесе әйелдер) тізімін көрсетеміз. Осындай бағдарламалар жазу кезінде *if* шартты операторы пайдаланылады. Оның синтаксисі келесі:

if B {S1}

else {S2}

мұнда B - өрнектің логикалық типті, ал S1 және S2 - операторлар.

Оның жұмыс істеуі: формуланы есептеу мәні B, егер ол ақиқат болса, онда S1операторы орындалады, егер ол жалған болса, онда S2операторы орындалады. else {S2} қатарын түсіру керек болады.Мысал бойынша түсінікті болады. Бізде форма бар болсын, оған қолданушы 3 маңызы бар мән енгізеді. Енгізілген сандардың барынша максималын анықтайтын сценарий жазамыз.

Ол үшін html-парақта форманың кодын жазамыз:

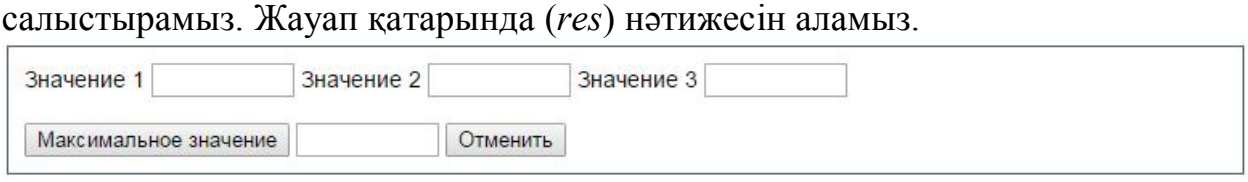
```
<html>
<head>
```

```
<title>javascript if</title>
<link rel="stylesheet" type="text/css" href="style1.css">
<script type="text/javascript" src="script.js"></script>
</head>
<body>
<form name="forma4">
        Значение 1 <input type="text" size="8" maxlength="8"
        name="zn1">
        Значение 2 <input type="text" size="8" maxlength="8"
        name="zn2">
        Значение 3 <input type="text" size="8" maxlength="8"
        name="zn3"><br><br>
<input type="button" value="Максимальное значение"
        onClick="maxZnach(forma4);">
<input type="text" size="8" maxlength="8" name="res">
<input type="reset" value="Отменить">
</form>
</body>
</html>
```

Енді script.js парағына функциялар кодын жазамыз:

```
function maxZnach(obj){
  var a=1*obj.zn1.value;
  var b=1*obj.zn2.value;
  var c=1*obj.zn3.value;
  var m=a;
  if (b>m) m=b;
  if (c>m) m=c;
  obj.res.value=m;
}
```

Сонымен, функция формадан үш мәндіқабылдайды, *a* мәні ең үлкені *(m)*болады. Содан кейін салыстырамыз: егер *b*-ның мәні максималдыдан (яғни *a*)артық болса, онда максимальдыға*b*айналады, олай болмаған жағдайда, *a*ең үлкен болып қалада (яғни, жақша ішіндегіөрнек, ақиқатемес болып табылады). Бұдан әрі келесі *c*мәнінмаксималдыменсол сияқты салыстырамыз. Жауап қатарында (*res*) нәтижесін аламыз.



Жалпы, мұндай сценариді Math объектінің max әдісін пайдалана отырып жазуға болады, бұл өткен сабақта қаралған және коды қысқа болар еді:

```
function maxZnach(obj){
  var a=1*obj.zn1.value;
```

```
  var b=1*obj.zn2.value;
  var c=1*obj.zn3.value;
  obj.res.value=Math.max(Math.max(a,b),c);
}
```

Бағдарламалау бұл шығармашылық процесс, және бір тапсырманы әр түрлі тәсілдермен шешуге болады. Бағдарламашы міндеті ең оңтайлы нұсқасын табу. Бұл лирикалық шегініс. if шартты операторға оралайық және қызықты мысалды қараймыз. Бейнеге тінтуір меңзерін әкелген жұмыс уақыты кезінде, ол өсім жасай отырып, жақындау әсерін тудыратын сценарий жазамыз.

Егер бейненің берілген мөлшерлері түпнұсқадан көп немесе аз болса, онда браузер автоматты түрде түпнұсқаның бұл мөлшерінреттеуге болады. Бізде осындай 3-сурет болсын:

3-сурет. Автоматты түрдесурет мөлшерінреттеу

Түпнұсқаның ені 302 пиксел. Парақта суреттің енінің 102 пиксел болғанын қалаймыз, ал меңзерді басқанда 302 пикселге дейін жоғарылайды. Сәйкес html-парақта бәрі түсінікті:

```
<html>
<head>
<title>javascript if</title>
<link rel="stylesheet" type="text/css" href="style1.css">
<script type="text/javascript" src="script.js"></script>
</head>
<body>
<img src="images/tigrenok.jpg" name="tigr" width="102"
      onMouseOver="bigPict()">
</body>
</html>
```

Ал функцияларда шартты операторлардан басқа, тағы *javascript* стандартты функциясы *setTimeout* қолданамыз, ол берілген уақыт аралығымен пайдаланушының функциясын тудырады:

```
function bigPict(){
    var w=document.tigr.width;
    if (w<302){
```

```
document.tigr.width=w+10;
document.tigr.src="images/tigrenok.jpg"
setTimeout("bigPict()", 500)
}
}
```

Осылайша, функция суреттердің енін (*width*) тексереді, егер ол 302 пикселден аз болса, онда бұл енді 10 пикселге арттырады. setTimeout функциясы *bigPict*функциясын әрбір жартысекундта туғызады, соның арқасында суреттер мөлшері*w<302* шарты жалған болғанша ұлғаятын болады

ТАҚЫРЫП 8. SWITCH ТАҢДАУ ОПЕРАТОРЫ.

Алдыңғы сабақта if операторымен таныстық. Ол пайдаланушының таңдауы бойынша, қандай да бір әрекетке байланысты қашан жасалса да ыңғайлы. Бірақ егер осындай таңдаулар көп болса? Мысалы, ауа-райын аптасынан бұрын білсек. Ал ауа-райын қолданушы таңдаған бір күн бойынша беру. Бұл қалай жүзеге асырылады?Бірінші нұсқасы - if операторын пайдалану. Ол үшін html-парақта форманың өзін жазамыз:

```
<html>
<head>
<title>javascript switch</title>
<link rel="stylesheet" type="text/css" href="style1.css">
<script type="text/javascript" src="script.js"></script>
</head>
<body>
    Прогноз погоды на какой-день Вас интересует:<br><br>
<form name="forma5">
<select name="day" size="7">
<option selected value="mon">понедельник
<option value="tue">вторник
<option value="wen">среда
<option value="thu">четверг
<option value="fri">пятница
<option value="sat">суббота
<option value="san">воскресенье
</select>
<input type="button" value="OK" onClick="Vday(forma5);">
<br><br>
<textarea cols="35" rows="5" wrap="virtual" name="res"></textarea>
</form>
</body>
</html>
```

Енді script.js парағындафункциялар кодын жазамыз:

```
function Vday(obj){
  if (obj.elements[0][0].selected)
   obj.res.value="В понедельник будет ветрено, температура воздуха
поднимется до +20 градусов";
  if (obj.elements[0][1].selected)
   obj.res.value="Во вторник будет солнечно, температура воздуха
поднимется до +25 градусов";
  if (obj.elements[0][2].selected)
   obj.res.value="В среду будет прохладно, температура воздуха
опустится до +17 градусов";
  if (obj.elements[0][3].selected)
   obj.res.value="В четверг будет пасмурно, температура воздуха
опустится до +10 градусов";
  if (obj.elements[0][4].selected)
   obj.res.value="В пятницу будет облачно, температура воздуха
поднимется до +15 градусов";
  if (obj.elements[0][5].selected)
   obj.res.value="В субботу будет ветрено, температура воздуха
поднимется до +27 градусов";
  if (obj.elements[0][6].selected)
   obj.res.value="В воскресенье будет ясно, температура воздуха
поднимется до +30 градусов";
}
```

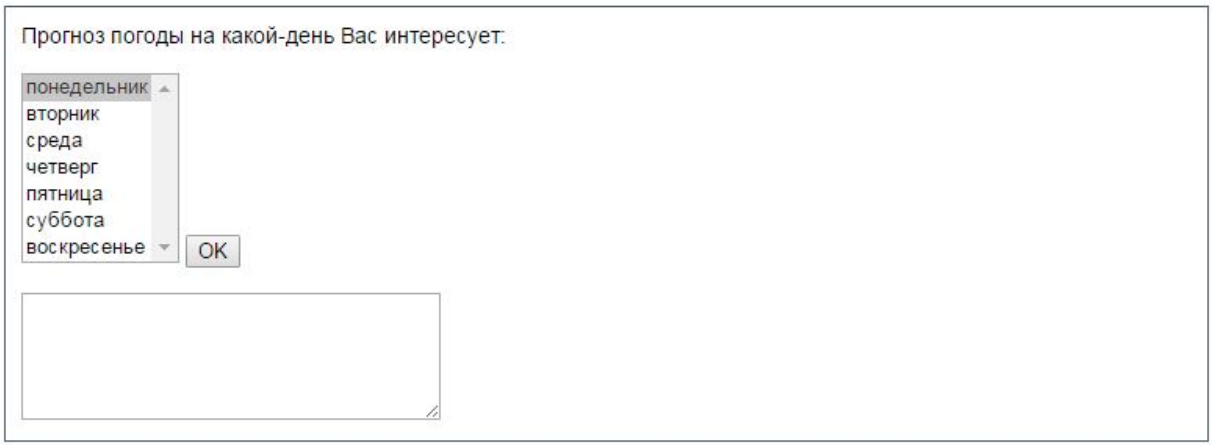
Мұнда кейбір түсініктемелер беру қажет. Сценарий қай элементтің таңдалғанын біледі.Формада үш элемент бар: select тегімен анықталатын тізім, OK батырмасы және көпқатарлы мәтіндік өріс. Сценарий барлық форма элементтерін жоғарыдан төмен қарай 0 бастап нөмірлейді. Сондықтан, тізімнің айналыс тәсілі былай көрінеді:

obj.elements[0]

Тізімдегі барлық элементтер, тиісінше 0-ден 6 дейін нөмірленген. Ал таңдалған элементтің, selected қасиеті бар. Сондықтан қатар

obj.elements[0][0].selected

Объектінің бірінші элементінің бірінші тармағының таңдалғанын білдіреді (бірінші тармақ форманың тізімі). Жалпы, бұл элементпен байланысудың ең ыңғайлы тәсілі емес, бірақ бұл туралы егжей-тегжейлі келесі сабақта айтатын боламыз, ал қазір мысалға оралайық.Сізге қай күннің ауа-райы керек:



Барлығы жұмыс істейді. Біз мәселенің шарттарын өзгерту туралы шешім қабылданды делік. Пайдаланушы енді апта күнін таңдамайды екен делік, және оны мәтіндік өріске ендіреді.

