# Constructor and prototype

## BasicConstructor

### Date

// при вызове конструктора Date без параметров, создается объект Date отражающий текущую дату и время

var myDate = new Date();

document.write("Текущая дата и время: " + myDate);

### new String

// переменая со строковым значеним

var simpleStr = "My String";

// объект типа String

var objectStr = new String("some String object");

function newLine(str) {

document.write(str + "<br />");

document.write("==========================================================<br />");

}

newLine(simpleStr);

newLine(objectStr);

// добавлять новые свойства и методы можно только к объектам созданым через оператор new

objectStr.customProperty = 123;

document.write(objectStr.customProperty + "<br />");

// добавить свойство в simpleStr невозможно

simpleStr.customProperty2 = 123;

### Function constructor

// Создаем функцию, которая принимает два параметра с именем x и y и тело, которое возвращает их сумму.

var func = new Function("x", "y", "return x + y;");

var result = func(20, 10);

alert("Результат выполнения первой функции " + result);

// создаем функцию без параметров, которая отображает модальное окно

var func2 = new Function("alert('Hello!!!')");

func2();

</script>

### New function constructor

// Функция-конструктор для создания объектов Human.

function Human(name) {

// свойство

this.firstName = name;

// метод - свойство объекта, которое содежит значение в виде функции.

this.sayHello = function () {

document.write("Hello! My name is " + this.firstName + "<br /> ");

}

}

// Создание двух экземпляров типа Human

var h = new Human("Homer");

var r = new Human("Rocky");

// Вызов метода SayHello() на созданых обьектах.

h.sayHello();

r.sayHello();

## Prototype

### Without prototype

function Rectangle(w, h) {

// свойства

this.width = w;

this.height = h;

// метод

// недостатком данного метода, является то, что он будет дублироваться во всех объектах созданных через функцию

// Rectangle. Метод не будет меняться от объекта к объекту но при это у будет занимать память, что не оптимально.

this.getArea = function () {

return this.width \* this.height;

}

}

var rect = new Rectangle(100, 50);

document.write("Площадь прямоугольника " + rect.getArea());

### With prototype

function Rectangle(w, h) {

// свойства

this.width = w;

this.height = h;

}

// Метод прототипа Rectangle будет доступен каждому экземпляру, но храниться будет в прототипе соответственно занимать меньше места

// чем если бы метод принадлежал каждому экземпляру.

Rectangle.prototype.getArea = function () {

return this.width \* this.height;

}

var rect = new Rectangle(100, 50);

document.write("Площадь прямоугольника " + rect.getArea());

### Property constructor

// Создаем несколько обьектов используя различные конструкторы

var MyArray = new Array(10);

var MyDate = new Date();

var MyString = new String("Some string value");

var MyObj = new Object();

var MyFunc = new Function("x", "y", "return x + y;");

var MyCtor = function (x, y) {

this.x = x;

this.y = y;

}

var MyCtorObject = new MyCtor(12, 3);

// Создаем функцию для вывода содержимого свойства constructor аргумента

function showCtor(obj, name) {

document.write("Конструктор обьекта " + name + " это " + obj.constructor + "<br />");

}

document.write("===============================================Значения конструкторов====================================<br />");

showCtor(MyArray, "MyArray");

showCtor(MyDate, "MyDate");

showCtor(MyString, "MyString");

showCtor(MyObj, "MyObj");

showCtor(MyFunc, "MyFunc");

showCtor(MyCtorObject, "MyCtorObject");

document.write("======================================================================================================<br />");

// Вызываем конструктор с помощью которого был создан объект MyDate

var someNewObject = new MyDate.constructor();

document.write("Значение обьекта someNewObject: " + someNewObject);

## Object methods

### ToString

function Rectangle(w, h) {

width = w;

this.height = h;

}

Rectangle.prototype.getArea = function () {

return this.width \* this.height;

}

// переопределение метода toString из Object

Rectangle.prototype.toString = function () {

return "Прямоугольник W:" + this.width + " H:" + this.height;

}

var rect1 = new Rectangle(100, 200);

var rect2 = new Rectangle(400, 300);

document.write(rect1 + "<br />");

document.write(rect2.toString() + "<br />");

### ValueOf

<script type="text/javascript">

function Rectangle(w, h) {

this.width = w;

this.height = h;

}

Rectangle.prototype.getArea = function () {

return this.width \* this.height;

}

// переопределение метода toString из Object

Rectangle.prototype.toString = function () {

return "Прямоугольник W:" + this.width + " H:" + this.height;

}

// переопределение метода valueOf из Object

Rectangle.prototype.valueOf = function () {

return this.getArea();

}

var rect1 = new Rectangle(100, 200);

var rect2 = new Rectangle(400, 300);

document.write(rect1 + "<br />"); // по умолчанию вызывается метод valueOf()

document.write(rect2.toString() + "<br />");

document.write("<br />values<br />");

document.write(rect1.valueOf() + "<br />");

document.write(rect2.valueOf() + "<br />");

document.write(rect2 + rect1);

</script>

### HasOwnProperty

<script type="text/javascript">

function Rectangle(w, h) {

this.width = w;

this.height = h;

}

Rectangle.prototype.getArea = function () {

return this.width \* this.height;

}

Rectangle.prototype.name = "RECTANGLE";

var rect1 = new Rectangle(100, 200);

// hasOwnProperty метод из Object с его помощью можно проверить существует ли свойство у конкретного объекта не проверяя наличие свойств у прототипа.

document.write("hasOwnProperty('width') " + rect1.hasOwnProperty("width") + "<br />");

document.write("hasOwnProperty('name') " + rect1.hasOwnProperty("name") + "<br />");

// с помощью ключевого слова in можно проверить наличие свойства у объекта включая свойства прототипов.

document.write("'width' in rect1 " + ("width" in rect1) + "<br />");

document.write("'name' in rect1 " + ("name" in rect1) + "<br />");

</script>

### Object equality by references

<script type="text/javascript">

function Rectangle(w, h) {

this.width = w;

this.height = h;

}

// объекты имеют одинаковые значения.

var rect1 = new Rectangle(100, 200);

var rect2 = new Rectangle(100, 200);

// оператор сравнения проверяет равенство ссылок а не равенство значений свойств объектов

var res = rect1 == rect2;

document.write("rect1 == rect2 -> " + res);//false

var rect3 = rect1; // rect1 и rect3 ссылки на один и тот же объект в памяти

res = rect3 == rect1;

document.write("<br /><br />rect3 == rect1 -> " + res);//true

### Object equality by values

<script type="text/javascript">

function Rectangle(w, h) {

this.width = w;

this.height = h;

}

// метод для проверки равенства объектов

Rectangle.prototype.equals = function (otherObj) {

if (this.width === otherObj.width && this.height === otherObj.height) {

return true;

}

return false;

}

// объекты имеют одинаковые значения.

var rect1 = new Rectangle(100, 200);

var rect2 = new Rectangle(100, 200);

// проверка равенства объектов с помощью метода equals

var res = rect1.equals(rect2);

document.write("rect1 == rect2 -> " + res);//true;

var rect3 = rect1; // rect1 и rect3 ссылки на один и тот же объект в памяти

res = rect3 == rect1;

document.write("<br /><br />rect3 == rect2 -> " + res);//true;

### CompareTo

<script type="text/javascript">

function Rectangle(w, h) {

this.width = w;

this.height = h;

}

Rectangle.prototype.getArea = function () {

return this.width \* this.height;

}

// переопределение метода toString из Object

Rectangle.prototype.toString = function () {

return "Прямоугольник W:" + this.width + " H:" + this.height;

}

// переопределение метода valueOf из Object

Rectangle.prototype.valueOf = function () {

return this.getArea();

}

// метод для проверки равенства объектов

Rectangle.prototype.equals = function (that) {

if (this.width === that.width && this.height === that.height) {

return true;

}

return false;

}

// Метод для сравнения объеков.

// Если текущий объект больше чем тот который передан в параметры - возвращаем значение > 0

// Если текущий объект меньше чем тот который передан в параметры - возвращаем значение < 0

// Если объекты равны возвращаем 0

Rectangle.prototype.compareTo = function (otherObject) {

if (this.getArea() > otherObject.getArea()) {

return 1;

}

else if (this.getArea() < otherObject.getArea()) {

return -1;

}

else {

return 0;

}

// или таким способом

return this.getArea() - otherObject.getArea();

}

// объекты имеют одинаковые значения.

var rect1 = new Rectangle(1000, 20);

var rect2 = new Rectangle(1000, 20);

// при наличии метода valueOf(), который возвращает простое значение операторы сравнения будут использовать это значение для

// определения какой объект больше, а какой меньше

// если реализация метода valueOf() не возможна, то в таком случае, рекомендуется реализовать метода compareTo(a)

if (rect1 > rect2) {

document.write("Объкет rect1 > rect2<br />");

}

if (rect1 < rect2) {

document.write("Объкет rect1 < rect2<br />");

}

if (rect1 <= rect2) {

document.write("Объкет rect1 <= rect2<br />");

}

if (rect1 >= rect2) {

document.write("Объкет rect1 >= rect2<br />");

}

if (rect1.valueOf() == rect2.valueOf()) {

document.write("Объкет rect1 == rect2<br />");

}

if (rect1.valueOf() != rect2.valueOf()) {

document.write("Объкет rect1 != rect2<br />");

}

document.write("<br/><br/>использование compareTo()<br/><br/>");

if (rect1.compareTo(rect2) > 0)/\*rect1 > rect2\*/ {

document.write("Объкет rect1 > rect2<br />");

}

if (rect1.compareTo(rect2) < 0)/\*rect1 < rect2\*/ {

document.write("Объкет rect1 < rect2<br />");

}

if (rect1.compareTo(rect2) <= 0)/\*rect1 <= rect2\*/ {

document.write("Объкет rect1 <= rect2<br />");

}

if (rect1.compareTo(rect2) >= 0)/\*rect1 >= rect2\*/ {

document.write("Объкет rect1 >= rect2<br />");

}

if (rect1.compareTo(rect2) == 0)/\*rect1 == rect2\*/ {

document.write("Объкет rect1 == rect2<br />");

}

if (rect1.compareTo(rect2) != 0)/\*rect1 != rect2\*/ {

document.write("Объкет rect1 != rect2<br />");

}

## OOP

### Incapsulation

var MyClass = function () {

// закрытый метод

var privateMethod = function () {

document.write("Закрытый метод");

};

// открытый метод

this.publicMethod = function () {

document.write("Открытый метод <br />");

privateMethod();

};

};

var obj = new MyClass();

obj.publicMethod();

obj.privateMethod(); // ошибка

### Inheritance

Насле́дование — механизм объектно-ориентированного программирования (наряду с инкапсуляцией, полиморфизмом и абстракцией), позволяющий описать

новый класс на основе уже существующего (родительского), при этом свойства и функциональность родительского класса заимствуются новым классом.

Другими словами, класс-наследник реализует спецификацию уже существующего класса (базовый класс). Это позволяет обращаться с объектами

класса-наследника точно так же, как с объектами базового класса.

-->

<html>

<head>

<title>Наследование в JavaScript</title>

<script type="text/javascript">

// Наследование

// базовая функция-конструктор

function Human(name) {

this.name = name;

this.talk = function () { document.write("Hello My Name is " + this.name + "<br />") }

}

// производная функция конструктор

function Student(name) {

this.name = name;

this.school = "CBS school";

}

function Worker(name) {

this.name = name;

this.speciality = "Software developer";

}

// Создаем экземпляр родителя

var human = new Human("name");

//human.talk();

// наследуем Student и Worker от Human устанавливая функцию как прототип.

Student.prototype = human;

Worker.prototype = human;

var Alex = new Student("Alexander");

var Andrew = new Worker("Andrew");

document.write("<br />Student<br />");

Alex.talk();

document.write(Alex.school);

document.write("<br />Worker<br />");

Andrew.talk();

document.write(Andrew.speciality);

### Polimorphizm

Только с помощью замещения

Механизм сначала проверяет объект, потом идет к прототипу.

Полиморфи́зм (в языках программирования) — возможность объектов с одинаковой спецификацией иметь различную реализацию.

-->

<html>

<head>

<title>Полиморфизм в JavaScript</title>

<script type="text/javascript">

// Наследование

// базовая функция-конструктор

function Human(name) {

this.name = name;

this.talk = function () { document.write("Hello My Name is " + this.name + "<br />") }

}

// производная функция конструктор

function Student(name) {

this.name = name;

this.school = "CBS school";

}

function Worker(name) {

this.name = name;

this.speciality = "Software developer";

// переопределение метода из базовой функции-конструктора (override).

this.talk = function () { document.write("Привет мое имя " + this.name + "<br />") }

}

// Создаем экземпляр родителя

var human = new Human("name");

//human.talk();

// наследуем Student и Worker от Human устанавливая функцию как прототип.