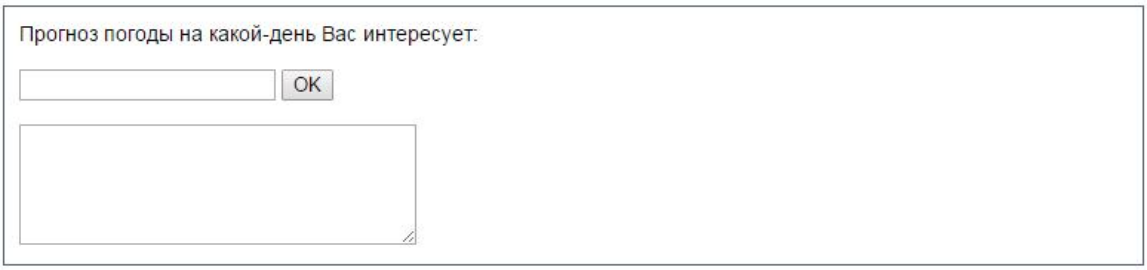
HTML-парақтар коды осы сияқты көрінеді:

```
<html>
<head>
<title>javascript switch</title>
<link rel="stylesheet" type="text/css" href="style1.css">
<script type="text/javascript" src="script.js"></script>
</head>
<body>
      Прогноз погоды на какой-день Вас интересует:<br><br>
<form name="forma6">
<input type="text" size="15" maxlength="15" name="zn">
<input type="button" value="OK"
       onClick="Vday1(forma6);"><br><br>
<textarea cols="35" rows="5" wrap="virtual" name="res"></textarea>
</form>
</body>
</html>
```

Ал функцияларға қайтадан 7 рет if операторын жазуға тура келеді, тағыда нұсқаларды қарау қажет, егер қолданушы орфографикалық қателікке жол берсе:

```
function Vday1(obj){
  var a=obj.zn.value;
  if (a=="понедельник")
   {obj.res.value="В понедельник будет ветрено, температура
    воздуха поднимется до +20 градусов";}
   else
  if (a=="вторник")
   {obj.res.value="Во вторник будет солнечно, температура воздуха
    поднимется до +25 градусов";}
   else
```

```
if (a=="среда")
 {obj.res.value="В среду будет прохладно, температура воздуха
  опустится до +17 градусов";}
 else
if (a=="четверг")
 {obj.res.value="В четверг будет пасмурно, температура воздуха
  опустится до +10 градусов";}
 else
if (a=="пятница")
 {obj.res.value="В пятницу будет облачно, температура воздуха
  поднимется до +15 градусов";}
 else
if (a=="суббота")
 {obj.res.value="В субботу будет ветрено, температура воздуха
  поднимется до +27 градусов";}
 else
if (a=="воскресенье")
 {obj.res.value="В воскресенье будет ясно, температура воздуха
  поднимется до +30 градусов";}
else
 obj.res.value="Укажите правильно день недели";
}
```



Негізінен барлығы жұмыс істейді, бірақ жауап кодында функциялар тым ұзын. *switch* таңдау операторы оны қысқартуға пайдаланылады. Оның синтаксисі келесі:

```
switch (B)
{
case L1: S1;
case L2: S2;
... ... ...
case Ln: Sn;
default:S
}
```

мұндағы B - өрнек, L1, L2, ..., LN - литералы және S1, S2, ..., Sn - операторлар.

Бұл былай жұмыс істейді: B өрнектің мәні есептеледі. Егер B өрнекке L1 тең болса, онда S1 орындалады. Егер B өрнекке L2 тең болып

табылса, онда S2 операторы орындалады, және т.с.с. Егер B бір мәнгеде тең емес болса, онда S оператор орындалады.

switch операторы көмегімен соңғы функцияның кодын жазайық.

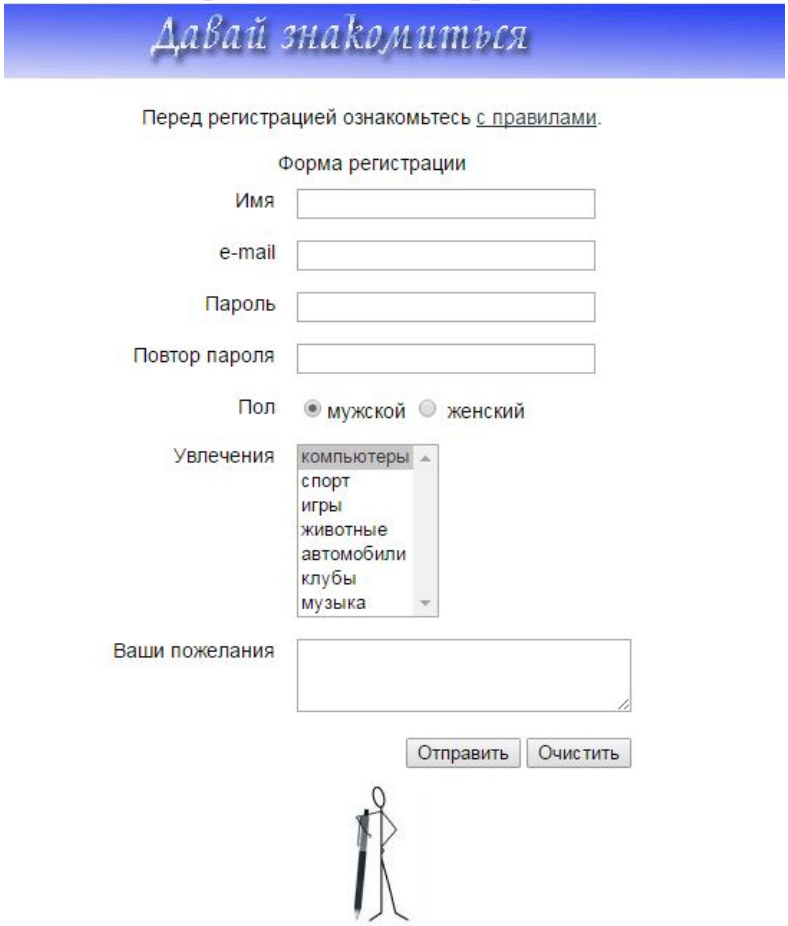
```
function Vday1(obj){
  var a=obj.zn.value;
  var s;
  switch (a)
  {
   case 'понедельник': s="В понедельник будет ветрено, температура
     воздуха поднимется до +20 градусов"; break;
   case 'вторник': s="Во вторник будет солнечно, температура
     воздуха поднимется до +25 градусов"; break;
   case 'среда': s="В среду будет прохладно, температура воздуха
     опустится до +17 градусов"; break;
   case 'четверг': s="В четверг будет пасмурно, температура
     воздуха опустится до +10 градусов"; break;
   case 'пятница': s="В пятницу будет облачно, температура воздуха
     поднимется до +15 градусов"; break;
   case 'суббота': s="В субботу будет ветрено, температура воздуха
     поднимется до +27 градусов"; break;
   case 'воскресенье': s="В воскресенье будет ясно, температура
     воздуха поднимется до +30 градусов"; break;
   default: s='Укажите правильно день недели'
  }
  obj.res.value=s;
}
```

Нәтижесі, код әлдеқайда қысқа, оқылмайды. *break* операторы қажетті нұсқаны ауыстырып-қосу орындағаннан кейін жұмысты аяқтаудықамтамасыз ету үшін қолданылады. Оны егер түсіру керекболса ештеңе өзгермейді, жай ғана онымен сценарий тезірек жұмыс істейді, бірақ осы мысалда бұл білінбейді.

ТАҚЫРЫП 9. WEB-ПАРАҚТАРДЫҢ ОБЪЕКТІЛЕРІН БАСҚАРУ.

3 –сабақта бұл тақырыпты зерттедік. Қазір бұл туралы егжей-тегжейлі қарастырамыз. JavaScript тілінде web-парақта барлық элементтер иерархиялық құрылыммен орнатылады. Әрбір элемент объекта түрінде ұсынылады. Және әрбір осындай обьекттің белгілі бір қасиеттері мен әдістері болуы мүмкін.

Мәселен, осындай web-парақ болсын (4-сурет):



4-сурет. Web -парақ

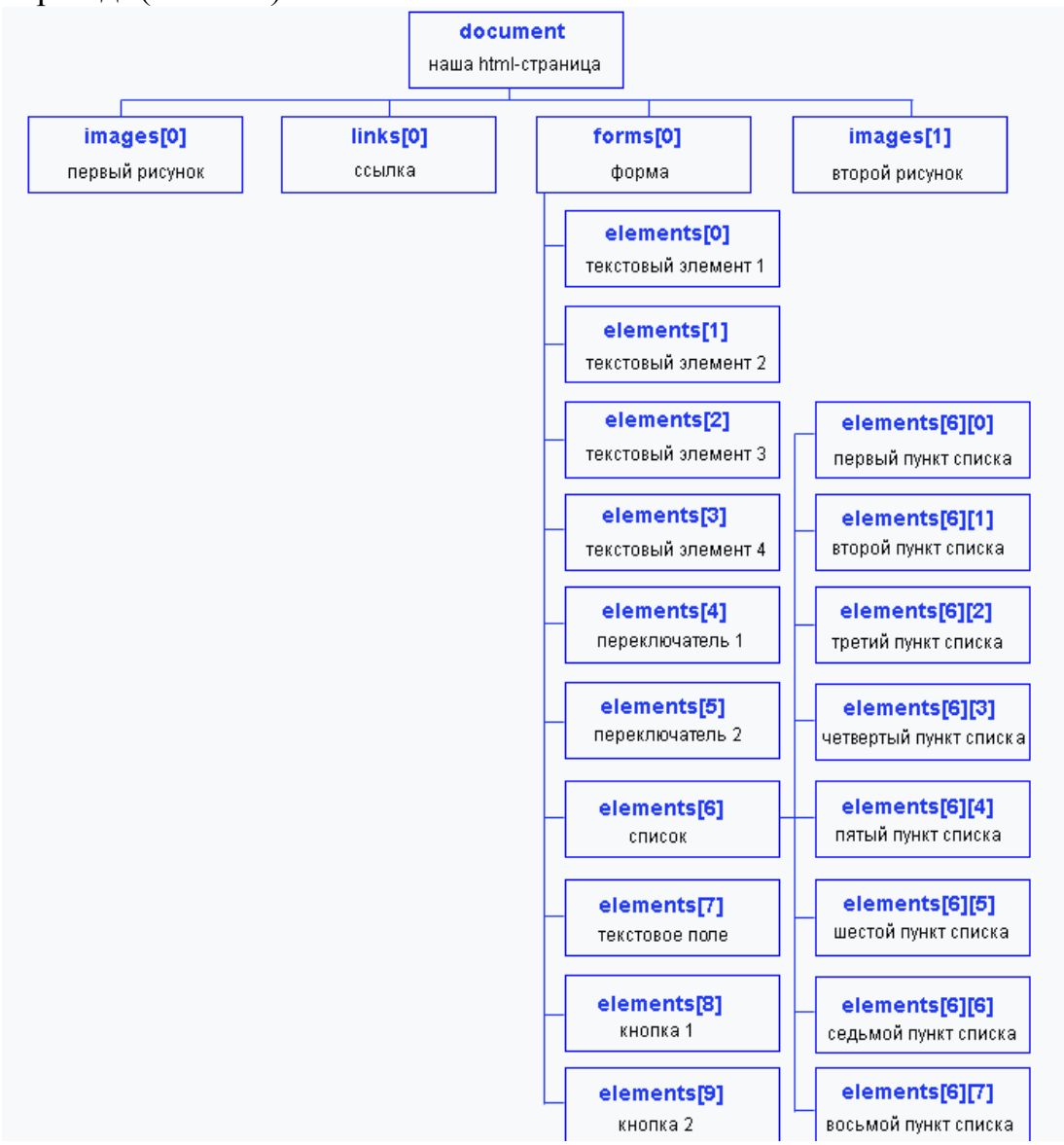
Оның html-коды:

```
<html>
<head>
<title>javascript if</title>
<link rel="stylesheet" type="text/css" href="style1.css">
<script type="text/javascript" src="script.js"></script>
</head>
<body>
<img src="images/pict1.gif"><br><br>
<center>
Перед регистрацией ознакомьтесь <a href="#">с правилами
</a>.<br><br>
```

```
<form name="forma1">
<table  border="0" cellspacing="5" cellpadding="5">
<caption>Форма регистрации</caption>
<tr><td align="right" valign="top">Имя</td>
<td><input type="text" name="name" size="25"></td>
</tr>
<tr><td align="right" valign="top">e-mail</td>
<td><input type="text" name="e-mail" size="25"></td>
</tr>
<tr><td align="right" valign="top" >Пароль</td>
<td><input type="password" name="password" size="25"></td>
</tr>
<tr><td align="right" valign="top" >Повтор пароля</td>
<td><input type="password" name="password1" size="25"></td>
</tr>
<tr><td align="right" valign="top" >Пол</td>
<td><input type="radio" name="sex" value="man" checked>мужской
<input type="radio" name="sex" value="woman"> женский
</td>
</tr>
<tr><td align="right" valign="top">Увлечения</td>
<td><select name="hobby" size="7" multiple>
<option selected value="1">компьютеры
<option value="2">спорт
<option value="3">игры
<option value="4">животные
<option value="5">автомобили
<option value="6">клубы
<option value="7">музыка
</select>
</td>
</tr>
<tr><td align="right" valign="top">Ваши пожелания</td>
<td><textarea cols="30" rows="3" wrap="physical"></textarea>
</td>
</tr>
<tr><td align="right" colspan="2">
<input type="submit" name="submit" value="Отправить">
<input type="reset" name="reset" value="Очистить">
</td>
</tr>
</table>
</form>
```

<img src="images/pict2.gif">
* *
</body>
</html>

Бұл html-парағымен құрылатын объектілердің иерархиясы, былай көрінеді (2-схема):



2-схема. Объектілер иерархиясы

Осы құрылымның әрбір объектісінің өз аты мен индексі бар. Индекс объектінің ережесімен парақта анықталады (жоғарыдан төмен қарай). Сондықтан парақтың жоғарында орналасқан сурет төменгі нөмірді иеленеді. Нөмірлеу 0-ден басталады. *forms* объекттің қасиеті бар - массив – *elements*, форма элементтеріне қамтылған сілтемелерді тәртіппен оларды *form*тегімен аударады. Олардың нөмірленуіде0-ден басталады.Форма объектілерінің өзіне бағынышты объектілері болуы мүмкін. Мысалда, тізімдер болып *elements[6]* объектісі саналады, бұл пункттер тізімі, олар өздерінің ішкі нөмірленуін иеленеді.

Объектіге кіруді алу үшін, қажетті элементке дейінгі иерархияның барлық жол шыңдарын көрсету керек. Мәселен, суретке бірінші жол былай көрінеді:

document.images[0]

Формаға жол:

document.forms[0]

Тізімге жол:

document.forms[0].elements[6]

Ал тізімнің үшінші тармағына жол:

document.forms[0].elements[6][2]

Бұл тәсіл әрқашан ыңғайлы емес, өйткені барлық осы индекстерден оңайшатасып кетуге болады. Сондықтан парақтың барлық элементтеріне әдетте аттары беріледі және элементтерге аты-жөндері бойынша қолжетімділікті жүзеге асырады. Мысалы, форма аты *forma1*, ал бірінші мәтіндік өріс аты *name*. Оған қол жеткізуді алу үшін, мынаны жазу керекпіз:

document.forma1.name

Пайдаланушының бұл мәтіндік өріске енгізгенін білу үшін, *value* қасиеті пайдаланылады:

document.forma1.name.value

Мысалда, екі өрістің сәйкес келуін тексеру үшін пароль енгізуді жүзеге асырамыз. Бұл үшін келесі функцияны жазамыз:

```
function prov(){
    var a=document.forma1.elements[2].value;
    var b=document.forma1.elements[3].value;
    if (a==b){
     alert ("Вы зарегистрированы!");
    }
    else
    {
     alert ("Введите правильный пароль.");
    }
   }
```

Бұл функция "Отправить" бастырмасын басқан кезде іске қосылатын болсын:

```
    ... ... ...
<input type="submit" name="submit" value="Отправить"
onclick="prov();">
    ... ... ...
```

Қолданушы қандай жыныстынұсқағанынбілу үшін*checked* ауыстырып қосқышқасиетін пайдаланады. Ал пайдаланушының таңдағанын анықтау үшін пункт тізімінен - selected қасиеті қолданылады.

Айталық, таңдалған жынысы мен қызығушылығына байланысты, пайдаланушыға қандай да бір хабар тағайындалғанын қалаймыз. prov() функциясын жазамыз:

```
function prov(){
    if (document.forma1.elements[4].checked &&
     document.forma1.elements[6][3].selected){
      alert ("Мужчина, который любит животных заслуживает
     уважения.");
    }
    else if (document.forma1.elements[5].checked &&
     document.forma1.elements[6][4].selected)
    {
      alert ("Женщина за рулем всегда вызывает интерес.");
    }
    else
    {
      alert ("Отличное увлечение.");
    }
}
```

ТАҚЫРЫП10. FOR ЖӘНЕ WHILE ЦИКЛДАР.

Циклдер сол кодты көп рет орындауға мүмкіндік береді және олар өте жиі пайдаланылады.

Цикл for

Циклдің ең танымал түрі - for циклі. Оның синтаксисі:

```
for (A; B; I){S}
```

мұндағы,

A - бастапқы көрінісі,

B - шартты жалғастыру. Егер өрнек жалған болса, онда циклдің орындалуы аяқталады,

I - өрнек инкременты,

{} - цикл денесі,

S – операторлар.

Әдеттегідей, мысалда түсінікті болады. Мысалы, сценарий жазу керек, барлық сандарды a-дан b-ға дейін жинау (a және b пайдаланушымен енгізіліп отырады) және алынған нәтижені көрсету.

Сонымен, html-парақта келесі кодты жазамыз:

```
<html>
<head>
<title>javascript for</title>
<link rel="stylesheet" type="text/css" href="style.css">
<script type="text/javascript" src="script.js"></script>
</head>
```

```
<body>
<form name="forma7">
        Вас интересует сумма всех чисел
        от <input type="text" name="a1" size="5" maxlength="5">
        до <input type="text" name="a2" size="5" maxlength="5">
<input type="button" value="Вычислить" onclick="summa(forma7);"><br>
        Сумма равна <input type="text" name="result" size="10"
        maxlength="10">
</form>
</body>
</html>
```

Ол былайша көрінеді:

script.js парағында summa() функциясын жазу қалды:

```
function summa(obj) {
    var summa=0;
    var a1=1*obj.a1.value;
    var a2=1*obj.a2.value;
    for (var i = a1; i <= a2; i++) {
     summa+=i;
    }
    obj.result.value = summa;
   }
```

Мұнда жаңа тек цикл, енді ол қалай жұмыс істейді анықтаймыз:

і циклі параметріне a1мәні беріледі, яғни саны, ол қолданушы көрсеткен мәтіндік өрістегі *OT* және цикл денесі (фигуралы жақшада) бұл мәнмен орындалады, яғни *summa* айнымалысы a1тең мәніне жазылады.

Содан кейін і++өрнегі есептелінеді, яғни і мәні 1-ге ұлғайтылады.

Одан кейін javascript *i <= a2* шартын тексереді, егер ол ақиқат болса, онда цикл денесі қайтадан орындалады, яғни summa айнымалы мәні көбейіп, өз-өзіне плюс 1 қосады.

Осылайша, *a1*-ден *a2*-ге дейін барлық санды қосып болғанша цикл орындалады.

i <= a2 шарт жалғанға айналған кезде, цикл тоқтатады, ал алынған нәтиже (*summa* айнымалының мәні) result өрісіне шығарылады.

Ескеріңіз,*i*айнымалы деп тікелей *var i = a1* цикліндежариялауға болады.

Жалпы, бұны аяқтау болар еді, бірақ тағы бір ескеретін жайт. Дұрыс жұмыс істеуі үшін *a2* циклінің мәні *a1*-ден артық болуының мағызы бар. Бұл шартты функцияға енгізе кету керек:

```
function summa(obj) {
```

```
var summa=0;
var a1=1*obj.a1.value;
var a2=1*obj.a2.value;
if (a2 > a1){
 for (var i = a1; i <= a2; i++) {
   summa+=i;
 }
 obj.result.value = summa;
}
else
 alert("Значение ОТ должно быть меньше значения ДО")
}
```

Цикл while

Танымал циклдің бір түрі - while циклі. Оның синтаксисі:

while (B){S}

мұндағы,

B - жалғастыру шарты. Егер өрнек жалған болса, циклдің орындалуы аяқталады,

{} - цикл денесі,

S - операторлар.

Басқаша айтқанда, S операторларB шарт жалған болғанша орындалатын болады.

Цикл екі түрі, for циклі, while циклі жиі пайдаланылады, сондықтан олармен кейінгі сабақта көп мысалды талқылаймыз (мысалы, 13 сабақта).

ТАҚЫРЫП11. КҮНДЕР, ҰСЫНУЛАР ЖӘНЕ ӨҢДЕУЛЕР.

JavaScript-та күн өткен 1970 жылдың, 1 қаңтарынан миллисекунд санымен анықталады.

Кірістірілген *Date* объектісі күн және уақытпен жұмыс үшін қолданылады. Бұл обектінің қасиеттері жоқ, бірақ бар бірнеше әдістермен мүмкіндік береді, күн мен уақытты белгілеуге және өзгертуге.

Date объектісі *new*операторы және *Date* конструктор көмегімен құрылады. Мысалы:

var myData=new Date();

myData айнымалының мәні ағымдағы күн және уақыт болады:

Tue Mar 14 2017 23:11:53 GMT 0500 (Батыс Азия (қыс))

Date объекті әдістерімен маңызы бар ай, күн, апта, сағат, минут және секундты бөлек алуға болады:

- *getDate* 1-ден 31 диапазонын білдіретін айдың санын қайтарады.
- *getHours* 0 (түн ортасы) 23 диапазонында сағат тәулікті қайтарады
- *getMinutes* 0-ден 59 диапазонда минутты қайтарады.
- *getSeconds* 0-ден 59 диапазонда секундты қайтарады.

Мысалы, сценарий жазсақ, ол ағымдағы уақыттыанықтауға болатын және оны"сс:мм:сс" форматында шығаруға.

html-парақ кодықарапайым болады:

```
<html>
<head>
<title>javascript ∂ama</title>
<link rel="stylesheet" type="text/css" href="style.css">
<script type="text/javascript" src="script.js"></script>
</head>
<body>
<form name="forma8">
<input type="button" value="Время" onClick="nTime(forma8);">
<input type="text" size="10" name="res">
</form>
</body>
</html>
```

Енді *nTime()*функциясының өзін жазамыз:

```
function nTime(obj) {
   var t=new Date();
   var h=t.getHours();
   var m=t.getMinutes();
   var s=t.getSeconds();
   var result=h+":"+m+":"+s;
   obj.res.value=result;
}
```

Өздеріңіз көріп отырғандай, барлығы қарапайым. Алдымен ағымдағы уақытты анықтаймыз, содан кейін әдістердің көмегімен одан бөлек маңызы бар сағат, минут және секундтарды шығарып аламыз.

Мұнда мына *var result=h+":"+m+":"+s*. қатарды түсіндіру керек. Алғаш рет қажеттігімен шығару нәтижесі ретінде маңызы бар айнымалы және қарапайым мәтінмен бетпе-бет келдік. Негізінен күрделі ештеңе жоқ: айнымалы сол күйінде жазылады, мәтін бағамдарға алынады, ал +белгісі конкатенация операциясын жүзеге асырады, яғни олардың бірлестіктері.