Student.prototype = human;

Worker.prototype = human;

var Alex = new Student("Alexander");

var Andrew = new Worker("Andrew");

document.write("<br />Student<br />");

Alex.talk();

document.write(Alex.school);

document.write("<br />Worker<br />");

Andrew.talk();

document.write(Andrew.speciality);

### InstanceOf

<script type="text/javascript">

var x = new Array(10);

if (x instanceof Array) { // аналог оператора is в C#

alert("x is array");

}

</script>

### Typeof

var test = "hello";

document.write(typeof test); // аналог GetType() в C#

document.write("<br />");

var test2 = 123;

document.write(typeof test2);

document.write("<br />");

var test3 = true;

document.write(typeof test3);

document.write("<br />");

var test4 = new Array();

document.write(typeof test4);

document.write("<br />");

var test5 = new Date();

document.write(typeof test5);

document.write("<br />");

document.write(typeof test5.toString);

document.write("<br />");

# Work with document

## Js integration

<title>Встраивание JavaScript кода в HTML страницу</title>

<!--В тэге Script в head страницы----------------------------------------------->

<script type="text/javascript">

// JavaScript код

</script>

<!--Из внешнего файла----------------------------------------------------------->

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

</head>

<body>

<!--В качестве обработчика события через html атрибут.-------------------------->

<input type="button" value="Кнопка" onclick="alert('Hello')" />

<br />

<!--Как тело URL адреса--------------------------------------------------------->

<a href="javascript:alert('Hello')">Ссылка</a>

<br />

<!--В тэге Script в body страницы----------------------------------------------->

<script type="text/javascript">

// JavaScript код

</script>

<!--Из внешнего файла----------------------------------------------------------->

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

</body>

## Handlers

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title>Обработчики на события</title>

<script type="text/javascript">

// Обработчик события click - нажатие по кнопке.

function buttonClickHandler() {

alert("Кнопка нажата.");

}

</script>

</head>

<body>

<input type="button" value="Кнопка" onclick="buttonClickHandler()" />

</body>

</html>

## JS Module

### BadPractice

<head>

<title>Зачем нужны пространства имен и модули</title>

<script src="BadModule1.js"></script>

<script src="BadModule2.js"></script>

<script>

// из-за глобальной переменной name метод startModule1() и startModule2() будут выводить одинаковые значения.

startModule1();

startModule2();

</script>

</head>

File BadModule1.js

var name = "MODULE 1"; // глобальная переменная

function startModule1() {

document.write(name + "<br />");

}

### BestPractice

<head>

<title>Использование модулей</title>

<script src="Module1.js"></script>

<script src="Module2.js"></script>

<script>

// В данном примере Module1 и Module2 выступают в роли пространств имен для переменной.

Module1.startModule1();

Module2.startModule2();

</script>

</head>

File.js

// -------------------------------------------------

// Module1.js модуль вспомогательных функций.

// Данный модуль единственное глобальное имя Module1

// -------------------------------------------------

// Пустой объект, который будет выполнять функцию проcтсранства имен.

// Это единственное глобальное имя, которое будет вмещать все остальные имена.

// Правило: имя модуля должно быть связано с именем файла

var Module1 = {};

Module1.name = "MODULE 1";

Module1.startModule1 = function () {

document.write(Module1.name + "<br />");

}

### BestPractice2

<head>

<title>Использование модулей</title>

<script src="Module3.js"></script>

<script src="Module4.js"></script>

</head>

File.js

(function () {// Определяем анонимную функцию.

// Отсутствие имени означает отсутствие глобального имени

// Тело функции

// здесь можно объявлять любые переменные - это не приведет к созданию глобальных переменных.

var message = "Hello from module3"; // переменная не глобальная

alert(message);

})(); // конец определения функции и ее вызов.

## Document

### Document writeln

<script type="text/javascript">

function ButtonHandler() {

// При вызове метода document.write() после формирования страницы происходит создание нового документа.

// Данный метод не следует вызывать в обработчиках событий, что бы не потерять содержимое текущей страницы.

document.write("Hello world");

}

</script>

</head>

<body>

<!--Использование метода write-->

<p style="padding: 8px; margin: 8px; color: Green; font-weight: bold; border: solid 1px black;">

<script type="text/javascript">

document.write("hello");

document.write(" 1", "-", "2", "-", "3");

</script>

</p>

<!--Использование метода writeln-->

<pre style="padding: 8px; margin: 8px; color: Green; font-weight: bold; border: solid 1px black;">

<script type="text/javascript">

document.writeln("hello");

document.writeln("world");

document.writeln(location.hostname);

</script>

</pre>

<input type="button" value="Кнопка" onclick="ButtonHandler()" />

### Document properties

<script type="text/javascript">

// bgColor - цвет фона документа.

document.write("<b>bgColor</b>: " + document.bgColor + "<br />");

// cookie - свойство позволяет читать и писать куки файлы.

document.write("<b>cookie</b>: " + document.cookie + "<br />");

// lastModified - дата последнего изменения документа.

document.write("<b>lastModified</b>: " + document.lastModified + "<br />");

// location - устаревший синоним свойства URL

document.write("<b>location</b>: " + document.location + "<br />");

// referrer - адрес документа, содержащего ссылку, которая привела на текущую страницу.

document.write("<b>referrer</b>: " + document.referrer + "<br />");

// title - содержимое тега title

document.write("<b>title</b>: " + document.title + "<br />");

</script>

<!--якорь-->

<a name="anchor">Anchor</a>

<!--ссылка ведущая к якорю-->

<a href="#anchor">Go To Anchor</a>

<script type="text/javascript">

// anchors - массив якорей(именованых позицый документа) - тегов <a> c атрибутом name.

document.write("<br /><b>anchors</b>: " + document.anchors[0].name + "<br />");

// forms - массив форм HTML документа.

document.write("<br /><b>forms</b>: " + document.forms + "<br />");

// images - массив изображений HTML документа.

document.write("<br /><b>images</b>: " + document.images + "<br />");

// links - массив ссылок HTML документа.

document.write("<br /><b>links</b>: " + document.links + "<br />");

</script>

### GetElementById

<body>

<div id="div1">

<p id="output">Hello world</p>

<a id="link" href="http://edu.cbsystematics.com">Education site</a>

</div>

<script type="text/javascript">

// скрипт должен располагаться после элементов HTML, которые в этом скрипте используются.

// получение элемента со страницы по значению атрибута id

var paragraph = document.getElementById("output");

var hyperlink = document.getElementById("link");

// innerHTML - текст между открывающим и закрывающим тегами.

paragraph.innerHTML = "HELLO WORLD !!!";

hyperlink.innerHTML = "lul";

### Window onload

<script>

// Ненавязчивый JavaScript (Unobtrusive JavaScript) – отделение функциональности страницы от ее структуры.

// JavaScript может находиться в отдельных модулях и блоки JavaScript кода не должны появляться в разметке.

// событие срабатывает после полной загрузки окна.

window.onload = function () {

// получение элемента со страницы по значению атрибута id

var paragraph = document.getElementById("output");

var hyperlink = document.getElementById("link");

paragraph.innerHTML = "HELLO WORLD !!!";

hyperlink.innerHTML = "EDUCATION SITE !!!";

}

</script>

</head>

<body>

<p id="output">Hello world</p>

<a id="link" href="http://edu.cbsystematics.com">Education site</a>

</body

### GetElementByName

<script type="text/javascript">

window.onload = function () {

// обработчик на событие нажатие по кнопке

document.getElementById("answerButton").onclick = function () {

// получение массива элементов с атрибутом name со значеним answer

var radioButtons = document.getElementsByName("answer");

var res = "";

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

res += radioButtons[i].value + " " + radioButtons[i].checked + "\r\n";

}

alert(res);

}

}

</script>

</head>

<body>

Да<input type="radio" name="answer" value="Да" />

Нет<input type="radio" name="answer" value="Нет" />

Не знаю<input type="radio" name="answer" value="Не знаю" />

<input id="answerButton" type="button" value="Получить ответ" />

</body>

### GetElementByTagName

<head>

<title>getElementsByTagName</title>

<script type="text/javascript">

window.onload = function () {

// получение всех элементов <p>

var paragraphs = document.getElementsByTagName("p");

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

// смена CSS атрибута color для всех параграфов

paragraphs[i].style.color = "red";

}

}

</script>

</head>

<body>

<p style="color:green">Hello world!!!</p>

<p>Hello world!!!</p>

<p>Hello world!!!</p>

<p>Hello world!!!</p>

<p>Hello world!!!</p>

</body>

## DOM

### ChildNodes

<script type="text/javascript">

// Основные типы узлов

// Интерфейс | Константа nodeType | Значение nodeType

// ------------------------------------------------------------------------

// Element | Node.ELEMENT\_NODE | 1

// Text | Node.TEXT\_NODE | 3

// Comment | Node.COMMENT\_NODE | 8

// Document | Node.DOCUMENT\_NODE | 9

// ------------------------------------------------------------------------

// Свойства для навигации по DOM

// childNodes - массив всех дочерних узлов

// firstChild - первый дочерний узел

// lastChild - последний дочерний узел

// nextSibling - следующий узел расположенный на одном уровне с текущим

// previousSibling - предыдущий узел расположенный на одном уровне с текущим

// parentNode - родительский узел для текущего элемента

window.onload = function () {

var e = document.getElementById("list");

// childNodes - получение всех дочерних узлов.

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

e.childNodes[i].style.color = "red";

alert(e.childNodes[i].nodeType);

}

}

</script>

</head>

<body>

<ul id="list"><li>Item 1</li><li>Item 2</li><li>Item 3</li></ul>

</body>

### NodeType

<script type="text/javascript">

window.onload = function () {

var e = document.getElementById("list");

// childNodes - получение всех дочерних узлов.

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

var temp = e.childNodes[i];

// если найденный узел не элемент то пропустить его

if (temp.nodeType != 1) {

continue;

}

e.childNodes[i].style.color = "red";

}

}

</script>

</head>

<body>

<!-- После каждого элемента находиться текстовый узел - перенос на новую строку. -->

<ul id="list">

<li>Item 1</li>

<li>Item 2</li>

<li>Item 3</li>

</ul>

### ParentNode

<script type="text/javascript">

window.onload = function () {

var e = document.getElementById("iz");

// parentNode - получить родительский узел для узла 'e'

e.style.border= "1px solid red";

}

</script>

</head>

<body>

<div id="iz">

Div

<p id="p1">Paragraph</p>

</div>

### First and last child

<script type="text/javascript">

window.onload = function () {

var e = document.getElementById("list");

// firstChild - получить первый дочерний узел

e.firstChild.style.color = "red";

// lastChild - получить последний дочерний узел

e.lastChild.style.color = "green";

}

</script>

</head>

<body>

<ul id="list"><li>Item 1</li><li>Item 2</li><li>Item 3</li></ul>

</body>

### Sibling

<script type="text/javascript">

window.onload = function () {

var e = document.getElementsByTagName("li")[2];

// nextSibling - получение следующего узла, который находиться на одном уровне с текущим (узлы имеют общий родительский узел)

e.nextSibling.style.color = "red";

// previousSibling - получение предыдущего узла, который находиться на одном уровне с текущим.

e.previousSibling.style.color = "green";

}

</script>

</head>

<body>

<ul id="list">

<li>Item 1</li><li>Item 2</li><li>Item 3</li><li>Item 4</li><li>Item 5</li>

</ul>

### Append|remove|insert child

<script>

// Методы для работы с DOM

// appendChild - добавить дочерний элемент

// insertBefore - добавить элемент перед указанным

// removeChild - удаление элемента

// replaceChild - замена одного элемента на другой

window.onload = function () {

var div1 = document.getElementById("div1");

var p1 = document.createElement("p");

p1.innerHTML = "Paragraph 1";

div1.appendChild(p1);

var p2 = document.createElement("p");

p2.innerHTML = "Paragraph 2";

document.body.insertBefore(p2, div1);

}

</script>

</head>

<body>

<div id="div1" style="border:solid 1px red; padding:10px; margin:10px;">

</div>

### Set|get|remove attribute

<script>

window.onload = function () {

var a1 = document.getElementById("a1");

var a2 = document.getElementById("a2");

var a3 = document.getElementById("a3");

// установка атрибута

a1.setAttribute("href", "http://edu.cbsystematics.com");

// удаление атрибута

a2.removeAttribute("href");

// чтение атрибута

alert(a3.getAttribute("href"));

}

</script>

</head>

<body>

<a id="a1">Text a1</a>

<a id="a2" href="http://edu.cbsystematics.com">Text a2</a>

<a id="a3" href="http://edu.cbsystematics.com">Text a3</a>

</body>

### Sort DOM

<script type="text/javascript">

// Основные типы узлов

// Интерфейс | Константа nodeType | Значение nodeType

// ------------------------------------------------------------------------

// Element | Node.ELEMENT\_NODE | 1

// Text | Node.TEXT\_NODE | 3

// Comment | Node.COMMENT\_NODE | 8

// Document | Node.DOCUMENT\_NODE | 9

// ------------------------------------------------------------------------

function sortchildren(e) {

if (typeof e == "string") e = document.getElementById(e);

// Преобразовываем дочерние элементы в массив

var children = [];

for (var x = e.firstChild; x != null; x = x.nextSibling)

if (x.nodeType == 1 /\* Node.ELEMENT\_NODE \*/) children.push(x);

// сортировка

children.sort(function (n, m) { // функция сравнения.

var s = n.firstChild.data; // текст узла n

var t = m.firstChild.data; // текст узла m

if (s < t) return -1; // n перед m

else if (s > t) return 1; // n после m

else return 0; // n и m равны

});

// Добавляем дочерние элементы обратно в родительский узел.

// Когда добавляется узел, который уже существует в документе, он автоматически удаляется со своей старой позиции и вставляется в новую.

for (var i = 0; i < children.length; i++) e.appendChild(children[i]);

}

</script>

</head>

<body>

<ul id="mylist">

<li>one</li>

<li>two</li>

<li>three</li>

<li>four</li>

</ul>

<button onclick="sortchildren('mylist')">

Sort list</button>

# Window and regular expression

## Timers

### TimeOut

<script type="text/javascript">

window.onload = function () {

// setTimeout - планирует один запуск указанной функции через определенное количество миллисекунд

// функция showMessage будет запущена через 5 секунд

setTimeout(showMessage, 5000)

function showMessage() {

alert("Hello world");

}

// второй вариант запуска

//setTimeout(function () {

// alert("Hello world");

//}, 5000);

}

</script>

### SetInterval

<head>

<title>setInterval</title>

<script type="text/javascript">

window.onload = function () {

var counter = 0;

function Count() {

counter++;

document.getElementById("output").innerHTML = counter;

}

// setInterval - планирование запуска указанной функции с заданным интервалом. Планирование и запуск будут происходить многократно.

// запускать функцию Count с интервалом в 1 секунду

setInterval(Count, 1000);

}

</script>

</head>

<body>

<p id="output">0</p>

</body>

### Cancel timeout

<title>setTimeout and clearTimeout</title>

<script>

var timer;

window.onload = function () {

timer = window.setTimeout(showMessage, 5000);

}

function showMessage() {

alert("Hello world");

}

function cancel() {

// отмена задержки, которая была установлена на строке 9

clearTimeout(timer);

}

</script>

</head>

<body>

<p>Сообщение будет отображено через 5 секунд</p>

<a href="javascript:cancel()">Отмена</a>

</body>

### Start|stop timer

<script>

window.onload = function () {

var counter = 0;

var intervalHandler;

function count(){

get("output").innerHTML = counter;

counter++;

}

get("startButton").onclick = function () {

intervalHandler = setInterval(count, 500);

}

get("stopButton").onclick = function () {

// останавливаем таймер, по которому вызывается функция count

clearInterval(intervalHandler);

}

}

// пользовательская функция для получения элементов страницы по id

var get = function (id){

return document.getElementById(id);

}

</script>

</head>

<body>

<p id="output">-</p>

<input id="startButton" type="button" value="Start" />

<input id="stopButton" type="button" value="Stop" />

## Location

### Location

<title>Location</title>

<script type="text/javascript">

// location - URL адрес текущего документа.

document.write("location = " + document.location + "<br />"); // весь url

document.write("protocol = " + document.location.protocol + "<br />"); // протокол

document.write("host = " + document.location.host + "<br />"); // имя хоста

document.write("pathname = " + document.location.pathname + "<br />"); // путь к запрашиваемому ресурсу

document.write("search = " + document.location.search + "<br />"); // строка запроса (после знака вопроса)

// location - http://edu.cbsystematics.com/info.aspx?id=10

// protocol - http:

// host - edu.cbsystematics.com

// pathname - /info.aspx

// search - ?id=10

</script>

### UrlParse

<script type="text/javascript">

// Функция для чтения данных, которые переданы при GET запросе (в адресной строке).

function getQueryString() {

var args = {}; // пустой объект

// login=admin password=123

var query = location.search.substring(1); // Получение строки запроса.

var pairs = query.split("&"); // Разделение строки по амперсанду

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

var pos = pairs[i].indexOf('='); // Проверка пары "name=value"

if (pos == -1) { // Если не найдено - пропустить

continue;

}

var argname = pairs[i].substring(0, pos); // Получение имени

var value = pairs[i].substring(pos + 1); // Получение значения

args[argname] = value; // Сохранение как свойства

}

return args;

}

// Для теста примера перейдите по URL текущей страницы + ?login=admin&password=123

function print() {

var args = getQueryString();

var e = document.createElement("p");

e.innerHTML = "Login " + args.login + "<br />";

e.innerHTML += "Password " + args.password;

document.body.appendChild(e);

}

</script>

</head>

<body>

<input type="button" name="name" value="Read Query Parameters" onclick="print()" />

</body>

### Location ref

<head>

<title>Перенаправление</title>

<script>

window.onload = function () {

document.getElementById("go").onclick = function () {

document.location = "http://edu.cbsystematics.com/ru/courses";

}

}

</script>

</head>

<body>

<input id="go" type="button" value="azaza" />

</body>

</html>

## Navigator

<head>

<title>Объект Navigator</title>

<script type="text/javascript">

// Объект Navigator поддерживает пять свойств, по которым можно опередить версию браузера.

// appName - Название веб браузера.

// appVersion - Номер версии или другая информация о браузере

// userAgent - строка, которую браузер посылает в HTTP заголовке USER-AGENT

// appCodeName - Кодовое имя браузера. Для Netscape используется кодовое имя "Mozilla" Для совместимости IE делает то же самое.

// platform - Аппаратная платформа, на которой работает браузер.

var browserInfo = "СВЕДЕНЬЯ О БРАУЗЕРЕ:\n";

for (var propname in navigator) {

browserInfo += propname + ": " + navigator[propname] + "\n";

}

alert(browserInfo);

</script>

</head>

## Windows

### Windows geometry

<script type="text/javascript">

var Geometry = {};

// В разных браузерах существуют разноименные методы и свойства выполняющие схожие операции.

// Например, для получения координаты X окна браузера в IE нужно использовать свойство screenLeft,

// а в Mozilla - screenX.

// Для того что бы упростить написание кроссбраузерного кода, разработчики создают специальные объекты

// на подобие Geometry в этом примере.

// screenLeft, screenX, innerWidth - свойства объекта window специфичные для версии браузера.

if (window.screenLeft || window.screenLeft == 0) { // для IE

Geometry.getWindowX = function () { return window.screenLeft; };

Geometry.getWindowY = function () { return window.screenTop; };

}

else if (window.screenX || window.screenX == 0) { // для Firefox

Geometry.getWindowX = function () { return window.screenX; };

Geometry.getWindowY = function () { return window.screenY; };

}

if (window.innerWidth) {

Geometry.getViewportWidth = function () { return window.innerWidth; };

Geometry.getViewportHeight = function () { return window.innerHeight; };

}

else if (document.documentElement && document.documentElement.clientWidth) { // Для IE6

Geometry.getViewportWidth = function () { return document.documentElement.clientWidth; };

Geometry.getViewportHeight = function () { return document.documentElement.clientHeight; };

}

</script>

<script type="text/javascript">

document.write("X position " + Geometry.getWindowX() + "<br />");

document.write("Y position " + Geometry.getWindowY() + "<br />");

document.write("Height " + Geometry.getViewportHeight() + "<br />");

document.write("Width " + Geometry.getViewportWidth() + "<br />");

</script>

### Screen

<script>

window.onload = function () {

// определение высоты и ширины экрана пользователя.

alert("Width = " + screen.width + "; Height = " + screen.height);

}

</script>

### Window Open

<title></title>

<script type="text/javascript">

window.onload = function () {

document.getElementById("openButton").onclick = openButtonHandler;

document.getElementById("closeButton").onclick = closeButtonHandler;

var w;

function openButtonHandler() {

// первый параметр - html страница, которая будет загружена в новом окне.

// второй параметр - имя окна

// третий параметр - список дополнительных параметров (необязательный)

w = window.open("window.html", "Window Name", "width=400,height=400,status=yes,resizable=no");

}

function closeButtonHandler() {

w.close();

}

}

</script>

</head>

<body>

<input type="button" id="openButton" value="Открыть" />

<input type="button" id="closeButton" value="Закрыть" />

</body>

## String

### String typeof

<script type="text/javascript">

// Создание строковой переменной.

var a = 'my string'; // к объекту a нельзя добавлять новые свойства так как объект создавался через строковой литерал.

var b = "my string"; // к объекту b нельзя добавлять новые свойства так как объект создавался через строковой литерал.

var c = new String("my string");

var d = String("my string");

// указание юникод символов явным образом.

var str = "\u1234";

document.write(str);

document.write("typeof объекта, созданного через значение, взятое в двойные кавычки - " + typeof a);

document.write("<br />");

document.write("typeof для объекта, созданного через оператор new " + typeof c);

// Объекты, созданные через оператор new и через значение, взятое в двойные кавычки, будут выдавать разные значения

// при использовании оператора typeof.

// typeof для объекта, созданного через кавычки - string

// typeof для объекта, созданного через new - object

// Интерпретатор оба объекта будет воспринимать как строковое значение.

// но дополнительные свойства и методы добавить можно будет только в тот объект, который был создан через оператор new

### String Methods

<script type="text/javascript">

// split - Разбивает строку на подстроки по указанному разделителю и возвращает массив подстрок.

var str = "Строка для тестирования примера";

var substrs = str.split(" ");

document.write(substrs.join("<br />"));

// charAt - возвращает символ по указанной позиции.

var firstChar = str.charAt(0);

var lastChar = str.charAt(str.length - 1);

document.write("<br /> первый символ - " + firstChar + " последний символ - " + lastChar + "<br />");

// indexOf - возвращает индекс первого встречающегося в строке указанного символа.

str = "ab-cd-ef-gh-ij";

var firstDash = str.indexOf("-");

document.write("Индекс первого символа '-' в строке " + str + "= " + firstDash + "</br>");

// lastIndexOf - возвращает индекс последнего встречающегося в строке указанного символа.

str = "ab-cd-ef-gh-ij";

var lastDash = str.lastIndexOf("-");

document.write("Индекс последнего символа '-' в строке " + str + "= " + lastDash + "</br>");

// toLowerCase() - перевод всех символов в нижний регистр, toUpperCase() - перевод всех символов в верхний регистр.

str = "Hello World!!!";

str = str.toLowerCase();

document.write("toLowerCase " + str + "<br />");

str = str.toUpperCase();

document.write("toUpperCase " + str + "<br />");

// substring - возвращает строку между двумя указанными индексами, не включая индексы.

str = "Hello, world!";

substr = str.substring(6, 12);

document.write(substr);

// substr - возвращает строку начиная с указанного индекса, указанной длины.

str = "Hello, world!";

substr = str.substr(7, 5);

document.write("<br />" + substr);

## Regex

### String search

<title>Метод search()</title>

<script type="text/javascript">

// search() - возвращает индекс первого найденного символа подстроки по шаблону или -1 если подстрока не найдена

var text = "Строка 123 для тестирования примера.";

var pattern = /\d\d\d/;

var firstIndex = text.search(pattern);

document.write("Подстрока найдена по индексу " + firstIndex);

</script>

</head>

<body>

### String replace

<title>Метод replace</title>

<script type="text/javascript">

// replace() - производит операцию поиска с заменой.

var text = "Строка для 123тести456рования при789мера.";

var pattern = /\d\d\d/g; // g - глобальный поиск - все вхождения, а не только первое.

var result = text.replace(pattern, "");

document.write(result);

</script>

### String match

<script type="text/javascript">

// match() - принимает регулярное выражение и возвращает массив с результатами поиска.

var text = "1 плюс 2 равно 3";

var res = text.match(/\d/g); // возвращает массив ['1', '2', '3']

document.write(res);

var text2 = "Текст 123 для тестирования примера";

var res2 = text2.match(/\d\d\d/);

document.write("<br />res2[0] = " + res2[0]);

</script>

### String split regex

<title>Метод split</title>

<script type="text/javascript">

// split() - разбивает строку на массив подстрок в соответствии с регулярным выражением.

var text = "test-А123test-B123test-C111test-D567";

var res = text.split(/\d\d\d/g); // возвращает массив ['test-A', 'test-B', 'test-C', 'test-D']

document.write(res);

</script>

### Regex exec

<title>Метод exec()</title>

<script type="text/javascript">

// Метод exec работает точно так же как и метод match типа данных string.

// Но всегда возвращает только одно значение или null

// При глобальном поиске (флаг g) метод возвращает найденную подстроку и продолжает поиск далее по строке.

var test = "Строка №2 для тестирования примера с регулярными выражениями в файле 6\_RegExMethods\_exec.htm";

var pattern = /\d/g;

var res;

while ((res = pattern.exec(test)) != null) {

alert("Найдено " + res + " в позиции "

+ res.index + ". Следующий поиск начнется с индекса " + pattern.lastIndex);

}

</script>

### Regex test

<script type="text/javascript">

// Метод test возвращает true если метод exec возвращает не null;

var pattern = /\d\d\d/;

var input = "hello world";

if (pattern.test(input)) {

document.write("Строка " + input + " соответствует шаблону<br />");

}

else {

document.write("Строка " + input + " НЕ соответствует шаблону<br />");

}

input = "123";

if (pattern.test(input)) {

document.write("Строка " + input + " соответствует шаблону<br />");

}

else {

document.write("Строка " + input + " НЕ соответствует шаблону<br />");

}

</script>

### Regex symbols

<script type="text/javascript">

// Метасимволы.

// \w Любой алфавитно-цифровой символ в верхнем и нижнем регистре и символ подчеркивания(тоже самое, что и [a-zA-Z0-9\_]).

// \W Любой символ не являющийся подчеркиванием и не относящийся к алфавитно-цифровым

// \s любой пробельный символ (тоже самое, что [\r\n\f\v\t])

// \S Любой не пробельный символ (тоже самое, что [^\r\n\f\v\t])

// \d любая цифра (digit)

// \D любой не цифровой символ (тоже самое, что [^0-9])

// \v (вертикальная табуляция)

// \t (табуляция)

// \r (перевод каретки)

// \n (перевод строки)

// \f (перевод страницы)

// Наборы символов

// [RrGgBb] - Соответствие указанным символам

// [a-z] Соответствие символам английского алфавита в нижнем регистре

// [0-9] Соответствие указанным цифрам

// [^0-9] Соответствие с чем угодно кроме цифр заданного диапазона

// Квантификатор (Кванторы)

// + Один или несколько символов

// \* ноль и более символов

// ? ноль или одно вхождение

// Интервалы

// {3} точное количество совпадений

// {2,3} диапазон

// {3,} не менее

// {,3} не более

// Якорные символы

// ^ Начало строки

// $ Конец строки

// \b соответствует границе слова

// Флаги

// i выполнение поиска не чувствительного к регистру.

// g выполнение глобального поиска.

// m многострочный режим.

</script>

### Regex example

<title>Примеры регулярных выражений</title>

<script type="text/javascript">

// 21-12-1999

var pattern = /\d{1,2}-\d{1,2}-\d{4}/;

var text = "21-12-1999";

document.write(text + " -> " + pattern.test(text) + "<br />");

// +38 (063) 223-23-23

pattern = /\+38 \(\d{3}\) \d{3}-\d{2}-\d{2}/;

text = "+38 (063) 223-23-23";

document.write(text + " -> " + pattern.test(text) + "<br />");

// Иванов Иван Иванович

pattern = /[а-яА-Я]+ [а-яА-Я]+ [а-яА-Я]+/;

text = "Иванов Иван Иванович";

document.write(text + " -> " + pattern.test(text) + "<br />");

// ivanov.ivan@example.com

pattern = /\b[a-z0-9.\_]+@[a-z0-9.-]+\.[a-z]{2,4}\b/i;

text = "ivanov.ivan@example.com";

document.write(text + " -> " + pattern.test(text) + "<br />");

// hello.html

pattern = /\w+\.html/;

text = "hello.html";

document.write(text + " -> " + pattern.test(text) + "<br />");

// http://www.cbsarea.com или https://www.cbsarea.com

pattern = /https?:\/\/[\w\d:\.:\?\&]+/g;

text = "http://www.cbsarea.com или https://www.cbsarea.com blah-blah-blah http://www.google.com:80";

var res;

while ((res = pattern.exec(text)) != null) {

document.write("Найдено " + res + " по индексу " + res.index + "<br />");

}

</script>

</head>

# CSS and JavaScript

## Css

### Css styles

<title>Inline CSS</title>

</head>

<body>

<!--Применение правила стиля к элементу документа-->

<!--Стили задаться в виде разделенных точкой с запятой пар атрибутов, состоящих из имени и значения-->

<p style="color: Green; font-weight: bold; font-size: 25px;">Hello world</p>

</body>

### Css link file

<head>

<title>External CSS file</title>

<!--Подключение файла со стилями-->

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

</head>

<body>

<div>This is DIV element</div>

<p>This is P element</p>

<div class="test">This is DIV element with class test</div>

<p class="test2">This is P element with class test2</p>

<ul>

<li>Item 1</li>

<li>Item 2</li>

<li>Item 3</li>

</ul>

<div id="test3">This is DIV element with id=test3</div>

## CSS for DHTML

### Position:Absolute;Relative:Fixed

<title>Атрибут position</title>

<!--Атрибуты позиционирования-->

<style type="text/css">

body {

height: 2000px;

}

div {

border: solid Red 1px;

padding: 10px;

height: 100px;

width: 100px;

background-color: pink;

}

#divAbsolute {

/\*

absolute - позволяет задать абсолютное позиционирование элемента относительно содержащего его элемента.