Мысалда жетіспеушілік бар, біз үшін уақыт "сс:мм:сс" форматында шығарылады, ал қазір ол "с:м:с"форматында шығарылады. Яғни, таңғы 5-

те, "5:0:0" уақыты бейнеленетін болады, ал "05:00:00" (бұл әдеттегідей). Үй тапсырмасы ретінде көруге болады, бұны түзетуге болады. Мысалы, if оператордың және "0" қатар литералыныңкөмегімен (қарапайым: егер сағат 10-ғакем болса, онда нәтижеде h алдына "0" жазу және барлық айнымалыға солай қою).Ал әзірге Date объектінің әдістерін үйренуді жалғастырамыз:

- getDay - 0 - (жексенбі) дейін 6 (сенбі) бүтін сан ретінде апта күнін қайтарады.
- getMonth - 0 (қаңтар) 11-ге дейін (желтоқсан) бүтін сан ретінде жылдағы айдың нөмірін қайтарады
- getYear - соңғы екі сан түрінде (getFullYear - төрт сан түрінде қайтарады) жылды қайтарады
  *Өкінішке қарай, 2000 жылдан бастап, әр түрлі браузерлерде жылдың бейнеленуіне проблема бар. Сондықтанда шектеулерді қолдану getFullYear әдісі тиімді.

Ағымдағы күнге және шығаруды анықтауға болатын сценарий жазайық, оны форматындағы "ай, жыл саны ".

html-парақтың қарапайым коды:

```
<html>
<head>
<title>javascript ðama</title>
<link rel="stylesheet" type="text/css" href="style.css">
<script type="text/javascript" src="script.js"></script>
</head>
<body>
<form name="forma9">
<input type="button" value="Дата" onClick="tData(forma9);">
<input type="text" size="20" name="res">
</form>
</body>
</html>
```

Енді *tData()* функциясын жазайық:

```
function tData(obj) {
   var s;
   var t=new Date();
   var y=t.getFullYear();
   var d=t.getDate();
   var mon=t.getMonth();
   switch (mon)
   {
    case 0: s="января"; break;
    case 1: s="февраля"; break;
    case 2: s="марта"; break;
```

```
  case 3: s="апреля"; break;
  case 4: s="мае"; break;
  case 5: s="июня"; break;
  case 6: s="июля"; break;
  case 7: s="августа"; break;
  case 8: s="сентября"; break;
  case 9: s="октября"; break;
  case 10: s="ноября"; break;
  case 11: s="декабря"; break;
 }
 var result=d+" "+s+" "+y;
 obj.res.value=result;
 }
```

Бірінші мысалға қарағанда өте ұзынырақ пайда болды, яғни айлар атауын орыс тіліне аударуға тура келеді.

Жоғарыда қаралған әдістер күнді алуға мүмкіндік береді. Егер күнді орналыстыру белгіленген болса, онда келесі әдістерді пайдалану керек:

- setDate - 1-ден 31-ге дейінгі диапазонда айды белгілейді.
- setHours - 0 (түн ортасы) дейін 23 диапазонында ағымдағы уақыт үшін сағатты белгілейді
- setMinutes - 0-ден 59диапазонда минутты белгілейді.
- setSeconds - 0-ден 59 дейінгі диапазонда секундты белгілейді.

- setYear –жылды белгілейді.
- setMonth - 0 (қаңтар) 11-ге дейін (желтоқсан) диапазонда айдың мәнін белгілейді
- setTime - өткен 1 қаңтар 1970 жылдан Date объектісінің санын белгілейді миллисекундты қайтарады

Сонымен, 2010 жылғы 06 желтоқсан күнін қою қызметі келесі код:...

```
var t=new Date();
var y=t.setYear(2010);
var d=t.setDate(6);
var mon=t.setMonth(11);
...
```

Тікелей құруда күнді көрсетіп қоюға болады, өлшемдер ретінде мына форматты "ай, күн, жыл сағат:минут:секунд":

var t=new Date("Feb,10,1975 17:45:10");

Сағат, минут және секунд мәнін шығару керек (олар нөлге тең болады):

var t=new Date("Feb,10,1975");

Бұл күндерді сандардың көмегімен орнатуға болады, үтір арқылы ұстанымммен жылы, айы, күні, сағаты, минут, секунд:

var t=new Date(75, 1, 10, 17, 45, 10);

Немесе, сағаттар, минуттар және секундтарды шығарып (олар нөлге тең болады):

var t=new Date(75, 1, 10);

* Бұл нұсқаға шектеулер қойылған

ТАҚЫРЫП 12. МАССИВТЕР.

Массив элементтер жиынтығын білдіреді, оларға қол жеткізу индекс бойынша жүзеге асырылады. Массив*new* операторыныңжәне массив конструкторы array функциясыкөмегімен құрылады .

Пайдаланушылар атауы қажет массив жасау. Оны былайша жасауға болады:

var users = new Array("Artem", "Irina", "Sergey", "Boris");

Бұл өрнек 4 элементті (пайдаланушылар) массив жасайды. Массивтің барлық элементтері, нөлден бастап нөмірленген. Массивтің мәніналу үшін массив аты жәнетік жақшаның ішінде элементтің реттік нөмірі (индексі)қажет. Массивтің бірінші элементіне қол жеткізу үшін жазуға болады:

users[0]

Массивтің мәнін бірден беру маңызды емес. Мысалы, мынадай конструкцияны пайдалануға болады:

var users = new Array(4);

Бұл өрнек 4 элементтен массив жасайды, бірақ элементтердің мәнін меншіктеу операторының көмегімен кейінірек көрсетуге болады:

users[0] = "Artem";
users[1] = "Irina";
users[2] = "Sergey";
users[3] = "Boris";

және, ақыр соңында, конструкторды параметрсіз пайдалануға болады:

var users = new Array();

Бұл жағдайда құрылатын *users* айнымалы массив болады, оның элементтерінің мөлшері пайда болуына қарай автоматты түрде анықталады.

Массив ұзындығын анықтау үшін (яғни онда қанша элемент орналасқан) length қасиеті пайдаланылады. Мысалы, массивтің соңғы элементіне (егер онда қанша элемент бар екенін білмесек) төмендегідей қол жеткізуге болады:

users[users.length-1];

Яғни, алдымен массивтің ұзындығы (users.length) анықталады, содан кейін, оның элементтерінің нөмірленуі нөлден басталатынын ескере отырып, ұзындықтан 1 шегеріледі және алынған мән массив индексі ретінде пайдаланылады. Мысал қарастырайық: айталық, парақты жүктеу кезінде қолданушы бүгін қандай апта күні екенін көрсін.

html-парақ коды мынадай болады:

```
<html>
<head>
<title>javascript массивы</title>
<link rel="stylesheet" type="text/css" href="style.css">
<script type="text/javascript" src="script.js"></script>
</head>
<body onload="showDay();">
</body>
</html>
```

showDay() функциясын жазамыз:

```
function showDay() {
   var nDays=new Array("воскресенье", "понедельник", "вторник",
"среда", "четверг", "пятница","суббота");
   var now=new Date();
   var day=now.getDay();
   var iDay=nDays[day];
   var str="Сегодня - " + iDay;
   document.write(str);
}
```

Толығырақ қарастырайық Алдымен, nDays массивті жеті элементтен құраймыз (жексенбіге нөлге сәйкес келеді):

```
var nDays=new Array("воскресенье", "понедельник", "вторник",
"среда", "четверг", "пятница","суббота");
```

Содан кейін, ағымдағы күннен ағымдағы апта күнін аламыз (оның сандық баламасы):

```
var now=new Date();
var day=now.getDay();
```

және оны массив элементі үшін индекс ретінде пайдаланамыз:

```
var iDay=nDays[day];
```

Соңғы екі қатар нәтижені қалыптастырады және жариялайды:

```
var str="Сегодня - " + iDay;
document.write(str);
```

Сценарий жұмысын әрекетте көріңіз және көз жеткізіңіз, парақты жүктеу кезінде, "Бүгін - жексенбі" түріндегі сөз тіркесі пайда болады. Әріқарай *concat()* әдісі көмегімен екі массивті біреуге біріктіруге болады. *a* және *b* екі массив болсын және оларды *c* –ға біріктіреміз:

```
var a=new Array(1, 2, 3);
var b=new Array(4, 5, 6);
var c=a.concat(b);
```

Алты элементтен *c* массивін шығуда аламыз: 1, 2, 3, 4, 5, 6.

sort() әдісінің көмегімен массивтерді сұрыптауға болады, шындығы сол лексикографикалық тәртіптеғана (яғни, қатарлар). Мысалы, егер sort() әдісін массив атауларына ауыстыру:

```
var users = new Array("Artem", "Irina", "Sergey", "Boris");
var c=users.sort();
```

онда шығуда *c* массивін аламыз: *Artem, Boris, Irina, Sergey.*

Ал, егер *sort()* әдісін сандар массивіне қолданансақ:

```
var n = new Array(11, 12, 10, 107, 3, 20, 25, 101, 14, 34, 44, 5, 4);
var c=n.sort();
```

онда шығуда келесі массивті аламыз: 10, 101, 107, 11, 12, 14, 20, 25, 3, 34, 4, 44, 5. Яғни, элементтері қатар ретінде салыстырылады, сан ретінде емес. Сондықтан sort() әдісін қолдануда абайлауыңыз керек.

Енді мысалды талқылайық, массивте берілген элементті қалай іздеу керек. Айталық, 6 атаудан тұратын массив бар, және онда белгілі бір атау мекенін білгіміз келеді.

html- парақ коды қарапайым болады:

```
<html>
<head>
<title>javascript массивы</title>
<link rel="stylesheet" type="text/css" href="style.css">
<script type="text/javascript" src="script.js"></script>
</head>
<body>
<form name="forma10">
   Введите имя для проверки:
<input type="text" size="20" name="name">
<input type="button" value="Проверить"
onClick="proverka(forma10);">
</form>
</body>
</html>
```

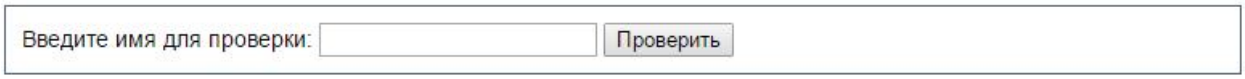
Енді *script.js* парағына *proverka()* функциясының өзін жазамыз:

```
function proverka(obj) {
   var k;
   var users = new Array("Artem", "Irina", "Sergey", "Boris", "Olga",
"Viktor");
   var n=obj.name.value;
   for (var i=0; i<=users.length-1; i++){
     if (users[i]==n)
     {k=0; break}
   }
   if (k==0){alert("Это имя присутствует в массиве.")}
```

else {alert("Такого имени в массиве нет.")}

}

Бұл жерде біз сызықтық іздеу деп аталатын алгоритмді пайдаландық, оған сәйкес массивтің барлық элементтері кезекпен үлгімен*users* салыстырылады (*n*, ол қолданушы енгізген атауға байланысты). Егер кезекті элементүлгімен сәйкес келсе, онда міндет шешілді (цикл үзіледі). Егер ізделінді атау массивте жоқ болса, онда бұл туралы барлық массив элементтері қарап шыққаннан кейін ғана білеміз.



ТАҚЫРЫП 13. ҚАТАРЛАР

Біз бірнеше рет, бір немесе қос тырнақшаға қоса алынып, беріліп отырған параметрмен, таңбалар тізбегін білдіретін қатарлық литералыны пайдаландық. Барлық қатарлар String типті объектілер болып табылады және кейбір іс-әрекеттерді олармен жасауға болады мысалы, қазірдің өзінде конкатенация операциясымен таныспыз, яғниқатарларды біріктіру. Бұл сабақта басқа мүмкіндігі қатарлармен жұмыс істеуді көрсетеміз.

Бастау үшін кейбір ұғымдардыанықтаймыз:

• Алфавит – түпкі көптеген таңбалар.

• Қатар – кейбір альфавит символдарының ақырғы жүйелілігі.

• Бос қатар - құрамында бірде-бір символ жоқ қатар.

Қатар объектісін жасау үшін newString конструкторын пайдалануға болады. Мысалы:

var s = newString("Итого:");

Егер объектінің жалғыз қасиеті String - length қолдансақ, онда қатарлар ұзындығын білеміз.Мысал үшін, s.length орындау нәтижесі 6 саны болады.