Такие элементы позиционируются отдельно от остальных элементов. Абсолютное позиционирование элемента происходит либо

относительно тела документа, либо, если он вложен в другой абсолютно позиционируемый элемент, относительно этого элемента.

значение по умолчанию - static. Элементы со static позиционированием не могут быть перемещены с помощью свойств left и top.

\*/

position: absolute;

top: 30px;

left: 300px;

}

#divFixed {

/\*

fixed - позволяет зафиксировать элемент относительно окна браузера. Элементы с таким позиционированием

не прокручиваются с остальной частью документа, такие элементы не зависят от остальных

\*/

position: fixed;

top: 300px;

right: 0px;

}

#divRelative {

/\*

relative - элемент располагается в соответствии с нормальным потоком вывода, а затем его положение смещается

относительно его обычного положения в потоке.

\*/

position: relative;

top: 20px;

left: 40px;

border: dashed 2px blue;

height: 20px;

width: 20px;

}

</style>

</head>

<body>

<div id="divAbsolute">

Absolute

</div>

<div id="divFixed">

Fixed

</div>

<div>

<div id="divRelative"></div>

</div>

</body>

### Z-Index

<style type="text/css">

/\*

CSS атрибут z-index представляет целое число, которое определяет наложение элементов, указывая, какой из перекрывающихся элементов.

Порядок наложения можно определять только для смежных элементов (для дочерних элементов одного контейнера).

Чем больше значение z-index тем выше находиться элемент.

\*/

div

{

height: 50px;

width: 50px;

border:solid 1px black;

position:absolute;

}

</style>

</head>

<body>

<div style="left:10px;top:10px; background-color:Red; z-index:3"></div>

<div style="left:20px;top:20px; background-color:Green; z-index:2"></div>

<div style="left:30px;top:30px; background-color:Yellow; z-index:1"></div>

</body>

### Display and Visibility

<title>Visibility & Display</title>

<style type="text/css">

div {

height: 20px;

width: 100px;

border: black 1px solid;

background-color: green;

float: left;

margin: 8px;

text-align: center;

color: White;

}

</style>

</head>

<body>

<div>1</div>

<!--display - атрибут служит для задания варианта отображения элемента, определяя блочный это элемент, списочный и т.д.-->

<div style="display:none;">2</div>

<div>3</div>

<!--visibility - если значение hidden элемент не отображается, если значение visible - элемент виден.-->

<div style="visibility: hidden;">4</div>

<div>5</div>

</body

## DHTML

### Change color using JS

<title>Соглашения об именах CSS атрибутов в JavaScript</title>

<script type="text/javascript">

// Если имя CSS аттрибута содержит дефисы, имя свойства объекта образуеться путем удаления

// дефисов и перевода в верхний регистр буквы, непосредственно следующей за каждым из них.

window.onload = function () {

var button = document.getElementsByTagName("input")[0];

button.onclick = function () {

var div = document.getElementsByTagName("div")[0];

// CSS атрибут background-color

div.style.backgroundColor = "green";

}

}

</script>

</head>

<body>

<div style="width: 200px; height: 200px; background-color: Red;">

</div>

<input type="button" value="Сменить цвет" />

</body>

</html>

### Class name set using JS

<style type="text/css">

.class1 {

font-family: Tahoma;

color: Blue;

}

.class2 {

font-family: Courier New;

color: Red;

}

</style>

<script type="text/javascript">

var div;

window.onload = function () {

document.getElementById("first").onclick = firstButtonHandler;

document.getElementById("second").onclick = secondButtonHandler;

div = document.getElementById("testDiv");

}

function firstButtonHandler() {

// установка значения для атрибута class (присвоение CSS класса для элемента)

div.className = "class1";

}

function secondButtonHandler() {

div.className = "class2";

}

</script>

</head>

<body>

<input id="first" type="button" value="First" />

<input id="second" type="button" value="Second" />

<div id="testDiv">

Test Div

</div>

</body

### Computed styles

<style>

div {

width: 200px;

height: 200px;

}

div#first {

background-color: green;

}

</style>

<script>

window.onload = function () {

var div1 = document.getElementById("first"),

div2 = document.getElementById("second");

// Свойство style дает доступ к тем стилям, которые были заданы через JavaScript или непосредственно в атрибуте style в HTML разметке.

document.getElementById("btn1").onclick = function () {

alert("First div style - " + div1.style.backgroundColor); // прочитать значение backgroundColor не удастся.

}

// Для второго div стили вычисляются браузером на основе тех правил, которые задаются через CSS.

document.getElementById("btn2").onclick = function () {

alert("Second div style - " + div2.style.backgroundColor);

}

}

</script>

</head>

<body>

<!--Для данного блока цвет задается через CSS стили определенные в элементе style в начале файла.-->

<div id="first"></div>

<!--Для данного блока установлены inline стили-->

<div id="second" style="background-color: red"></div>

<button id="btn1">Get first styles</button>

<button id="btn2">Get second styles</button>

</body>

### Computed styles solve

<title>Computed Styles (Вычесляемые свойства)</title>

<style>

div {

width: 200px;

height: 200px;

}

div#first {

background-color: green;

}

</style>

<script>

window.onload = function () {

var div1 = document.getElementById("first"),

div2 = document.getElementById("second");

document.getElementById("btn1").onclick = function () {

alert("First div style - " + getStyle(div1, "backgroundColor"));

}

document.getElementById("btn2").onclick = function () {

alert("Second div style - " + getStyle(div2, "backgroundColor"));

}

// getComputedStyle - метод позволяет получить доступ к вычисляемым стилям.

function getStyle(element, styleName) {

if (element.currentStyle) { // получение доступа к Computed стилям для IE

return element.currentStyle[styleName];

}

else if (window.getComputedStyle) {

return window.getComputedStyle(element, null)[styleName];

}

}

}

</script>

</head>

<body>

<!--Для данного блока цвет задается через CSS стили определенные в элементе style в начале файла.-->

<div id="first"></div>

<!--Для данного блока установлены inline стили-->

<div id="second" style="background-color: red"></div>

<button id="btn1">Get first styles</button>

<button id="btn2">Get second styles</button>

</body>

### ToolTip

<title>Пример - ToolTip</title>

<style type="text/css">

.tooltip {

-webkit-box-shadow: 2px 2px 3px black;

box-shadow: 2px 2px 3px black;

background-color: whitesmoke;

border: solid black 1px;

padding: 5px;

font: bold 10pt sans-serif;

}

</style>

<script type="text/javascript">

function Tooltip() { // Конструктор для всплывающей подсказки

this.tooltip = document.createElement("div"); // Создание div для тени

this.tooltip.style.position = "absolute"; // Абсолютное позиционирование.

this.tooltip.style.visibility = "hidden";

this.tooltip.className = "tooltip";

}

// Установка содержимого для подсказки и отображение

Tooltip.prototype.show = function (text, x, y) {

this.tooltip.innerHTML = text; // Текст подсказки

this.tooltip.style.left = x + "px"; // Координаты

this.tooltip.style.top = y + "px";

this.tooltip.style.visibility = "visible";

// Добавление подсказки в документ если он еще не присутствует.

if (this.tooltip.parentNode != document.body)

document.body.appendChild(this.tooltip);

};

// Функция для скрытия подсказки

Tooltip.prototype.hide = function () {

this.tooltip.style.visibility = "hidden";

};

window.onload = function () {

var d = document.getElementById("test");

d.onmousemove = mouseMoveHandler;

d.onmouseout = mouseOutHandler;

}

var t = new Tooltip();

function mouseMoveHandler(e) {

if (!e) e = window.event;

t.show("This is tooltip text!!!", e.clientX + 10, e.clientY + 10);

}

function mouseOutHandler() {

t.hide();

}

</script>

</head>

<body>

<div id="test">

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem

Ipsum has been the industry's standard dummy text ever since the 1500s, when an

unknown printer took a galley of type and scrambled it to make a type specimen book.

It has survived not only five centuries, but also the leap into electronic typesetting,

remaining essentially unchanged. It was popularised in the 1960s with the release

of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop

publishing software like Aldus PageMaker including versions of Lorem Ipsum.

</div>

</body>

</html>

### Color Animation

<body>

<div id="rect">

</div>

<script type="text/javascript">

var e = document.getElementById("rect");

e.style.width = "100px";

e.style.height = "100px";

e.style.border = "solid black 5px";

var colors = ["white", "yellow", "orange", "red"]

var nextColor = 0;

// Вызов следующей функции каждые 500 миллисекунд для анимации цвета.

setInterval(function () {

e.style.backgroundColor = colors[nextColor++ % colors.length];

}, 500);

</script>

</body>

HSL Color Animation

<!-- HSL Hue, Saturation, Lightness (Intensity) - Цветовая модель, в которой цветовымикоординатами является цвет, насыщенность и яркость.

http://ru.wikipedia.org/wiki/HSL -->

<div id="rect">

</div>

<script type="text/javascript">

var e = document.getElementById("rect");

e.style.width = "100px";

e.style.height = "100px";

e.style.border = "solid black 1px";

var color = 0;

setInterval(function () {

color = color + 1 % 360;

e.style.backgroundColor = "hsl(" + color + ", 100%, 50%)"; // hsl(0, 100%, 50%)

}, 50);

</script>

### Move Animation <head>

<title>Простая анимация</title>

<script type="text/javascript">

var div1;

var leftPos = 0;

window.onload = function () {

div1 = document.getElementById("div1");

setInterval(animate, 10);

}

function animate() {

div1.style.left = leftPos + "px";

leftPos += 1;

}

</script>

</head>

<body>

<div id="div1"

style="height: 30px; width: 30px; background-color: Green; position: relative;">

</div>

</body>

# Event1

## DOM Level 0

### Event as attribute

<title>Обработчик события как атрибут</title>

<script type="text/javascript">

function validate() {

// проверка формы.

alert("validate");

}

</script>

</head>

<body>

<!--Обработчик события click-->

<input type="button" name="button1" value="Click me" onclick="alert('Hello')" />

<!--Если обработчик состоит из нескольких инструкций, они должны быть отделены ';'-->

<input type="button" name="button1" value="Click me" onclick="if (window.counter) counter++; else counter = 1; this.value = counter;" />

<form>

<input type="text" name="firstName" />

<br />

<!--Если обработчик возвращает false действие отменяется.-->

<input type="reset" onclick="return confirm('Вы уверены?')" />

<!--Обработчик установлен как функция из тега script-->

<input type="submit" onclick="validate()" />

</form

### Handler as property

window.onload = loadHandler;

function loadHandler() {

var p1 = document.getElementById("p1");

var p2 = document.getElementById("p2");

// обработичк как анонимная функция.

p1.onclick = function () { alert("hello") }; // Ненавязчивый JS

// обработчик как другая функция из этого сценария.

p2.onclick = clickHandler;

}

function clickHandler() {

alert("hello");

}

</script>

</head>

<body>

<p id="p1">

click me!

</p>

<p id="p2">

click me!

</p>

### This keyword

<title>Ключевое слово this в обработчике</title>

<script type="text/javascript">

function loadHandler() {

var e = document.getElementById("input1");

e.onclick = function () {

// так как дання функция является значением свойства onclick, ключевое слово this

// ссылается на тот элемент, который выдал событие.

this.value = "clicked!";

};

}

</script>

</head>

<body onload="loadHandler()">

<!--Ключевое слово this ссылается на элемент, в котором произошло событие-->

<input type="button" value="Click me!" onclick="this.value = 'clicked!'" />

<input type="button" id="input1" value="Click me!" />

</body>

### Change color

window.onload = function () {

var pArr = document.getElementsByTagName("p");

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

pArr[i].onclick = function(){

this.style.backgroundColor = "red";

}

}

}

## DOM Level 2

### AddEventListener

На один объект можна присваивать несколько событий

window.onload = function () {

// addEventListener(событие, обработичк, нужен\_перехват\_события)

var btn1 = document.getElementById("button1");

if (btn1.addEventListener)// проверка поддержки DOM Level 2

{

// Установка обработичка на событие click для кнопки с id button1

btn1.addEventListener("click",

function () {

alert("Обработчик события нажатия кнопки");

},

false);

}

}

</script>

</head>

<body>

<input id="button1" type="button" value="Кнопка" />

</body>

### RemoveEventHandler

window.onload = function () {

var btn1 = document.getElementById("button1");

var btn2 = document.getElementById("button2");

if (btn1.addEventListener)

{

btn1.addEventListener("click", handler, false);

btn2.addEventListener("click", removeHandler, false);

}

function handler() {

alert("Обработчик события нажатия кнопки");

}

function removeHandler() {

// удаление обработчика handler с кнопки с id button1

btn1.removeEventListener("click", handler, false);

}

}

</script>

</head>

<body>

<input id="button1" type="button" value="Кнопка" />

<input id="button2" type="button" value="Удалить обработчик" />

</body>

### EventRoutes

// В модели DOM Level 2 события распространяются по следующему принципу:

// На этапе ПЕРЕХВАТА событие распространяется от Document вниз до целевого элемента управления.

// Второй этап - СОБЫТИЕ В ЦЕЛЕВОМ УЗЛЕ.

// На третьем этапе событие ВСПЛЫВАЕТ от целевого элемента обратно вверх до корневого элемента.

// последний параметр в методе addEventListener -

// true - обработать событие на этапе ПЕРЕХВАТА

// false - обработать событие на этапе ВСПЛЫВАНИЕ

var isTunnel = false;

window.onload = function () {

var e1 = document.getElementById("first");

var e2 = document.getElementById("second");

var e3 = document.getElementById("third");

e1.addEventListener("click",

function () {

this.style.backgroundColor = "green";

alert("Обработчик DIV 1");

},

isTunnel);

e2.addEventListener("click",

function () {

this.style.backgroundColor = "yellow";

alert("Обработчик DIV 2");

},

isTunnel);

e3.addEventListener("click",

function () {

this.style.backgroundColor = "red";

alert("Обработчик DIV 3");

},

isTunnel);

}

</script>

<style>

div {

border: 3px solid gray;

padding: 50px;

background-color: white;

}

</style>

</head>

<body>

Первый

<div id="first">

Второй

<div id="second">

Третий

<div id="third">

</div>

</div>

</div>

### PreventDefault

<script>

window.addEventListener("load", function () {

document.forms[0].addEventListener("submit", function (e) {

if (document.getElementById("nameInput").value.length == 0)

{

// Если поле ввода пустое отменить действие по умолчанию, которое связанно с событием.

e.preventDefault();

alert("Введите имя");

}

}, false);

}, false);

</script>

</head>

<body>

<form action="/test.html" method="post">

Имя <input id="nameInput" type="text" name="name"/>

<input type="submit" value="Отправить" />

</form>

</body>

### StopPropagation

var isTunnel = false;

window.onload = function () {

var e1 = document.getElementById("first");

var e2 = document.getElementById("second");

var e3 = document.getElementById("third");

e1.addEventListener("click",

// e - параметр хранит объект с информацией о событии.

function (e) {

this.style.backgroundColor = "green";

alert("Обработчик DIV 1");

// прекращение дальнейшего маршрута события.

e.stopPropagation();

},

isTunnel);

### Input handler

<title>Пример использования событий DOM Level 2</title>

<script>

window.addEventListener("load", function () {

var guard = document.getElementById("guard");

guard.addEventListener("keypress", function (e) {

if (String.fromCharCode(e.charCode) != '1'

&& String.fromCharCode(e.charCode) != '0') {

// если нажатая кнопка не 0 или 1 отменить действие по умолчанию.

e.preventDefault();

}

}, true);

}, false);

</script>

</head>

<body>

<div id="guard">

<input type="text" value="" />

### Event target

<title>Пример использования событий DOM Level 2</title>

<style>

#wrapper p {

width:100px;

height:100px;

background-color:green;

float:left;

margin:10px;

}

</style>

<script>

window.addEventListener("load", function () {

var wrapper = document.getElementById("wrapper");

// e - объект события которое возникло в элементе wrapper или в его дочерних элементах

wrapper.addEventListener("click", function (e) {

// target - элемент, который был инициатором события.

e.target.style.backgroundColor = "red";

}, false);

}, false);

</script>

</head>

<body>

<div id="wrapper">

<p></p>

<p></p>

<p></p>

<p></p>

<p></p>

<p></p>

<p></p>

<p></p>

<p></p>

<p></p>

<div>

<p></p>

<p></p>

<p></p>

<p></p>

</div>

</div>

</body>

# Event2

## Events

### CrossBrowser events

<script>

window.onload = handler;

// параметр e будет передаваться в IE только при регистрации события через attachEvent

function handler(e){

if (!e) e = window.event; // получение информации о событии в IE

// тело обработчика события

}

function handler2(event) {

var e = event || window.event; // второй вариант.

}

</script>

</head>

### EventInterface DOM

<title>Интерфейс Event в моделе DOM</title>

<script type="text/javascript">

window.addEventListener("load", init, false);

function init() {

var button1 = document.getElementById("button1");

button1.addEventListener("click", handler, false);

}

function handler(e) {

// тип события

var message = "type = " + e.type + "\r\n";

// узел документа, в котором произошло событие, может не совпадать currentTarget

message += "target = " + e.target + "\r\n";

// узел чей обработчик события работает в данный момент. На этапах перехвата и всплывания значение свойства отличается от target.

message += "currentTarget = " + e.currentTarget + "\r\n";

// число указывающее этап. Одна из констант Event.CAPTURING\_PHASE=1, Event.AT\_TARGET=2, Event.BUBBLING\_PAHSE=3

message += "eventPhase = " + e.eventPhase + "\r\n";

// объект Date, указывающий, когда произошло событие.

message += "timeStamp = " + e.timeStamp + "\r\n";

// true если событие может всплывать по дереву элементов. События blur, focus, load, unload не высплывают.

message += "bubbles = " + e.bubbles + "\r\n";

// true если с этим событие связано действие по умолчанию и его можно отменить с помощью вызова метода preventDefault();

message += "cancelable = " + e.cancelable + "\r\n";

alert(message);

}

</script>

### Mouse Event

<title>MouseEvent Интерфейс</title>

<style type="text/css">

div {

background-color: #ceffff;

height: 100px;

width: 100px;

}

</style>

<script type="text/javascript">

window.addEventListener("load", init, false);

function init() {

var div = document.getElementsByTagName("div")[0];

div.addEventListener("mousemove", handler, false);

}

function handler(e) {

var message = "";

message += e.button + "<br />"; // 0 - левая клавиша, 1 - средняя клавиша, 2 - правая.

message += "Alt: " + e.altKey + "<br />"; // true если клавиша Alt зажата.

message += "Ctrl: " + e.ctrlKey + "<br />"; // true если клавиша Ctrl зажата.

message += "Shift: " + e.shiftKey + "<br />"; // true если клавиша Shift заэата.

message += "X: " + e.clientX + "<br />"; // координаты относительно левого верхнего угла документа.

message += "Y: " + e.clientY + "<br />"; // координаты относительно левого верхнего угла документа.

message += "screen X: " + e.screenX + "<br />"; // координаты относительно экрана.

message += "screen Y: " + e.screenY + "<br />"; // координаты относительно экрана.

document.getElementById("output").innerHTML = message;

}

</script>

</head>

<body>

<div></div>

<p id="output"></p>

</body

## Mouse Events

### Hover

title>События мыши</title>

<script>

window.onload = function () {

var p = document.getElementsByTagName("p")[0];

// курсор вошел в пределы элемента.

p.onmouseover = function () {

this.style.backgroundColor = "green";

}

// курсор вышел за пределы элемента.

p.onmouseout = function () {

this.style.backgroundColor = "yellow";

}

}

</script>

<style>

p {

background-color:red;

width:100px;

height:100px;

}

</style>

</head>

<body>

<p></p>

### Mouse move|down|up

window.onload = function () {

var div1 = document.getElementById("div1");

var div2 = document.getElementById("div2");

var div3 = document.getElementById("div3");

div1.onmousedown = function (e) {

if (!e) e = window.event;

// e.layerX - для Firefox

this.innerHTML = "Событие onmouse<b>down</b> X:" + (e.offsetX == undefined ? e.layerX : e.offsetX)

+ " Y:" + (e.offsetY == undefined ? e.layerY : e.offsetY);

}

div2.onmousemove = function (e) {

this.innerHTML = "Событие onmouse<b>move</b> X:" + e.offsetX + " Y:" + e.offsetY;

}

div3.onmouseup = function (e) {

this.innerHTML = "Событие onmouse<b>up</b> X:" + e.offsetX + " Y:" + e.offsetY;

}

}

</script>

<style>

div {

border: 1px solid black;

margin: 10px;

padding: 10px;

float: left;

height:400px;

width:400px;

}

#div1 {

background-color:red;

}

#div2 {

background-color:green;

}

#div3 {

background-color:blue;

}

</style>

</head>

<body>

<div id="div1"></div>

<div id="div2"></div>

<div id="div3"></div>

</body>

## Keyboard

Onekey up|down|press

<title>События клавиатуры</title>

<script>

window.onload = function () {

var main = document.getElementById("main");

main.onkeydown = function () {

print("onkeydown event");

};

main.onkeyup = function () {

print("onkeyup event");

};

main.onkeypress = function () {

print("onkeypress event");

};

function print(message) {

document.getElementById("output").innerHTML += message + "<br />";

}

}

</script>

</head>

<body id="main">

<p>Начните нажимать клавиши на клавиатуре. Обратите внимание какие события генерируются при нажатии на клавиши Shift, F1-F12, стрелочки и какие при нажатии на клавиши с печатными символами</p>

<p id="output"></p>

</body>

</html>

### Keyboard events

<script>

window.onload = function () {

var main = document.getElementById("main");

main.onkeydown = function () {

print("onkeydown event");

};

main.onkeyup = function () {

print("onkeyup event");

};

main.onkeypress = function () {

print("onkeypress event");

};

function print(message) {

document.getElementById("output").innerHTML += message + "<br />";

}

}

</script>

</head>

<body id="main">

<p>Начните нажимать клавиши на клавиатуре. Обратите внимание какие события генерируются при нажатии на клавиши Shift, F1-F12, стрелочки и какие при нажатии на клавиши с печатными символами</p>

<p id="output"></p>

</body>

### Onekeydown vs Onekeypress

<title>События клавиатуры</title>

<script>

window.onload = function () {

var main = document.getElementById("main");

// onkeydown - событие подходит для обработки функциональных клавиш.

main.onkeypress = function (e) { // снять комментарий

// main.onkeydown = function (e) {

clear();

print("alt = " + e.altKey);

print("ctrl = " + e.ctrlKey);

print("shift = " + e.shiftKey);

print("charCode = " + e.charCode); // код символа (передается при событии keypress)

print("keyCode = " + e.keyCode);

};

function print(message) {

document.getElementById("output").innerHTML += message + "<br />";

}

function clear() {

document.getElementById("output").innerHTML = "";

}

}

</script>

</head>

<body id="main">

<p id="output"></p>

</body>

### CharCode

window.onload = function () {

// Событие onkeypress используется для обработки клавиш с печатными символами.

document.getElementById("main").onkeypress = function (e) {

clear();

print("alt = " + e.altKey);

print("ctrl = " + e.ctrlKey);

print("shift = " + e.shiftKey);

print("charCode = " + String.fromCharCode(e.charCode)); // код символа (передается при событии keypress)

print("keyCode = " + e.keyCode); // код клавиши

}

function print(message) {

document.getElementById("output").innerHTML += message + "<br />";

}

function clear() {

document.getElementById("output").innerHTML = "";

}

}

</script>

</head>

<body id="main">

<p id="output"></p>

</body>

## Samples

### Input Filter

<style>

body {

font-family: Verdana, Geneva, 'DejaVu Sans', sans-serif;

}

label {

width: 150px;

float: left;

}

/\*стиль для span, которые расположены в разметке сразу за input\*/

input + span {

visibility: hidden;

color: red;

}

</style>

<script type="text/javascript">

window.addEventListener("load", init, false);

function init() {

var inputs = document.getElementsByTagName("input");

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

var e = inputs[i];

if (e.type != "text") continue; // пропускаем элементы input, которые не поля вводов

if (e.dataset.charsAllowed == undefined) continue; // пропускаем элементы у которых нет атрибута data-chars-allowed

// dataset - свойство, дающее возможность обратиться к атрибутам data-\*

// другой способ получения атрибута

//if(e.getAttribute("data-chars-allowed") == undefined)

e.addEventListener("keypress", keyFilter, false);

}

function keyFilter(e) {

if (!e) e = window.event;

// если нажата функциональная клавиша или введена управляющая последовательность (например, нажата клавиша Enter)

if (e.charCode == 0 || e.charCode < 32) return true;

var allowedText = e.target.dataset.charsAllowed;

var element = e.target.dataset.messageId; // элемент, который необходимо отобразить в случае ошибки

var symbol = String.fromCharCode(e.charCode).toLowerCase();

if (allowedText.search(symbol) == -1) {

if (element) {

element = document.getElementById(element);

element.style.visibility = "visible";

}

// отменяем ввод символа в поле ввода

e.preventDefault();

return false;

}

else {

if (element) {

element = document.getElementById(element);

element.style.visibility = "hidden";

}

return true;

}

}

}

</script>

</head>

<body>

<p>

<label>Имя</label>

<input type="text" name="firstName" data-chars-allowed="абвгдеёжзийклмнопрстуфхцчэюя" data-message-id="firstNameError" />

<span id="firstNameError">допускаются только буквы русского алфавита</span>

</p>

<p>

<label>Почтовый индекс</label>

<input type="text" name="zip" data-chars-allowed="1234567890" data-message-id="zipError" />

<span id="zipError">допускаются только цифры</span>

</p>

</body>

### Drag Element

<script type="text/javascript">

(function(){

window.addEventListener("load", init, false);

function init() {

var elem = document.getElementById("elem");

elem.addEventListener("mousedown", function (e) {

drag(this, e);

})

}

function drag(elementToDrag, event) {

// координаты мыши в начале перетаскивания.

var startX = event.clientX,

startY = event.clientY;

// начальные координаты элемента, который будет перемещаться.

var origX = elementToDrag.offsetLeft,

origY = elementToDrag.offsetTop;

// разница между координатами мыши и координатами перетаскиваемого элемента.

var deltaX = startX - origX,

deltaY = startY - origY;

// Регистрация событий mouseup и mousemove

document.addEventListener("mousemove", moveHandler, true);

document.addEventListener("mouseup", upHandler, true);

function moveHandler(e) {

if (!e) e = window.event;

// перемещаем элемент с учетом отступа от первоначального клика.

elementToDrag.style.left = (e.clientX - deltaX) + "px";

elementToDrag.style.top = (e.clientY - deltaY) + "px";