Қатарға әсер ететін әдістері қарастырайық:

- charAt(n) – n параметр анықтайтын символ ұстанымын қайтарады. мысалда, s.charAt(0) орындау нәтижесі "И" әрпі.
- substr(n1,n2) – қатардың төменгі қатарларын қайтарады, мұндағы, n1 - бірінші символдың төменгі қатардағы орыны, ал n2 – қатардағы символдар саны.

  Мысал үшін, орындау нәтижесі s.substr(0, 4) " Итог " қатары. Сценарий жазамыз, ол берілген сөздің белгілі бір мәтінде қанша рет кездесетін анықтайтын болады.

  html-парағы коды мынадай болады:

  <html>

  <head>

  <title>javascript строки</title>

  <link rel="stylesheet" type="text/css" href="style.css">

  <script type="text/javascript" src="script.js"></script>

```
</head>
<body>
<form name="forma11">
Введитетекст:<br>
<textarea name="textin" rows="4" cols="20"></textarea><hr>
Введитеслово: <input type="text" size="10" name="slovo"><hr>
<input type="button" value="Определить"
onClick="numword(forma11);">
<input type="reset" value="Отменить"><hr>
Количествословвтексте: <input type="text" size="10" name="res">
</form>
</body>
</html>
```

Енді script.js парғында numword () функциясының өзін жазамыз:

```
function numword(obj) {
  var t=obj.textin.value;
  var s=obj.slovo.value;
  var m=s.length;
  var res=0;
  var i=0;
  while (i < t.length-1)
  {var ch=t.substr(i,m)
   if (ch==s){
   res+=1;
   i=i+m
   }
   else
   i++
  }
  obj.res.value=res
}
```

Нәтижелерді қараймыз, содан кейін жазбаларды саралаймыз:

Сонымен, функцияларға не жазғанымызды көрейік.Алдымен екі қатарлық айнымалыны параметрмен анықтадық, біреуі пайдаланушы енгізілген мәтін болып табылады, екіншісі - сөз.

```
var t=obj.textin.value;
var s=obj.slovo.value;
```

Содан кейін ізделінді сөздердіңұзындығын анықтадық және нәтижесінде айнымалыны нөлге айналдырдық:

```
var m=s.length;
var res=0;
```

Бұдан әрі *while* цикл көмегімен, келесілерді көрсетеміз: әзірше, *i* циклінің параметрі мәтін ұзындығынан аз немесе тең (бір кеміту, яғни таңбаны нөмірлеу нөлден басталады) төменгі қатарларды алу, ағымдағы *i*таңбаданбастап және ізделінді сөздер ұзындығыныңұзындығына тең және оны ізделінді сөздің өзімен салыстыру. Егер нәтиже ақиқат болса, онда көшпелі *res*бірлігіне артады, ал цикл өз жұмысынсимволмен*(i=i+m-1)*табылған сөздер үшін жалғастырады. Олай болмаған жағдайда, келесі символ үшін:

```
var i=0;
 while (i<=t.length-1)
{var ch=t.substr(i,m)
  if (ch==s){
  res+=1;
  i=i+m-1
  }
  else
  i++
 }
```

Соңында, нәтиже:

```
obj.res.value=res
```

Javascript тілінде бірқатар стандартты функциялардың қатарлармен жұмысы анықталған:

- Number(s) - қатар параметрін *s* санына түрлендіреді.
- String(n) - *n* санын қатарға түрлендіреді.
- isNaN(s) - параметр *s* сандыма, сонытексереді. Егер параметр *s*сан болып табылмаса, онда true (ақиқат) мәні қайтарылады, кері жағдайда - false (жалған) болады.

Есіңізде болсын, алдыңғы сабақтарда біз қолданушымен енгізілген санды алдық, және 1 көбейттік. Осылайша, оларды қатарларда санға аударды қ. Мысалы:

```
var a1=1*obj.a1.value;
```

Number(s) стандартты функциясын пайдалана отырып жаза аламыз:

```
var a=obj.a1.value;
var a1=Number(a);
```

Басқа да стандартты функциялар бар, бірақ олардың сізге қажеті болмайды, сондықтан оларды қарастырмайтын боламыз.

ТАҚЫРЫП14. ТҰРАҚТЫ ӨРНЕКТЕР.

Тұрақты өрнек - бұл нұсқаулық, арнайы тілде (RegExp) әзірленген іздеу қатарының үлгімен"ұқсастығы бар" заң табысын сипаттайтын.

Бұл не үшін керек? Мысалы:

- мәтінде іздеулерді ұйымдастыру үшін
- мәліметтерді басқа бір бөліктермен ауыстыру үшін.
- пайдаланушының енгізу дұрыстығын тексеру үшін (бәлкім, сіз бірнеше рет электрондық пошта адресінің қандай да бір нысаннан "Некорректный e-mail" типті қателік алған жағдай жасадыңыз).

Тұрақты өрнекті құрудың екі тәсілі бар, осы сабақта литералды аннотация құруды қарастырайық:

var p=/pattern/flags;

мұндағы:

pattern - үлгі, тұрақты өрнектің негізі болып табылатын, іздеу шартына сәйкес айқындайтын қатарлар. Литерал және метасимволдан тұрады.

flags - белгіше (түрлендіргіштер), салыстыру үлгісіне қосымша параметрлер қояды.

Мысал:

var par=/[0-9a-z]+/i;

Мұнда *[0-9a-z]+* - үлгі, сөзбе-сөз мынаныбілдіреді "1 және одан да көп кез келген цифр мен әріптердің сандары".

i - белгіше, бұл таңба регистрінің маңызды емес екенін көрсетеді. Түсінікті болу үшін, ол туралы мысал қарастырайық. Айталық, қолданушы өз e-mail мен құпия сөзін енгізетін формасы бар. Біз "Зарегистрировать" батырмасын басқан кезде дұрыстығын тексеруді енгізу жүзеге асырылады. html-парақкоды мынадай болады:

```
<html>
<head>
<title>javascript регулярныевыражения</title>
<link rel="stylesheet" type="text/css" href="style.css">
<script type="text/javascript" src="script.js"></script>
</head>
<body>
<h2>Форма регистрации</h2>
<form name="forma12">
<label>E-mail:</label><br>
<input type="text"  name="mail"><br>
<label>Password:</label><br>
```

```
<input type="password" name="pas"><br>
<input type="button" value="Зарегистрировать"
onclick="prov_adress(forma12);">
</form>
</body>
</html>
```

Сонымен, *prov_adress()* функциясы не жасауға тиіс? Бастау үшін екі айнымалы керек, оларда қолданушымен енгізілген кез келген мәнді орналастырамыз:

```
function prov_adress(obj) {
  var adr=obj.mail.value;
  var par=obj.pas.value;
}
```

Енді бізге үлгілерді (өрнектер) қою керек, онда қолданушы енгізгендерді салыстыратын боламыз:

```
function prov_adress(obj) {
  var adr=obj.mail.value;
  var par=obj.pas.value;
  var adr_pattern=/[0-9a-z_]+@[0-9a-z_]+\.[a-z]{2,5}/i;
  var par_pattern=/[0-9a-z]+/i;
}
```

Енді салыстыру үлгісімен тексеруді жүзеге асырамыз. Бұл үшін RegExp объектінің test әдісін қолданамыз:

```
function prov_adress(obj) {
  var adr=obj.mail.value;
  var par=obj.pas.value;
  var adr_pattern=/[0-9a-z_]+@[0-9a-z_]+\.[a-z]{2,5}/i;
  var par_pattern=/[0-9a-z]+/i;
  var prov=adr_pattern.test(adr);
  var prov1=par_pattern.test(par);
}
```

adr_pattern.test(adr) қатары келесіні білдіреді: adr қатарындағы реттілікті тексеру, adr_pattern тұрақты салыстыруды білдіруге байланысты. test әдісі логикалық мәнді қайтарады (true немесе false).

Бізге функцияларда, табысты (немесе неуспешной) тексеру жағдайын көрсету ғана қалды:

```
function prov_adress(obj) {
  var adr=obj.mail.value;
  var par=obj.pas.value;
  var adr_pattern=/[0-9a-z_]+@[0-9a-z_]+\.[a-z]{2,5}/i;
  var par_pattern=/[0-9a-z]+/i;
  var prov=adr_pattern.test(adr);
  var prov1=par_pattern.test(par);
```

```
 if (prov==true &&  prov1==true) {
alert("Вы зарегистрированы!");
 }
 else {
     alert("Введенные данные некорректны!");
 }}
```

Дайын, бірақ сценаридің жұмысын тексеруден бұрын, тұрақты өрнектер неден тұратынын көреміз:

Тұрақты өрнекті білдіруде пароль үшін - */[0-9a-z]+/i* аламыз:

- */[0-9a-z]+/* - үлгісі, онда:
  - *0-9* - кез келген цифр.
  - *a-z* - кез-келген қатар әрітері *a*-дан *z*-ке дейін.
  - *[]* - квадратты жақша білдіреді, үлгідеолардың кез келген тізімделген литералдары қатыса алады (цифрлар мен әріптер)
  - +– бұл бөлігі қалыпты (т. е., онда тік жақша ішінде көрсетіледі) бір және одан да көп рет қайталануы мүмкіндігін көрсетеді.
- *i* - бұл таңба регистрінің маңызды емес екенін көрсетеді.

Басқаша айтқанда, пароль 1 және одан да көп рет қолданылатын кез келген цифрлар мен әріптер саны болуы мүмкін (яғни, ол бір цифр, бір әріп, көптеген цифрлар, көптеген әріптер, цифрлар және әріптерден тұруы мүмкін). Мысалы, егер қолданушы пароль өрісінде *"2", "a3b"* немесе *"leopard"* енгізсе, онда мұндай құпия сөз сыпайы болып есептеледі. Ал егер, ол *"ab&s"* немесе *"24?"* берсе, мұндай құпия сөз сыпайы болып саналмайды, яғни оның құрамында спецсимволдар бар, ал оларға жүйелі түрде рұқсат етпедік. Тұрақты өрнектерді не үшін пайдалануға болатыны енді түсінікті болды, оларды қалай жасау керек принциптерін білу қалды.

Негізінен, тұрақты өрнекті құру міндеті, оның үлгісін жасау болып табылады. Ал үлгі, естеріңізде болса, литерал және метасимволдардан тұруы мүмкін.

Ең қарапайым – литералдан бастайық:

- ондай таңбалардың әрқайсысы өзін-өзі ұсынады. Мысалы */abc/* - осындай үлгіге тек қана "*abc*" қатары салыстырмалы.
- *a-z* - барлық қатар әріптері *a*-дан *z*-ке дейін. Мысалы, */a-z/* - осындай үлгіге 26-қатар салыстырылады: *"a", "b", "c" - ... "z"*
- *A-Z* - барлық бас әріптері *A*- дан *Z*-ке дейін.
- *0-9*–барлық цифрлар.

Егер цифрдың немесе әріптердің бірнеше болуын қаласақ, онда басқарушылар рәміздерді пайдалануға тура келеді:

- * - бұл символ (немесе үлгіні, егер ол мәміле " квадрат жақшалар) 0 және одан да көп рет қайталануы мүмкін екенін көрсетеді. Мысалы, /ab*c/ - білдіреді, бұл қатардың *a* символдан басталатынын, содан

кейін мүмкін кез бірнеше*b* символдары, содан кейін *c*символы керек. Мысал , яғни болуы мүмкін, мұндай қатарлар: *"ac", "abc", "abbbbbbc"* және т. б.