}

function upHandler(e) {

if (!e) e = window.event;

document.removeEventListener("mouseup", upHandler, true);

document.removeEventListener("mousemove", moveHandler, true);

}

}

})()

</script>

</head>

<body>

<div id="elem" style="position: absolute; left: 100px; top: 100px; height: 50px; width: 50px; background-color: Green;">

</div>

</body>

### Before Close

<title>Обработка закрытия окна браузера</title>

<script>

// Событие произойдет перед закрытием браузера.

window.onbeforeunload = function () {

return "Вы уверены, что хотите закрыть окно?";

}

</script>

</head>

<body>

# Forms

## Access to form

### Form elements

<script>

window.onload = function () {

var f = document.forms[0]; // получение первой формы на странице.

f.elements[0].value = "text"; // изменение первого по счету поля ввода в форме.

f.elements[1].value = "text";

f.elements[2].value = "text";

}

</script>

</head>

<body>

<form>

Имя <input type="text" value=" " /><br />

Фамилия<input type="text" value=" " /><br />

Возраст<input type="text" value=" " /><br />

<input type="button" value="Регистрация" />

</form>

</body>

### Get form by form name

<title>Работа с HTML формой</title>

<script>

window.onload = function () {

// получение формы по имени

var f = document.myform;

// получение элементов по имени.

f.fName.value = "text";

f.lName.value = "text";

f.age.value = "text";

}

</script>

</head>

<body>

<form name="myform">

Имя <input type="text" value=" " name="fName" /><br />

Фамилия<input type="text" value=" " name="lName" /><br />

Возраст<input type="text" value=" " name="age" /><br />

<input type="button" value="Регистрация" />

</form>

</body>

## Form elements

### All elements type

<form>

<!--Кнопка, основное событие onclick-->

<input type="button" name="button1" value="Burron1" />

<br />

<!--Флажок, основное событие onclick-->

<input type="checkbox" checked="checked" name="checbox1" /> Check Box 1

<br />

<!--Поле для ввода файла загружаймого на сервер, основное событие onchange-->

<input type="file" name="file1" value="" />

<br />

<!--Данные сохраняемые вместе с формой но невидимые пользователю.-->

<input type="hidden" name="hidden1" value="" />

<br />

<!--Поле для ввода пароля-->

<input type="password" name="password1" value=" " />

<br />

<!--Переключатель-->

Radio 1<input type="radio" checked="checked" name="group1" value="on" />

Radio 2<input type="radio" name="group1" value="off" />

<br />

<!--Кнопка очищающая значения формы.-->

<input type="reset" name="reset1" value="Reset" />

<br />

<br />

<!--Список в котором может быть выбран один элемент-->

<select name="select1">

<option value="value">text</option>

<option value="value">text</option>

<option value="value">text</option>

</select>

<br />

<br />

<!--Список со множественным выбором-->

<select multiple="multiple" name="multipleSelect1">

<option value="value">text</option>

<option value="value">text</option>

<option value="value">text</option>

</select>

<br />

<br />

<!--Кнопка для передачи данных формы-->

<input type="submit" name="submit1" value="Submit" />

<br />

<!--Однострочное поле ввода-->

<input type="text" name="text1" value=" " />

<br />

<!--Многострочное поле ввода-->

<textarea name="textArea1" rows="4" cols="10"></textarea>

</form>

### Elements properties

<script type="text/javascript">

function getAllElements() {

var message = "";

for (var i = 0; i < form1.elements.length; i++) {

var fe = form1.elements[i];

message += "value: " + fe.value + "\r\n"; // значение элемента

message += "form: " + fe.form.name + "\r\n"; // ссылка на форму, в которой элемент находится

message += "type: " + fe.type + "\r\n"; // тип элемента управления

message += "name: " + fe.name + "\r\n"; // имя элемента управления

message += "----------------------------------\r\n";

}

alert(message);

}

</script>

</head>

<body>

<form name="form1">

First Name

<input type="text" name="firsNameEdit" />

<br />

Last Name

<input type="text" name="lastNameEdit" />

<br />

<input type="button" value="Get All Elements" onclick="getAllElements()" />

</form>

</body>

### Element OnChange

<script>

window.onload = function () {

for (var i = 0; i < document.forms[0].elements.length; i++) {

// обработчик на событие смены значения в элементе

document.forms[0].elements[i].onchange = function () {

document.getElementById("output").innerHTML = "Значение изменилось в " + this.name;

}

}

}

</script>

</head>

<body>

<form>

First <input type="text" name="first" value="" /> <br />

Second <input type="text" name="second" value="" /> <br />

Third <input type="text" name="third" value="" /> <br />

</form>

<p id="output"></p

### Elements OnFocus

<script>

window.onload = function () {

for (var i = 0; i < document.forms[0].elements.length; i++) {

// cобытие на получение фокуса.

document.forms[0].elements[i].onfocus = function () {

this.style.border = "1px solid red";

}

// событие на потерю фокуса.

document.forms[0].elements[i].onblur = function () {

this.style.border = "1px dotted green";

}

}

}

</script>

</head>

<body>

<form>

First <input type="text" name="first" value="" /> <br />

Second <input type="text" name="second" value="" /> <br />

Third <input type="text" name="third" value="" /> <br />

</form>

### Watermark

<title>Watermark</title>

<script>

window.onload = function () {

var watermark = "Введите логин";

var input = document.forms[0].login;

initInput();

input.onfocus = function () {

if (input.value == watermark)

{

input.value = "";

input.style.color = "black";

input.style.fontStyle = "normal";

}

}

input.onblur = function () {

if (input.value == "")

{

initInput();

}

}

function initInput()

{

input.value = watermark;

input.style.color = "gray";

input.style.fontStyle = "italic";

}

}

</script>

</head>

<body>

<form>

Логин

<input type="text" name="login" />

<br />

Email

<input type="text" name="email" placeholder="Enter your email" />

</form>

## Form validation

### Validation inputs length

<script>

window.addEventListener("load", function () {

// обработка события отправки запроса

document.forms[0].addEventListener("submit", function (e) {

var isValid = true;

var login = document.getElementsByName("login")[0],

password = document.getElementsByName("password")[0];

// если условие сработает значение в форме будет считаться не правильным.

if (login.value.length == 0) {

isValid = false;

}

if (password.value.length == 0) {

isValid = false;

}

// В случае если форма заполнена не правильно - отображаем сообщение об ошибке

// и предотвращаем отправку запроса с помощью вызова preventDefault()

if (!isValid) {

e.preventDefault();

alert("Вcе поля вводов должны быть заполнены");

}

});

})

</script>

</head>

<body>

<form>

Login

<input type="text" name="login" />

Password

<input type="password" name="password" />

<input type="submit" />

</form>

</body>

### Form validation

// регистрация события загрузки документа.

if (window.addEventListener) window.addEventListener("load", init, false);

else if (window.attachEvent) window.attachEvent("onload", init);

// регистрация обработчиков событий элементов формы.

function init() {

form1.userName.onchange = nameOnChange;

form1.email.onchange = emailOnChange;

form1.zip.onchange = zipcodeOnChange;

form1.onsubmit = onsubmiHandler;

}

// метод проверки значения в элементе по регулярному выражению.

function validate(elem, pattern) {

var res = elem.value.search(pattern);

if (res == -1) elem.className = "invalid"; // установка CSS класса

else elem.className = "valid";

}

// обработчики событий изменения текста в окне.

function nameOnChange() {

var pattern = /\S/;

validate(this, pattern);

}

function emailOnChange() {

var pattern = /\b[a-z0-9.\_]+@[a-z0-9.-]+\.[a-z]{2,4}\b/i;

validate(this, pattern);

}

function zipcodeOnChange() {

var pattern = /\d{5}/;

validate(this, pattern);

}

// событие при отправке формы на сервер.

function onsubmiHandler() {

var invalid = false;

for (var i = 0; i < form1.elements.length; ++i) {

var e = form1.elements[i];

// проверка типа элемента и наличия обработчика события onchange.

if (e.type == "text" && e.onchange) {

e.onchange(); // запуск события onchanhe

if (e.className == "invalid") invalid = true;

}

}

if (invalid) {

alert("Допущены ошибки при заполнении формы.");

return false; // отмена отправки формы.

}

}

<head>

<title>Пример проверки достоверности ввода</title>

<script type="text/javascript" src="Validate\_001.js"></script>

<style type="text/css">

input.invalid {

background-color: Pink;

}

input.valid {

background-color: LightGreen;

}

</style>

</head>

<body>

<form name="form1">

Name:

<input type="text" name="userName" /><br />

email:

<input type="text" name="email" /><br />

zipcode:

<input type="text" name="zip" /><br />

<input type="submit" />

</form>

</body>

</html>

### Form validation 2

// регистрация события загрузки документа.

if (window.addEventListener) window.addEventListener("load", init, false);

// установка обработчиков для форм и элементов форм.

function init() {

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

var form = document.forms[i];

var formValidation = false;

for (var j = 0; j < form.elements.length; j++) {

var e = form.elements[j];

// пропускаем все что не поле ввода.

if (e.type != "text") {

continue;

}

// проверка имеются ли атрибуты требующие проверки.

var pattern = e.getAttribute("data-val");

if (pattern) {

e.onchange = validateInput; // обработчик на изменение.

formValidation = true; // форма требует проверку.

}

}

if (formValidation) {

form.onsubmit = validateForm; // установка обработчика для формы на submit

}

}

}

// обработчик на изменение содержимого полей ввода.

function validateInput() {

var pattern = this.dataset.val,

msg = this.dataset.valMsg,

msgId = this.dataset.valMsgId,

value = this.value;

var res = value.search(pattern);

if (res == -1) {

document.getElementById(msgId).innerHTML = msg;

this.className = "error";

}

else {

document.getElementById(msgId).innerHTML = "";

this.className = "valid";

}

}

// обработчик на submit формы.

function validateForm() {

var invalid = false;

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

var e = this.elements[i];

if (e.type == "text" && e.onchange != null) {

e.onchange();

if (e.className == "error") invalid = true;

}

}

if (invalid) {

alert("Допущены ошибки при заполнении формы.");

return false;

}

}

### Html5 validation

<title>HTML 5 - Пример проверки достоверности ввода</title>

</head>

<body>

<form>

Login

<!-- required="required" - поле обязательно для заполнения -->

<input type="text" name="login" required="required" />

<br />

Password

<input type="password" name="password" required="required" /><br />

Email

<!-- type="email" - поле должно содержать данные в формате email адреса -->

<input type="email" name="emailAddress" value="" /><br />

Phone

<!-- pattern="\d{3}-\d{2}-\d{2}" - поле должно содержать значение которое соответствует формату заданному через регулярное выражение -->

<input type="text" pattern="\d{3}-\d{2}-\d{2}" title="Формат телефона 000-00-00" name="emailAddress" value="" /><br />

<input type="submit" />

</form>

## Dates

### Date

<script type="text/javascript" language="JavaScript">

var now = new Date(); // Объект Date с текущей датой и временем.

document.write("<strong>текущая дата= </strong>" + now + "<br />");

var now = new Date(5000); // Параметр - количество миллисекунд между нужной датой и полночью 1 января 1970

document.write("<strong>date1 = </strong>" + now + "<br />");

var date = new Date("December 25, 1995 21:46:10");

document.write("<strong>date2 = </strong>" + date + "<br />");

var date2 = new Date("03/09/2000"); // полночь 9 марта 2000 года

document.write("<strong>date3 = </strong>" + date2 + "<br />");

// Date(год, месяц, день, часы, минуты, секунды, мс)

var date3 = new Date(2000, 2, 9); // полночь 9 марта 2000 года

document.write("<strong>date4 = </strong>" + date3 + "<br />");

</script>

### Date get properties

<script type="text/javascript" language="JavaScript">

var now = new Date(); // Объект Date с текущей датой и временем.

document.write("<strong>текущая дата= </strong>" + now + "<br />");

var now = new Date(5000); // Параметр - количество миллисекунд между нужной датой и полночью 1 января 1970

document.write("<strong>date1 = </strong>" + now + "<br />");

var date = new Date("December 25, 1995 21:46:10");

document.write("<strong>date2 = </strong>" + date + "<br />");

var date2 = new Date("03/09/2000"); // полночь 9 марта 2000 года

document.write("<strong>date3 = </strong>" + date2 + "<br />");

// Date(год, месяц, день, часы, минуты, секунды, мс)

var date3 = new Date(2000, 2, 9); // полночь 9 марта 2000 года

document.write("<strong>date4 = </strong>" + date3 + "<br />");

</script>

### Date set properties

<script type="text/javascript" language="JavaScript">

var date1 = new Date();

date1.setDate(27); // устанавливает новую дату (число месяца) в объекте

date1.setMonth(4); // устанавливает месяц

date1.setFullYear(2005); // устанавливает год; параметр функции должен быть 4-значным

date1.setHours(5); // устанавливает час. Если значение аргумента > 23, дата увеличивается на 1, а часы принимают значение аргумент-24

date1.setMinutes(5); // устанавливает минуты. Если значение аргумента > 59, часы увеличиваются на 1, а минуты принимают значение аргумент-60

date1.setSeconds(5); // устанавливает секунды. Если значение аргумента > 59, минуты увеличиваются на 1, а секунды принимают значение аргумент-60

date1.setMilliseconds(555); // устанавливает миллисекунды. Если значение аргумента > 999, секунды увеличиваются на 1, а миллисекунды принимают значение аргумент-1000

document.writeln("<strong>date1 = </strong>" + date1 + "<br />");

var date2 = new Date();

// устанавливает значение объекта Date. Значение аргумента setTime определяет количество миллисекунд, между требуемой датой и полночью 1 января 1970 года.

date2.setTime(1000000000000);

document.writeln("<strong>date2 = </strong>" + date2);

</script>

# Cookies

## Cookies

### SavingCookies

<script type="text/javascript">

function setCookie() {

// Что бы связать временное значение cookie файла с текущим документом. достаточно выполнить следующую операцию

document.cookie = "info=123"

// Значение Cookie не могут содержать точку с запятой, запятые или символы разделители. Перед сохранением значение в cookie желательно вызвать

// функцию encodeURIComponent() при чтении значения надо вызвать decodeURIComponent()

// записанный таким способом cookie сохраняются в текущем сеансе браузера но удаляются после его закрытия.

}

function getCookie() {

alert(document.cookie);

}

</script>

</head>

<body>

<input type="button" value="Set cookie" onclick="setCookie()" />

<input type="button" value="Get cookie" onclick="getCookie()" />

</body>

### Time of storing cookies

<script type="text/javascript">

function setCookie() {

// следующий cookie будет хранится браузером на проятжении 1 недели (60 \* 60 \* 24 \* 7).

document.cookie = "info=123; max-age=604800";

// Существует возможность указать время жизни cookie через устаревшее свойство expires

var nextYear = new Date();

nextYear.setFullYear(nextYear.getFullYear() + 1);

document.cookie = "name=Ivan Ivanv; expires=" + nextYear.toGMTString();

}

function getCookie() {

alert(document.cookie);

}

</script>

</head>

<body>

<input type="button" value="Set cookie" onclick="setCookie()" />

<input type="button" value="Get cookie" onclick="getCookie()" />

</body>

### Change cookies

<title>Установка времени жизни для cookie</title>

<script type="text/javascript">

function setCookie() {

document.cookie = "lang=HTML; max-age=120";

}

function getCookie() {

alert(document.cookie);

}

function changeCookie() {

document.cookie = "lang=JavaScript; max-age=120";

}

function deleteCookie() {

document.cookie = "lang=JavaScript; max-age=0";

}

</script>

</head>

<body>

<input type="button" value="Set cookie" onclick="setCookie()" />

<input type="button" value="Get cookie" onclick="getCookie()" />

<input type="button" value="Change cookie" onclick="changeCookie()" />

<input type="button" value="Delete cookie" onclick="deleteCookie()" />

</body>

</html>

### Url encode|decode

<script>

window.onload = function () {

// name=a;b;c; max-age=1000;

function $(id) {

return document.getElementById(id);

}

$("encodeBtn").onclick = function () {

// URL кодирование данных

var res = encodeURIComponent($("forEncode").value);

$("encodedOutput").innerHTML = res;

}

$("decodeBtn").onclick = function () {

// URL декодирование данных

var res = decodeURIComponent($("forDecode").value);

$("decodedOutput").innerHTML = res;

}

}

</script>

</head>

<body>

Текст для кодирования <input id="forEncode" /><input id="encodeBtn" type="button" value="Кодировать" />

<p id="encodedOutput"></p>

Текст для декодирования <input id="forDecode" /><input id="decodeBtn" type="button" value="Декодировать" />

<p id="decodedOutput"></p>

</body>

### Read cookies

<script type="text/javascript">

function setCookie() {

// Что бы связать временное значение cookie файла с текущим документом. достаточно выполнить следующую операцию

document.cookie = "version=" + encodeURIComponent(document.lastModified);

// Cookie не могут содержать точку с запятой, запятые или символы разделители. Перед сохранением значение в cookie желательно вызвать

// функцию encodeURIComponent() при чтении значения надо вызвать decodeURIComponent()

// записанный таким способом cookie сохраняются в текущем сеансе браузера но удаляются после его закрытия.

}

function getCookie() {

alert(findCookieValue("version"));

}

// version=123; test=a1; test2=a2

function findCookieValue(cookieName) {

var allcookies = document.cookie;

var pos = allcookies.indexOf(cookieName + "="); // version=

// Если cookie с указанным именем найден, извлечь его значения.

if (pos != -1) {

var start = pos + cookieName.length + 1;

var end = allcookies.indexOf(";", start);

if (end == -1) {

end = allcookies.length;

}

var value = allcookies.substring(start, end);

return decodeURIComponent(value);

}

}

</script>

</head>

<body>

<input type="button" value="Set cookie" onclick="setCookie()" />

<input type="button" value="Get cookie" onclick="getCookie()" />

</body>

### Cookies example

window.onload = function () {

function $(id) {

return document.getElementById(id);

}

var f = document.forms[0];

$("saveBtn").onclick = function () {

var cookieStr = "";

cookieStr += f.cookieName.value + "=" + f.cookieValue.value + ";";

cookieStr += "max-age=" + f.cookieAge.value + ";";

document.cookie = cookieStr;

}

$("readAll").onclick = function () {

alert(document.cookie);

}

$("readSpec").onclick = function () {

alert(findCookieValue(f.nameToRead.value));

}

function findCookieValue(cookieName) {

var allcookies = document.cookie;

var pos = allcookies.indexOf(cookieName + "=");

// Если cookie с указанным именем найден, извлечь его значения.

if (pos != -1) {

var start = pos + cookieName.length + 1;

var end = allcookies.indexOf(";", start);

if (end == -1) end = allcookies.length;

var value = allcookies.substring(start, end);

value = decodeURIComponent(value);

return value;

}

}

}

</script>

</head>

<body>

<form>

Имя

<input type="text" name="cookieName" value="" /><br />

Значение

<input type="text" name="cookieValue" value="" /><br />

Время жизни

<input type="text" name="cookieAge" value="" /><br />

<input id="saveBtn" type="button" value="Сохранить" />

<br />

<br />

<br />

<input id="readAll" type="button" value="Прочитать все cookies" />

<br />

Имя для чтения

<input type="text" name="nameToRead" />

<input id="readSpec" type="button" value="Прочитать" />

</form>

### Cookies example remember color

window.addEventListener("load", function () {

var savedColor = findCookieValue("page-color");

if (savedColor) {

document.body.style.backgroundColor = savedColor;

}

document.getElementById("saveButton").addEventListener("click", function () {

var selectedColor = getCheckedRadioId("color")

document.cookie = "page-color=" + encodeURIComponent(selectedColor) + ";max-age=" + (60 \* 60);

document.body.style.backgroundColor = selectedColor;

});

function getCheckedRadioId(name) {

var elements = document.getElementsByName(name);

for (var i = 0, len = elements.length; i < len; ++i)

if (elements[i].checked) return elements[i].value;

}

function findCookieValue(cookieName) {

var allcookies = document.cookie;

var pos = allcookies.indexOf(cookieName + "=");

// Если cookie с указанным именем найден, извлечь его значения.

if (pos != -1) {

var start = pos + cookieName.length + 1;

var end = allcookies.indexOf(";", start);

if (end == -1) end = allcookies.length;

var value = allcookies.substring(start, end);

value = decodeURIComponent(value);

return value;

}

}

});

</script>

</head>

<body>

<form>

<input type="radio" name="color" value="#ffd800" /> Orange

<input type="radio" name="color" value="#5ade54" /> Green

<input id="saveButton" type="button" value="Сохранить" />

</form>

## Storage mechanism

### Session storage

<title>Сохранение и извлечение данных sessionStorage</title>

<script type="text/javascript">

function init() {

var output = document.getElementById("output");

var btn1 = document.getElementById("btn1");

var btn2 = document.getElementById("btn2");

btn1.onclick = function () {

window.sessionStorage.setItem("key1", "Test value");

// или

window.sessionStorage.key2 = "Test value 2";

// или

window.sessionStorage["key3"] = "Test value 3";

}

btn2.onclick = function () {

output.innerHTML = window.sessionStorage.getItem("key1");

output.innerHTML += "<br />";

output.innerHTML += window.sessionStorage.key2;

output.innerHTML += "<br />";

output.innerHTML += window.sessionStorage["key3"];

}

}

window.addEventListener("load", init, false);

</script>

</head>

<body>

<p>

Значения существуют до тех пор, пока открыто окно или вкладка, в котором они сохранены

</p>

<p>

Значения видимы только в окне или вкладке, в которой они были созданы.

</p>

<input id="btn1" type="button" value="Запись в sessionStorage" />

<input id="btn2" type="button" value="Чтение из sessionStorage" />

<div id="output">

</div>

### Local storage

<script type="text/javascript">

function init() {

var output = document.getElementById("output");

var btn1 = document.getElementById("btn1");

var btn2 = document.getElementById("btn2");

btn1.onclick = function () {

window.localStorage.setItem("key1", "Test value");

// или

window.localStorage.key2 = "Test value 2";

// или

window.localStorage["key3"] = "Test value 3";

}

btn2.onclick = function () {

output.innerHTML = window.localStorage.getItem("key1");

output.innerHTML += "<br />";

output.innerHTML += window.localStorage.key2;

output.innerHTML += "<br />";

output.innerHTML += window.localStorage["key3"];

}

}

window.addEventListener("load", init, false);

</script>

</head>

<body>

<p>

Значения в localStorage существуют не зависимо от закрытия вкладки или браузера

</p>

<p>

Значения совместно используются всеми окнами и вкладками, выполняющимися на одном

и том же источнике.

</p>

<input id="btn1" type="button" value="Запись в localStorage" />

<input id="btn2" type="button" value="Чтение из localStorage" />

<div id="output">

</div>

### Local storage remember color

<script>

window.addEventListener("load", function () {

if (window.localStorage.pageColor) {

document.body.style.backgroundColor = window.localStorage.pageColor;

}

document.getElementById("saveButton").addEventListener("click", function () {

var selectedColor = getCheckedRadioId("color")

window.localStorage.pageColor = selectedColor;

document.body.style.backgroundColor = selectedColor;

});

function getCheckedRadioId(name) {

var elements = document.getElementsByName(name);

for (var i = 0, len = elements.length; i < len; ++i)

if (elements[i].checked) return elements[i].value;

}

});

</script>

</head>

<body>

<form>

<input type="radio" name="color" value="#ffd800" /> Orange

<input type="radio" name="color" value="#5ade54" /> Green

<input id="saveButton" type="button" value="Сохранить" />

</form>

# Graphics

## Images

### Get images

<title>Работа с изображениями</title>

<script>

window.onload = function () {

var img1 = document.getElementById("logoImg"); // получение по id.

var img2 = document.images.logo; // получение по name (первый способ).

var img3 = document.images["logo"]; // получение по name (второй способ).

var img4 = document.logo; // если name уникальный для всего документа.

alert(img1 + ", " + img2 + ", " + img3 + ", " + img4);

}

</script>

</head>

<body>

<img id="logoImg" name="logo" src="images/logo.png" alt="CyberBionic Systematics" />

</body>

### Change image

<script>

window.addEventListener("load", function () {

var flag;

document.getElementById("changeBtn").addEventListener("click", function () {

if (flag) {

//смена изображения, приводит к обращению на сервер если изображение на загружено и не закешировано

document.getElementById("imgBtn").src = "images/button.jpg";

}

else {

document.getElementById("imgBtn").src = "images/button-hover.jpg";

}

flag = !flag;

});

})

</script>

</head>

<body>

<img id="imgBtn" src="images/button.jpg" alt="Button" />

<button id="changeBtn">Change</button>

</body>

### Image caching

<body>

<script>

var images = []

function preload() {

for (i = 0; i < preload.arguments.length; i++) {

images[i] = new Image()

images[i].src = preload.arguments[i]

}

}

preload(

"images/frame1.jpg",

"images/frame2.jpg",

"images/frame3.jpg",

"images/frame4.jpg",

"images/frame5.jpg"

)

</script

### Change image by mouseover

<script>

window.onload = function () {

addRollover("btn", "images/button-hover.jpg");

}

function addRollover(id, path) {

var e = document.getElementById(id);

// если элемент не img пропускаем, завершаем функцию.

if (e.tagName.toLowerCase() != "img") return;

(new Image()).src = path;

var basePath = e.src;

e.addEventListener("mouseover", function () { this.src = path; })

e.addEventListener("mouseout", function () { this.src = basePath; })

}

</script>

</head>

<body>

<img id="btn" src="images/button.jpg" />

</body>

### Image animation

<script src="Animation.js"></script>

<script>

window.addEventListener("load", function () {

var animation = new Animation(

"animCanvas",

5,

["images/frame1.jpg",

"images/frame2.jpg",

"images/frame3.jpg",

"images/frame4.jpg",

"images/frame5.jpg",

"images/frame6.jpg"]);

document.getElementById("startButton").addEventListener("click", function () {

animation.start();

});

document.getElementById("stopButton").addEventListener("click", function () {

animation.stop();

});

});

</script>

</head>

<body>

<img id="animCanvas" src="images/frame1.jpg" />

<br />

<input id="startButton" type="button" value="Start" />

<input id="stopButton" type="button" value="Stop" />

</body>

function Animation(id, fps, urls) {

this.imageId = id; // id элемента img

this.fps = 1000 / fps; // скорость смены кадров в секунду

this.imgElement = null; // элемент img в котором будет воспроизводиться анимация

this.frames = new Array(urls.length); // массив с загруженными изображениями.

this.loadedFrames = 0; // количество загруженных изображений

this.isLoaded = false; // true если все изображения загружены

this.currentFrame = -1; // текущее изображение, которое должно отображаться из массива frames

this.timer = null; // объект таймера, который будет создан через setInterval

this.startOnLoad = false; // true если анимация должна запуститься сразу после загрузки всех изображений

// загрузка изображений

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

this.frames[i] = new Image();

this.frames[i].onload = countLoaded; // после загрузки изображения увеличить счетчик loadedFrames

this.frames[i].src = urls[i];

}

var current = this;

function countLoaded() {

current.loadedFrames++;

if (current.loadedFrames == urls.length) {

current.isLoaded = true; // все изображения загружены

if (current.startOnLoad) current.start(); // запустить анимацию если startOnLoad true

}

}

// нижнее подчеркивание в имени метода указывает на то что он предназначен только для внутреннего использования.