- + - бұл символ (немесе үлгіні, егер ол мәміле " квадрат жақшалар) 1 және одан да көп рет қайталануы мүмкін екенін көрсетеді. Мысалы, */ab+c/* - білдіреді, бұл қатардың *a* символ басталып, содан кейін b бірнешесимволдарыболу мүмкіндігін (бірақ 1-ден кем болса), кейін *c* символыкерек. мысалы, яғни мұндай қатарлар болуы мүмкін: *"abc", "abbbbbbc"* және т. б.
- . - ол бұл жерде жаңа қатардың символынан басқа, кез келген жеке символ болуы мүмкін екенін көрсетеді. Мысалы, шаблон /ab.c/ мұндай салыстырмалы қатарлар: *"ab6c", "abxc", "ab=c"* және т. б.
- ? - бұл символ (немесе үлгінің бөлігі, егер ол квадрат жақшаларға алынса) 0 немесе 1 рет қайталануы мүмкін екенін көрсетеді. Мысалы, */ab?c/* - бұл қатардың *a* символдан басталатыны, содан кейін бір b символының болу немесе болмау мүмкіндігі, одан кейін *c*символы болатынын білдіреді.Бұл, осындай қатар болуы мүмкін: *"ac", "abc"*.
- {n} - бұл символ (немесе үлгі бөлігі, егер ол квадрат жақшалармен аяқталса) осыдан n рет қайталануы мүмкін екенін көрсетеді. Мысалы, */ab{3}c/* - білдіреді, бұл қатар *a* символдан басталады, содан кейін 3 *b* символы болады, содан кейін *c* символы керек. Яғни, бұл жолы *"abbbc"*.
- {n,} - бұл символ (немесе үлгіні, егер ол мәміле " квадрат жақшалар) n және одан көп рет қайталануы мүмкін екенін көрсетеді. Мысалы, */ab{3,}c/* - білдіреді, бұл қатар*a* символдан басталады, содан кейін 3 немесе одан да көп таңбадан b болады, одан кейін *c* символы керек. Бұл осындай болуы мүмкін: *"abbbc", "abbbbbbbc"* және т. б.
- {n, m} - бұл символ (немесе үлгі бөлігі, егер ол квадрат жақшаларда болса) n дейін m рет қайталануы мүмкін екенін көрсетеді. Мысалы, */ab{1,3}c/* - бұл қатар*a* символдан басталатыны, содан кейін 1-ден 3-дейінb таңбадан жүріп, кейін c символыкерек екенін білдіреді. Бұлосындай болуы мүмкін: *"abc", "abbc", "abbbc"*.
- [] - осындай үлгінібелгілі бір жақшадағы көбтікке тиесілі, кез келген жеке символменсалыстыруға болады. Көптік ауыстыру арқылы қойылады немесе ауқымы көрсетіледі. Мысалы, шаблон /[abc]/ мыналармен салыстырылуы мүмкін: *"a", "b", "c"*.
- [^] - осындай үлгіні белгілі бір жақшадағы көптікке тиесілі емес, кез келген жеке символмен салыстыруға болады. Мысалы, шаблон */[^abc]/* мыналармен салыстырылуы мүмкін: *"f", "x", "Z",* бірақ мыналармен салыстырылуы мүмкін емес: *"a", "b", "c"*.
- ^ - бұл символдар қатардыңбасынан салыстырмалы екенін көрсетеді. Мысалы, шаблон */^abc/*мыналармен салыстырылуы мүмкін: *"abcd",*

"abcfh", бірақ мыналармен салыстырылуы мүмкін емес: *"dabc", "cbabc"* және т. б.

- $ - бұл символдар қатар соңымен салыстырмалы екенін көрсетеді. Мысалы, шаблон */abc$/* мыналармен салыстырылуы мүмкін *"dabc", "fhabc",* бірақ мыналармен салыстырылуы мүмкін емес: *"abcd", "abccb"* және т. б.
- | - бірнеше балама үлгілерді көрсетеді. Мысалы, шаблон */ab|c/* қатарлар салыстырылуы мүмкін: *"ab"* және *"c"*.
- \ - экрандау үшін арнайы таңбалар қызмет етеді, яғни кері слэш алдындағы символға, ол арнайы ретінде қызмет көрсетуі керек екені түсіндіріледі. Мысалы:
  - \d - кез-келген 0-ден 9-ға дейінгі цифр сәйкес келеді.
  - \D – цифрдан басқасының барлығы сәйкес келеді.
  - \ ' s - бос орын сәйкес келеді.
  - \ ' S - бос орыннан басқасының барлығы сәйкес келеді.
  - \w - әріп, цифр немесе астыңғы сызық сәйкес келеді.
  - \W - әріп, цифр немесе астыңғы сызықтан басқасының барлығы сәйкес келеді

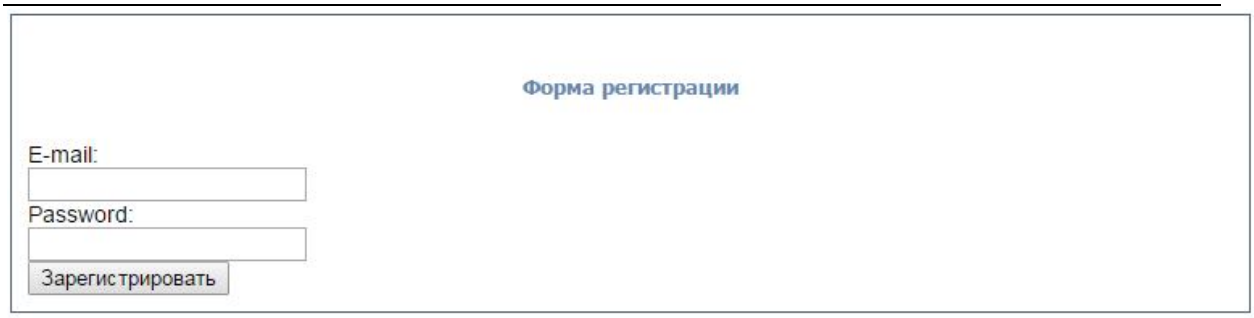
Мысалы, шаблон */x\d\d/* мынаған сәйкес болады: *"x01", "x25"* және т.б., бірақ мынаған сәйкес болмайды: *"A15", "x0A"...*

Сондай-ақ, кері слэш арнайы таңбаны литералды жасау үшін пайдаланылады. Мысалы, егер қатарды табу *"a*b"* керек болса, онда келесі үлгіні */a*b/* береміз.

Жоғарыда аталған литерал және метасимволды пайдалана отырып, қандайда үлгілерді жасауға болады (тұрақты өрнектерді есептеңіз). мысалы, e-mail үлгіні үшін мысалда не жаздық, көрейік:

var adr_pattern=/[0-9a-z_]+@[0-9a-z_]+\.[a-z]{2,5}/i;

Сонымен, электрондық почта мекенжайында 1 және одан да көп рет цифрлар, әріптер және белгілерді сызуды көрсеттік, содан кейін @ символы, содан кейін қайтадан 1 және одан да көп рет цифрлар, әріптер және белгілер сызуы, содан кейін нүктеден кейін 2-ден 5 рет әріптер символы жүреді. Шамамен осындай түрлер электрондық пошталар адресі болады.



"serega@mail" түрінде e-mail енгізіп көріңіз және қандай болатынын қараңыз. Айта кету керек, үлгілердіжасау ол өзінше бір өнер. Оның үстіне,

бір тұрақты өрнекті әр түрлі жазуға болады. Мысалы, */[0-9a-z_]/* - бұл да */\w/.* секілді.

ТАҚЫРЫП15. ТЕРЕЗЕЛЕРМЕН ЖҰМЫС.

Бізде автомобильдердің атаулары орналасқан батырмалары бар және түймешігін басу батырмасы бойынша сақтандыру ережесі құжатында бұл автокөліктің картинкасының терезесі бар парақ болсын.

html-парақ коды мынадай болады:

```
<html>
<head>
<title>javascript окно</title>
<link rel="stylesheet" type="text/css" href="style.css">
<script type="text/javascript" src="script.js"></script>
</head>
<body>
<form name="forma13">
<input type="button" value="ford" onClick="open_ford()">
<input type="button" value="mazda" onClick="open_mazda()">
<input type="button" value="volvo" onClick="open_volvo()">
</form>
</body>
</html>
```

Енді автомобильдер бейнелерімен (ford.html, mazda.html, volvo.html) жеке және жаңа терезелерге белгіленетін үш парақ жасау керек.

ford.html парақ коды мынадай болады:

```
<html>
<head>
<title>javascript окно</title>
<link rel="stylesheet" type="text/css" href="style.css">
<script type="text/javascript" src="script.js"></script>
</head>
<body>
<img src="images/ford.jpg" width="300">
<input type="button" value="Закрыть" onClick="close_pict()">
</body>
</html>
```

Екі басқа ұқсас парақты өзіңізжасаңыз (images папкаға тиісті суреттерді орналастыруды ұмытпаңыз).

Бізфункцияларда window объектісінің екі әдісін пайдаланатын боламыз - *open (), close()*. Біріншісі жаңа терезе жасайды, екіншісі - оны жабады.

script.js парағында функциялар кодын жазамыз, содан кейін талқылайық.

```
function open_ford() {
  ford=window.open("ford.html", "display_ford",
    "width=400,height=300,status=no,toolbar=no,menubar=no");
}
function open_mazda() {
  mazda=window.open("mazda.html", "display_mazda"
    "width=400,height=300,status=no,toolbar=no,menubar=no");
}
function open_volvo() {
  volvo=window.open("volvo.html", "display_volvo"
    "width=400,height=300,status=no,toolbar=no,menubar=no");
}
function close_pict() {
  window.close();
}
```

Біз енді не жаздық көрейік. Алғашқы үш функцияларда қатарлар түрі болады:

```
=window.open("ford.html", "display_ford",
"width=400,height=300,status=no,toolbar=no,menubar=no");
```

Бұл қатар*window* объектінің *open()*әдісі арқылы жаңа терезе жасайды (мысалы, есіңізде болар, әдістернүктелермен бөлектенеді), және оған ford.html парағын жазады. Бұл әдістің үш параметрі бар, олардың әрқайсысы тырнақшаға алынады:

- бірінші параметр экранда жүктелетін терезенің парағын көрсетеді (мысалы, ford.html),
- екінші параметр ашылатын терезенің атын береді (мысалда display_ford),
- үшінші параметр терезені құру процесін басқару мүмкіндігін береді.

Мүнда терезенің мөлшерін беруге болады, жаңа терезеде статус қатары, құралдар тақтасын немесе мәзір болуын көрсету керек. Біздің мысалда, терезе ені 400 және биіктігі 300 пикселде алдық, статус қатары, құралдар тақтасы және мәзір жоқ. Ал *no*орнына *yes* жазсақ, сонда олартерезеде пайда болар еді.

Осында басқа да бірқатар параметрлерді көрсетуге болады:

- *location* қатарын URL шығарады
- *scrollbars* жылжыту сызығын қосады, егер құжат экранға сыймаса
- *resize* пайдаланушыға терезе мөлшерін өзгертуге мүмкіндік беру

біз қолданған соңғы функцияларда, close ()әдісі ағымдағы терезені жабады.

confirm

Диалогтық терезе көрсетілген хабарламамен және панельдегі "OK" және "Cancel" көрсетеді. Хабар пайдаланушынышешім қабылдауғаынталандыру тиіс. Растау trueәдісінжүктейді, егер қолданушы "OK" таңдаса немесе false егер қолданушы "Cancel"таңдаса.

Синтаксис:

confirm ("сообщение")

Мысал:

Бізде "Открыть окно" батырмасы болсын, шерту арқылы диалогты терезенің пайда болуын қаласақ, мүмкіндік берушілер оның ашылуын қалайтынымызды тағыда тексереді. Терезе тек диалогты терезеде "OK" таңдағанда ғана, терезе бейнелермен ашылады.html-парағының коды:

```
<html>
<head>
<title>javascript окно</title>
<link rel="stylesheet" type="text/css" href="style.css">
<script type="text/javascript" src="script.js"></script>
</head>
<body>
<form name="forma14">
<input type="button" value="Открытьокно" onClick="choiceOf();">
</form>
</body>
</html>
```

script.js парағында choiceOf() функциясының кодын жазамыз:

```
function choiceOf(){
   if (confirm("Вы действительно хотите открыть окно?")) {
     volvo=window.open("volvo.html", "display_volvo",
          "width=400,height=300,status=no,toolbar=no,menubar=no");
    }}
```

prompt

Бұл әдіс пайдаланушының диалогты терезеге енгізуді көрсетеді.

Синтаксис:

prompt(сообщение, [значение по умолчанию])

мұнда, *[]* - бұл параметр міндетті емес екенін білдіреді.

Мысал:

html парағының коды:

```
<html>
<head>
<title>javascript окно</title>
<link rel="stylesheet" type="text/css" href="style.css">
<script type="text/javascript" src="script.js"></script>
</head>
<body>
<form name="forma15">
<input type="button" value="Давайзнакомиться!"
onClick="acquaintance()">
</form>
</body>
</html>
```

Функция коды:

```
function acquaintance(){
  var YouName=prompt("Как тебя зовут?", "Напиши здесь свое
имя.");
  alert("Привет "+YouName);
}
```

setTimeout

Бұл әдіс көрсетілген миллисекунд уақыт аралығыөткен соң орындау.

Синтаксис:

setTimeout (что делать, время в миллисекундах)

Мысал:

"Можно начинать?" батырмасын шерткеннен кейін 3 секундта "Начинайте!" мәтінде хабарлама терезесі пайда болады.

Парақтың html- коды:

```
<html>
<head>
<title>javascript окно</title>
<link rel="stylesheet" type="text/css" href="style.css">
<script type="text/javascript" src="script.js"></script>
</head>
<body>
<form name="forma16">
<input type="button" value="Можноначинать?"
onClick="setTimeout('startMessage()',3000)">
</form>
</body>
</html>
```

startMessage()функциясы 3 секундтан кейін орындалуын қарастырамыз.

```
function startMessage(){
   alert("Начинайте!")
}
```

clearTimeout

Бұл әдіс таймерді белгіленген setTimeout әдісі көмегімен ажыратады. Синтаксис:

clearTimeout (timerID)

мұндағы:

timerID –бірегей таймерді орнату кезінде алынған идентификатор.

Екі "Можно начинать?" және "Отменить вопрос" батырмасын жасайық:

Егер "Можно начинать?" батырмасын шертіп, одан кейін хабарлама терезесінің ашылуын күтпей, "Отменить вопрос" батырмасын шертсек онда хабарлама терезесі тіпті пайда болмайды. Парақтың html-коды:

```
<html>
<head>
<title>javascript окно</title>
<link rel="stylesheet" type="text/css" href="style.css">
<script type="text/javascript" src="script.js"></script>
</head>
<body>
<form name="forma17">
<input type="button" value="Можноначинать?"
onClick="timer1=setTimeout('startMessage()',3000)">
<input type="button" value="Отменитьвопрос"
onClick="clearTimeout(timer1)">
</form>
</body>
</html>
```

1-15 ТАҚЫРЫПТАРЫН ҚОЛДАНЫПJAVASCRIPT-таБАҒДАРЛАМАЛАУ

1 тапсырма:

Оң бүтін сан берілсін (төрт таңбаға дейін):

1) 7-ден кем цифрлары бар мәліметті табу;

2) А цифрларыныңсанда бар-жоғын анықтау?

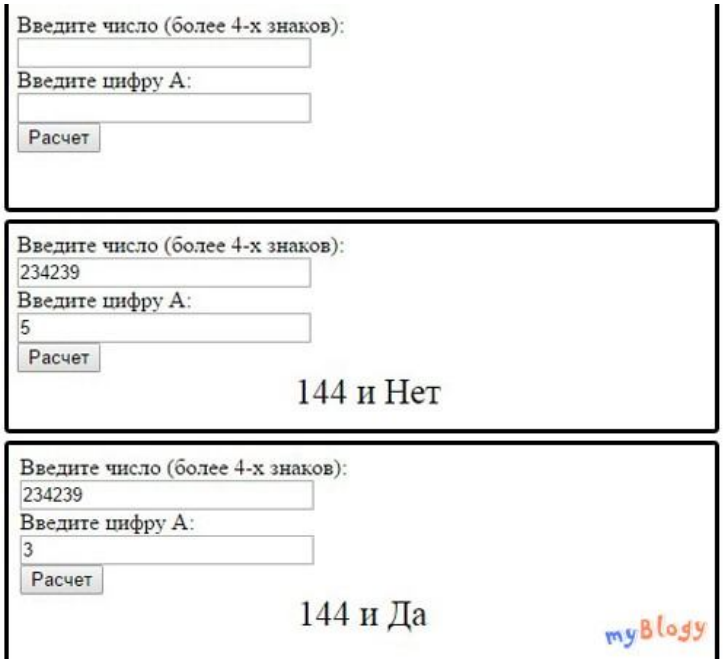
Шешімі:

```
<!DOCTYPE HTML>
```

```
<html>
<head>
<title>Задачипо Javascript</title>
</head>
<body>
<script>
function f_click(Str) {
var div1 = document.getElementById("1").value; //Введенноенашечисло
var div2 = document.getElementById("2").value; //ЦифраА
var b_lines = document.getElementById('lines'); //Блоккудавыведемрезультат
var x=div1.length; //Количество символов
var r="Нет";
var t=1;
for( var i = 0; i < x; ++i ) { //Перебираем каждый символ
if (parseInt(div1.charAt(i))<7){
t=t*parseInt(div1.charAt(i)); //Если меньше 7, умножаем на предыдущее
произведение
}
}
if (div1.indexOf(div2)!=-1){ //Ищем нашу цифру А
r="Да";
}
b_lines.innerHTML=t+' и '+r; //Вывод результата
}
</script>
Введите число (более 4-х знаков): <br />
<input id='1' type="text" size="25" maxlength="30" value="" /><br />
ВведитецифруА: <br />
<input id='2' type="text" size="25" maxlength="30" value="" /><br />
<input type="button" name="enter" value="Расчет" onclick="f_click()" />
<div style="margin: 0 auto;width: 200px; font-size:25px" id="lines"></div>
</body>
</html>
```

Кодта қолданылған функциялар мен әдістер:

- document.getElementById(id) – онымен әрі қарай жұмыс үшін белгіленген ID элементі қайтарылады.
- .value –"value" элементінің мәні қайтарылады.
- .length – массив ұзындығы, яғни қатар ұзындығы.
- parseInt() – нөмірге түрлендіру
- .charAt(i) – і индексі бойынша қатардағы символды шақыру
- .indexOf(x) – қатарда солдан оңға қарай х іздеу. Егер х табылмаса, онда -1 қайтарылады.
- .innerHTML – элемент мазмұнын алады, өзгертеді.



2. тапсырма:

Оң сан (төрт таңбадан астам) берілген:

1) сандардың тақ санын табу;

2) санда 3 –тен кем цифр жоқ па, соны табу керек?

Шешімі:

```
<!DOCTYPE>
<html>
<head>
<title>Задачипо JavaScript</title>
</head>
<body>
<script>
function f_click() {
var div1 = document.getElementById("1").value; //введенноенашечисло
var b_lines = document.getElementById('lines'); //блокдлявыводарезультата
var x=div1.length; //количество цифр в числе
var r="Да";
var t=1;
for( var i = 0; i < x; ++i ) {
if (parseInt(div1.charAt(i))%2==1){ //проверканечетности
t=t*parseInt(div1.charAt(i));}
if (parseInt(div1.charAt(i))%3==0){ //проверка кратности
r="Нет";
}
}
b_lines.innerHTML=t+' и ответ на 2 вопрос - '+r; //вывод результата
```

```
}
</script>
Введите число (более 4-х знаков): <br />
<input id='1' name="login" type="text" size="25" maxlength="30" value=""
/><br />
<input type="button" name="enter" value="Расчет" onclick="f_click()" />
<div style="margin: 0 auto;width: 400px; font-size:25px" id="lines"></div>
</body>
</html>
```

Кодта қолданылған функциялар мен әдістер:

- document.getElementById(id) – онымен әрі қарай жұмыс үшін белгіленген ID элементі қайтарылады.
- .value –"value" элементінің мәні қайтарылады.
- .length – массив ұзындығы, яғни қатар ұзындығы.
- parseInt() – нөмірге түрлендіру
- .charAt(i) – i индексі бойынша қатардағы символды шақыру
- .innerHTML – элемент мазмұнын алады, өзгертеді.

3.Тапсырма:

Қосындысы S-ге тең болатын, 1-ден 200-ге аралықтағы барлық бүтін сандарды табу

Шешімі:

```
<!DOCTYPE>
<html>
<head>
<title>Задачипо JavaScript</title>
</head>
<body>
<script>
function f_click() {
var div1 = document.getElementById("1").value;
var b_lines = document.getElementById('lines');
var x=div1.length;
var r=0;
var t='';
for( var i = 1; i <=200; ++i ) {
for(var j = 0; j <i.toString().length ; ++j ){
r=r+parseInt(i.toString().charAt(j));
}
if (r==parseInt(div1)){
t=t+' '+i;}
r=0;
```

```
}
b_lines.innerHTML=t;
}
</script>
Введитечисло S: <br />
<input id='1' name="login" type="text" size="25" maxlength="30" value=""
/><br />
<input type="button" name="enter" value="Расчет" onclick="f_click()" />
<div style="margin: 0 auto;width: 200px; font-size:25px" id="lines"></div>
</body>
</html>
```

Кодта қолданылған функциялар мен әдістер:

- document.getElementById(id) – онымен әрі қарай жұмыс үшін белгіленген ID элементі қайтарылады.
- .value –"value" элементінің мәні қайтарылады.
- .length – массив ұзындығы, яғни қатар ұзындығы.
- .toString() – қатарға айналдыру.
- parseInt() – нөмірге түрлендіру
- .charAt(i) – i индексі бойынша қатардағы символды шақыру
- .innerHTML – элемент мазмұнын алады, өзгертеді.

4. Тапсырма:

54 санын бөлетін тақ сандарын табу, 3-тен кем

Шешімі:

```
<!DOCTYPE>
<html>
<head>
<title>Задачипо JavaScript</title>
</head>
<body>
<script>
A=54;
 B=3;
    var r=1;
var t='';
for (j = 1; j <=A; j++){ //Перебор чисел
 if (j % 2!=0 && j % B==0 && A %j==0) //Условие проверки нечетности
числа, его кратности и того, что он является делителем
```

```
 {
 r=r*j; // Искомое произведение
 t=t+j+"*"; // Делители числа
 }
}
 t=t.slice(0,-1); //Удаляем последний символ строки - '*'
 document.write(t+'='+r); //Вывод результата
</script>
</body>
</html>
```

Кодта қолданылған функциялар мен әдістер:

- .slice(a,b) – А индекстен және В индекске дейінгі массивтің бір бөлігін қайтарады. Бұл жағдайда, соңғысын қоспағанда барлық элементтер - (0, -1).
- document.write(x) – әдіс, х мәтінді құжатқа қосатын

5. Тапсырма:

Екі қатардың түрлі символдарын (біреуінде барын ғана) көрсету.

Шешімі:

```
<!DOCTYPE>
<html>
<head>
<title>Задачипо JavaScript</title>
</head>
<body>
<script>
function f_click() {
var first = document.getElementById("1").value;
var second= document.getElementById("2").value;
var b_lines = document.getElementById('lines');
var S = first.split('');
var lj = S.length;
for (j = 0; j < lj; j++)
 if (second.indexOf(S[j]) != -1)
 {first = first.split (S[j]).join(''); second = second.split(S[j]).join('')}
b_lines.innerHTML=((first + second).split(''));
}
</script>
Введите 1 строку: <br />
```

```
<input id='1' name="login" type="text" size="25" maxlength="30" value="" />
<br />
Введите 2 строку: <br />
<input id='2' name="pd" type="text" size="25" maxlength="30" value="" />
<br />
<input type="button" name="enter" value="Расчет" onclick="f_click()" />
<div style="margin: 0 auto;width: 200px; font-size:25px" id="lines"></div>
</body>
</html>
```

Кодта қолданылған функциялар мен әдістер:

- document.getElementById(id) – онымен әрі қарай жұмыс үшін белгіленген ID элементі қайтарылады.
- .value –"value" элементінің мәні қайтарылады.
- .length – массив ұзындығы, яғни қатар ұзындығы.
- .split(x) – жаңа массивті қайтарады, X қатар немесе тұрақты өрнек, оған сәйкес қатар бөлінеді
- .indexOf(x) – қатарда солдан оңға қарай x іздеу. Егер x табылмаса, онда -1 қайтарылады..
- .join(x) –массивтің барлық элементтерінқатар ішіне жинақтайды, X – қатарда болатын элементтерарасындағы бөлгіш.
- .innerHTML – элемент мазмұнын алады, өзгертеді.

1-15 ТАҚЫРЫПТАРЫН ҚОРЫТЫНДЫЛАУ СҰРАҚТАРЫ?

1. Тақырып 1. Javascript-тің негізгі ұғымдары.
   1.1. JavaScript бағдарламалау тілі қандай ынтымақтастықта және қай жылы қолданысқа енді?
   1.2. JavaScript бағдарламалау тілі қандай фирмамен әзірленді?
   1.3. JavaScript-та негізгі пайдалану саласында нені қарастырады?
   1.4. Қарапайым деректер типі не?
   1.5. Литерал дегеніміз не?
   1.6. Литерал түрлері мен оның анықтамалары қандай?
   1.7. Айнымалылар дегеніміз не және олардың операторлары қандай?
   1.8. Өрнектер дегеніміз не және олардың операторлары қандай?

1.9. Javascript-та логикалық операциялар қандай?
1.10. Логикалық операторлардың әсер ету нәтижесі операнд мәндерінің түрлі комбинациялар кестесі қандай?

2. Тақырып 2. Javascript-кодын қайда орналастыруға болады.
2.1. Javascript тілінде жазылған сценарийлер қайда және қалай орналастырылуы мүмкін?
2.2. Орналастырылу тегтары қандай?
2.3. Javascript-кодының сыртқы файлда жазылуы қандай?
2.4. script.js – парағы не?
2.5. <script> тегі қызметі?
2.6. language параметрі қызметі?

3. Тақырып 3. Бірінші бағдарлама, оқиғаларды өңдеу.
3.1. Бағдарламаны жазуда javascript-кодтарын html-параққа қалай енгіземіз?
3.2. html-парақты түсіндіру кезінде браузерде иерархиялық құрылым түрінде сақталу қандай?
3.3. Ең жоғарғы деңгейде, браузердің терезесін білдіретін "нақты" болып табылатын қандай объекті орналасады?
3.4. Объектілердің әдістері қандай?
3.5. Javascript қолдайтын оқиғаларды тізімдеңіз?

4. Тақырып 4. Javascript-функцияларды құру.
4.1. Функциялардың синтаксисті иеленеуін жазыңыз?
4.2. function кілт сөзі қызметі қандай?
4.3. div-у қызметі қандай?
4.4. "showMessage" функциясы қызметі қандай?
4.5. Javascript-та стандартты функциялар жиынтығы қандай?
4.6. Оқиға функцияларының денесін қалай жазамыз?

5. Тақырып 5. Функция параметрлері.
5.1. Функция параметрлері ретінде қандай мәнді беруге болады?
5.2. html-парақта қандай функциялар болуы тиіс екенін көрсету жолдары қандай?
5.3. Функцияларда біз ненің атын қолданамыз?
5.4. Функцияны жазу түрлері қандай?
5.5. Сценарий жұмысын браузерде тексеру жолдары?

6. Тақырып 6. Javascript-та Math объектісі.
6.1. math объектісі қандай?
6.2. math әдістері қандай?
6.3. math объектісі қандай функциялармен жұмыс істейді?
6.4. Бөлшекті сан нәтижесін үтірден кейін N-ге дейінгі белгімен дөңгелектеу үшін қандай объектінің, қандай әдісін пайдалануға болады?
6.5. Дөңгелектеудің синтаксис жазбасы қандай?

7. Тақырып 7. Тармақталу бағдарламасы - if операторы.

7.1. Шартты оператор қандай?
7.2. Шартты операторы синтаксис жазбасы қандай?
7.3. Функцияларда шартты операторлан басқа, тағы javascript қандай стандартты функцияны қолданамыз?
7.4. bigPict функциясы қызметі және синтаксисі қандай?
7.5. setTimeout функциясы қызметі және синтаксисі қандай?

8. Тақырып 8. switch таңдау операторы.
8.1. if операторын қалай пайдаланамыз?
8.2. switch таңдау операторының қызметі және синтаксисі қандай?
8.3. break операторы қызметі қандай?
8.4. switch операторы қызметі қандай?
8.5. Тізімнің айналыс тәсілінің көріну жолдары қандай?

9. Тақырып 9. web-парақтардың объектілерін басқару.
9.1. JavaScript тілінде web-парақта барлық элементтер қандай құрылыммен орнатылады?
9.2. Құрылымның әрбір объектісінің өз аты мен индексі қандай?
9.3. Индекс объектінің ережесімен қайда анықталады
9.4. Парақтың жоғарында орналасқан сурет қандай нөмірді иеленеді?
9.5. Нөмірлеу нешеден басталады?
9.6. forms объекттің қасиеті қандай?
9.7. elements[6] объектісі қандай?
9.8. Объектіге кіруді алу үшін нелерді көрсету керек?
9.9. value қасиеті не үшін пайдаланылады?
9.10. Пароль енгізуді қалай жүзеге асырамыз?

10. Тақырып 10. for және while циклдар.
10.1. for циклі. Оның синтаксисі қандай?
10.2. for циклінің орындалу жолы қандай?
10.3. while циклі. Оның синтаксисі қандай?
10.4. while циклінің орындалу жолы қандай?

11. Тақырып 11. Күндер, ұсынулар және өңдеулер.
11.1. JavaScript-та күн қалай анықталады?
11.2. Date объектісі қызметі қандай?
11.3. Date объекті әдістері қандай?
11.4. nTime()функциясының қызметі қандай?
11.5. tData() функциясы қызметі қандай?
11.6. Күнді орналыстыруда қандай әдістерді пайдалану керек?
11.7. Date объектісі қандай оператор және конструктор көмегімен құрылады

12. Тақырып 12. Массивтер.
12.1. Массив нелер жиынтығын білдіреді және анықтамасы қандай?
12.2. Массив нелердің көмегімен құрылады?
12.3. Массив ұзындығын анықтау үшін қандай қасиет пайдаланылады?

12.4. showDay() функциясын жазу жолы қандай?
12.5. concat() әдісі көмегімен не істеуге болады?
12.6. proverka() функциясының қызметі қандай?
12.7. Қандай алгоритмді пайдаланамыз?

13. Тақырып 13. Қатарлар.
13.1. Қатарлар түрі мен ұғымдары қандай?
13.2. Қатар объектісін жасау үшін қандай конструкторды пайдалануға болады?
13.3. String - length қасиеті қолдансақ, онда біз нені білеміз?
13.4. Қатарға әсер ететін әдістер қандай?
13.5. numword () функциясының қызметі қандай?
13.6. Javascript тілінде бірқатар стандартты функциялардың қатарлармен жұмысы қалай анықталған?
13.7. Number(s) стандартты функциясының қызметі қандай?

14. Тақырып 14. Тұрақты өрнектер.
14.1. Тұрақты өрнек қызметі қандай?
14.2. Тұрақты өрнекті құрудың қанша тәсілі бар және қандай?
14.3. pattern - қызметі қандай?
14.4. flags - қызметі қандай?
14.5. prov_adress() функциясы қызметі қандай?
14.6. adr_pattern.test(adr) қатары нелерді білдіреді?
14.7. Тұрақты өрнекті білдіруде пароль үшін нелерді аламыз?
14.8. тұрақты өрнекті құру міндеті және литерал мен метасимволдар қандай?

15. Тақырып 15. Терезелермен жұмыс.
15.1. window объектісінің қанша әдісін пайдаланамыз және қандай?
15.2. window объектінің open() әдісінің қызметі қандай?
15.3. open() әдісінің қанша параметрлері бар және қандай?
15.4. confirm-әдісі. Оның синтаксисі қандай?
15.5. prompt-әдісі. Оның синтаксисі қандай?
15.6. setTimeout-әдісі. Оның синтаксисі қандай?
15.7. clearTimeout-әдісі. Оның синтаксисі қандай?

ҚОЛДАНЫЛҒАН ӘДЕБИЕТТЕР

1 Прохоренок, Н. HTML, JavaScript, PHP и MySQL Джентльменский набор Web-мастера (+CD-ROM) [Текст] / Н. Прохоренок. - 3-е изд., перераб. и доп. - СПб. : БХВ-Петербург, 2010. - 912 с. : ил. - (Профессиональное программирование). - ISBN978-5-9775-0540-6

2 Брогден, Б. Электронный магазин на Java и XML [Текст] / Б. Брогден, К. Минник - СПб. : Питер, 2002. - 400 с. : ил. - ("Библиотека программиста"). - ISBN5318-00-400-8

3 Перроун Старший, Поль Дж. Создание корпоративных систем на базе Java 2 Enterprise Edition [Текст] : рук. разработчика : [пер. с

англ.] / П. Д. Перроун Старший, С.Р. Венката "Кришна", Р. Чаганти [и др.] - М.: Вильямс, 2001. - 1179 с. - ISBN 5-8459-0168-0233-4
4 Пыркова, А.Ю.Создание и использование JAVA-приложений в среде MicrosoftVisualJ++6.0 [Текст] : метод. пособие для студ. спец. 010540 - "Информатика", 010240 - "Прикладная информатика" / А.Ю. Пыркова. - Алматы : Қазақ университеті, 2004. - 96 с. - ISBN9965-12-646-1
5 Свистунов, А. Н. Построение распределенных систем на Java [Текст] : учеб. пособие / А.Н. Свистунов. - М. : Интернет-Университет Информационных Технологий; БИНОМ. Лаборатория знаний, 2011. - 279 с. : ил. - (Основы информационных технологий). - ISBN978-5-9963-0444-8
6 Столбовский, Д.Н.Основы разработки Web-приложений на ASP.NET [Текст] : учеб. пособие / Д.Н. Столбовский. - М. : Интернет-Университет Информационных Технологий; Бином. Лаборатория знаний, 2009. - 304 с. : ил. - (Основы информационных технологий). - ISBN978-5-94774-991-5
7 Храмцов,П.Б. Основы Web-технологий [Текст] : учеб. пособие для студ. вузов / П.Б. Храмцов [и др.]. - 2-е изд., испр. - М. : Интернет-Университет Информационных Технологий; Бином. Лаборатория знаний, 2009. - 374 с. : ил. - (Основы информационных технологий). - ISBN978-5-94774-648-8

МАННАПОВА ТОРҒЫН МЕНДИКУЛОВНА
ҚАСЫМОВА АҚМАРАЛ ХАМЗИЕВНА

Маннапова Т.М., магистр, аға оқытушы
Қасымова А.Х., п.ғ.к., доцент