// пользователь объекта Animation не должен запускать этот метод самостоятельно.

this.\_nextFrame = function () {

// при запуске метода через setInterval this будет ссылаться на window

// поэтому вместо ссылки this используется переменная current

current.currentFrame = (current.currentFrame + 1) % current.frames.length;

current.imgElement.src = current.frames[current.currentFrame].src;

}

}

// метод для запуска анимации

Animation.prototype.start = function () {

if (this.timer) return;

if (!this.isLoaded) {

this.startOnLoad = true;

}

else {

if (!this.imgElement) {

this.imgElement = document.getElementById(this.imageId);

}

this.\_nextFrame();

this.timer = setInterval(this.\_nextFrame, this.fps);

}

}

// метод для остановки анимации

Animation.prototype.stop = function () {

if (this.timer) clearInterval(this.timer);

this.timer = null;

}

### Chart

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<title>Создание гистограммы</title>

<script src="BarChart.js"></script>

<script>

window.addEventListener("load", function () {

var chart = createBarChart([30, 6, 88, 10, 2, 45, 100, 150],

600,

300,

"green");

var container = document.getElementById("chartContainer");

container.appendChild(chart);

});

</script>

</head>

<body>

<div id="chartContainer"></div>

</body>

</html>

function createBarChart(data, width, height, color) {

// создаем контейнер для диаграммы

var chart = document.createElement("div");

chart.style.width = width + "px";

chart.style.height = height + "px";

chart.style.position = "relative";

// находим максимальное значение в массиве данных

var max = Number.NEGATIVE\_INFINITY;

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

if (max < data[i]) max = data[i];

}

var scale = height / max;

var barWidth = Math.floor(width / data.length);

// создаем отдельный элемент диаграммы

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

var bar = document.createElement("div");

bar.style.height = data[i] \* scale + "px";

bar.style.width = barWidth - 4 + "px";

bar.style.position = "absolute";

bar.style.margin = "4px";

bar.style.bottom = "0px";

bar.style.left = barWidth \* i + "px";

bar.style.backgroundColor = color;

chart.appendChild(bar);

}

return chart;

}

## SVG

### Shapes

<body>

<!--прямоугольник-->

<svg xmlns="http://www.w3.org/2000/svg" version="1.1" style="width:500px; height:200px; display:block;">

<rect width="300"

height="100"

style="fill: rgb(0,0,255); stroke-width: 1; stroke: rgb(0,0,0)" />

</svg>

<!--круг-->

<svg xmlns="http://www.w3.org/2000/svg" version="1.1" style="width:500px; height:200px; display:block;">

<circle cx="100"

cy="50"

r="40"

stroke="black"

stroke-width="2" fill="red" />

</svg>

<!--эллипс-->

<svg xmlns="http://www.w3.org/2000/svg" version="1.1" style="width:500px; height:200px; display:block;">

<ellipse cx="300"

cy="80"

rx="100"

ry="50"

style="fill: yellow; stroke: purple; stroke-width: 2" />

</svg>

<!--Линия-->

<svg xmlns="http://www.w3.org/2000/svg" version="1.1" style="width:500px; height:200px; display:block;">

<line x1="0"

y1="0"

x2="200"

y2="200"

style="stroke: rgb(255,0,0); stroke-width: 2" />

</svg>

</body>

Filter

<body>

<!--Размытие-->

<svg xmlns="http://www.w3.org/2000/svg" version="1.1">

<defs>

<filter id="f1" x="0" y="0">

<feGaussianBlur stdDeviation="15" />

</filter>

</defs>

<rect width="90"

height="90"

stroke="green"

stroke-width="3"

fill="yellow"

filter="url(#f1)" />

</svg>

</body>

### Gradient

<body>

<svg xmlns="http://www.w3.org/2000/svg" version="1.1">

<defs>

<linearGradient id="grad1"

x1="0%" y1="0%"

x2="100%" y2="0%">

<stop offset="0%"

style="stop-color: rgb(255,255,0); stop-opacity: 1" />

<stop offset="100%"

style="stop-color: rgb(255,0,0); stop-opacity: 1" />

</linearGradient>

</defs>

<ellipse cx="200"

cy="70"

rx="85"

ry="55"

fill="url(#grad1)" />

</svg>

</body>

### SVG Chart

<script src="BarChart.js"></script>

<script>

window.addEventListener("load", function () {

var chart = createBarChart([30, 6, 88, 91, 2, 45, 100, 150, 200],

600,

300,

"green");

var container = document.getElementById("chartContainer");

container.appendChild(chart);

});

</script>

</head>

<body>

<div id="chartContainer"></div>

</body>

function createBarChart(data, width, height, color) {

// создаем контейнер для диаграммы

// createElementNS первый параметр определяет пространство имен для элемента <svg xmlns="http://www.w3.org/2000/svg">

var chart = document.createElementNS("http://www.w3.org/2000/svg", "svg");

chart.style.width = width;

chart.style.height = height;

// находим максимальное значение в массиве данных

var max = Number.NEGATIVE\_INFINITY;

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

if (max < data[i]) max = data[i];

}

var scale = height / max;

var barWidth = Math.floor(width / data.length);

// создаем отдельный элемент диаграммы

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

var bar = document.createElementNS("http://www.w3.org/2000/svg", "rect");

var barHeight = data[i] \* scale;

bar.setAttribute("height", barHeight + "px");

bar.setAttribute("width", barWidth - 4 + "px");

bar.setAttribute("y", height - barHeight);

bar.setAttribute("x", barWidth \* i );

bar.style.fill = color;

bar.addEventListener("mouseover", onOver);

bar.addEventListener("mouseout", onOut);

chart.appendChild(bar);

}

function onOver() { this.style.fill = "red"; }

function onOut() { this.style.fill = color; }

return chart;

}

## Canvas

### Check canvas support

<script>

// Создание элемента и проверка наличия свойств у полученного объекта.

function supports\_canvas() {

return !!document.createElement("canvas").getContext;

}

if (supports\_canvas()) {

document.write("Ваш браузер поддерживает элемент Canvas");

}

else {

document.write("Ваш браузер НЕ поддерживает элемент Canvas");

}

</script>

### Canvas line

<script type="text/javascript">

function draw() {

// получение элемента и его графического контекста.

var canvas = document.getElementById("canvas");

var context = canvas.getContext("2d");

// создание пути.

context.beginPath();

context.moveTo(100, 100);

context.lineTo(200, 200);

// черчение линии на холсте.

context.stroke();

}

if (window.addEventListener)

window.addEventListener("load", draw, true);

</script>

</head>

<body>

<canvas id="canvas" height="300" width="300" style="border: solid 1px black;">

Ваш браузер не поддерживает элемент <b>Canvas</b>

</canvas>

### Canvas transform

<script type="text/javascript">

function draw() {

// получение элемента и его графического контекста.

var canvas = document.getElementById("canvas");

var context = canvas.getContext("2d");

// сохраняем копию текущего состояния контекста.

context.save();

// переносим графический контекст вправо и вниз.

context.translate(40, 100);

// рисуем линию.

context.beginPath();

context.moveTo(0, 0);

context.lineTo(100, 100);

context.stroke();

// восстанавливаем прежнее состояние контекста.

context.restore();

}

if (window.addEventListener)

window.addEventListener("load", draw, true);

</script>

### Canvas Path

<script type="text/javascript">

function draw() {

var canvas = document.getElementById("canvas");

var context = canvas.getContext("2d");

context.save();

context.translate(100, 100);

// начало пути.

context.beginPath();

context.moveTo(0, 0);

context.lineTo(100, 0);

context.lineTo(100, 100);

context.lineTo(0, 100);

// прорисовка линни от точки (0, 100) к началу пути.

context.closePath();

// прорисовка контура полученного пути.

context.stroke();

context.restore();

}

if (window.addEventListener)

window.addEventListener("load", draw, true);

</script>

</head>

<body>

<canvas id="canvas" height="300" width="300" style="border: solid 1px black;">

Ваш браузер не поддерживает элемент <b>Canvas</b>

</canvas>

### Canvas line style

<script type="text/javascript">

function draw() {

// получение элемента и его графического контекста.

var canvas = document.getElementById("canvas");

var context = canvas.getContext("2d");

context.save();

context.translate(100, 100);

// начало пути.

context.beginPath();

context.moveTo(0, 0);

context.lineTo(100, 0);

context.lineTo(100, 100);

context.lineTo(0, 100);

context.closePath();

// настройка стилей линии.

context.lineWidth = 8;

context.strokeStyle = "green";

context.lineJoin = "round";

context.stroke();

// восстанавливаем прежнее состояние контекста.

context.restore();

}

if (window.addEventListener)

window.addEventListener("load", draw, true);

</script>

</head>

<body>

<canvas id="canvas" height="300" width="300" style="border: solid 1px black;">

Ваш браузер не поддерживает элемент <b>Canvas</b>

</canvas>

### Canvas fill style

<script type="text/javascript">

function draw() {

// получение элемента и его графического контекста.

var canvas = document.getElementById("canvas");

var context = canvas.getContext("2d");

context.save();

context.translate(100, 100);

// начало пути.

context.beginPath();

context.moveTo(0, 0);

context.lineTo(100, 0);

context.lineTo(100, 100);

context.lineTo(0, 100);

context.closePath();

// настройка стиля заливки

context.fillStyle = "blue";

context.fill();

// настройка стилей линии.

context.lineWidth = 8;

context.strokeStyle = "green";

context.lineJoin = "round";

context.stroke();

// восстанавливаем прежнее состояние контекста.

context.restore();

}

if (window.addEventListener)

window.addEventListener("load", draw, true);

</script>

</head>

<body>

<canvas id="canvas" height="300" width="300" style="border: solid 1px black;">

Ваш браузер не поддерживает элемент <b>Canvas</b>

</canvas><script type="text/javascript">

function draw() {

// получение элемента и его графического контекста.

var canvas = document.getElementById("canvas");

var context = canvas.getContext("2d");

context.save();

context.translate(100, 100);

// начало пути.

context.beginPath();

context.moveTo(0, 0);

context.lineTo(100, 0);

context.lineTo(100, 100);

context.lineTo(0, 100);

context.closePath();

// настройка стиля заливки

context.fillStyle = "blue";

context.fill();

// настройка стилей линии.

context.lineWidth = 8;

context.strokeStyle = "green";

context.lineJoin = "round";

context.stroke();

// восстанавливаем прежнее состояние контекста.

context.restore();

}

if (window.addEventListener)

window.addEventListener("load", draw, true);

</script>

</head>

<body>

<canvas id="canvas" height="300" width="300" style="border: solid 1px black;">

Ваш браузер не поддерживает элемент <b>Canvas</b>

</canvas>

### Canvas fill rect

<script type="text/javascript">

function draw() {

var canvas = document.getElementById("canvas");

var context = canvas.getContext("2d");

// заполнение прямоугольной области синим цветом.

context.fillStyle = "blue";

context.fillRect(100, 100, 50, 50);

// вычерчивание красной границы толщиной в 10 пикселей

context.strokeStyle = "red";

context.lineWidth = 10;

context.strokeRect(100, 100, 50, 50);

// очистка прямоугольной области от контента.

context.clearRect(125, 125, 35, 35);

}

if (window.addEventListener)

window.addEventListener("load", draw, true);

</script>

</head>

<body>

<canvas id="canvas" height="300" width="300" style="border: solid 1px black;">

Ваш браузер не поддерживает элемент <b>Canvas</b>

</canvas>

### Canvas load image

<script type="text/javascript">

function draw() {

var canvas = document.getElementById("canvas");

var context = canvas.getContext("2d");

var logo = new Image();

logo.onload = function () {

context.drawImage(logo, 22, 22);

}

logo.src = "Logo.png";

}

if (window.addEventListener)

window.addEventListener("load", draw, true);

</script>

</head>

<body>

<canvas id="canvas" height="300" width="430" style="border: solid 1px black;">

Ваш браузер не поддерживает элемент <b>Canvas</b>

</canvas>

### Canvas chart

<script src="BarChart.js"></script>

<script>

window.addEventListener("load", function () {

createBarChart("chartContainer",

[30, 6, 88, 91, 2, 45, 100, 150, 200],

600,

300,

"green");

});

</script>

</head>

<body>

<canvas id="chartContainer"></canvas>

</body>

function createBarChart(canvas, data, width, height, color) {

if (typeof canvas == "string") canvas = document.getElementById(canvas);

canvas.width = width;

canvas.height = height;

var context = canvas.getContext("2d");

// находим максимальное значение в массиве данных

var max = Number.NEGATIVE\_INFINITY;

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

if (max < data[i]) max = data[i];

}

var scale = height / max;

var barWidth = Math.floor(width / data.length);

// создаем отдельный элемент диаграммы

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

var barHeight = data[i] \* scale,

x = barWidth \* i,

y = height - barHeight;

context.fillStyle = color;

context.fillRect(x, y, barWidth - 2, barHeight);

}

}

# Ajax

## XMLHttpRequest

### SyncRequest

<script>

window.onload = function () {

document.getElementById("btnGet").onclick = function () {

var xhr = new XMLHttpRequest(); // Создание объекта для HTTP запроса.

xhr.open("GET", "testfile.html", false); // Настройка объекта для отправки синхронного GET запроса

xhr.send(); // Отправка запроса, так как запрос является синхронным, следующая строка кода выполнится только после получения ответа со стороны сервера.

document.getElementById("output").innerHTML += xhr.responseText; // responseText - текст ответа полученного с сервера.

}

}

</script>

</head>

<body>

<input id="btnGet" type="button" value="Get Info" />

<p id="output"></p>

</body>

### AsyncRequest

<script>

window.onload = function () {

document.getElementById("btnGet").onclick = function () {

var xhr = new XMLHttpRequest(); // Создание объекта для HTTP запроса.

xhr.open("GET", "testfile.html", true); // Настройка объекта для отправки асинхронного GET запроса

// функция-обработчик срабатывает при изменении свойства readyState

// Значения свойства readyState:

// 0 - Метод open() еще не вызывался

// 1 - Метод open() уже был вызван, но метод send() еще не вызывался.

// 2 - Метод send() был вызван, но ответ от сервера еще не получен

// 3 - Идет прием данных от сервера. Для значения 3 Firefox вызывает обработчик события несколько раз IE только один раз.

// 4 - Ответ от сервера полностью получен (Запрос успешно завершен).

xhr.onreadystatechange = function () {

if (xhr.readyState == 4) { // если получен ответ

if (xhr.status == 200) { // и если статус код ответа 200

document.getElementById("output").innerHTML += xhr.responseText; // responseText - текст ответа полученного с сервера.

}

}

}

xhr.send(); // Отправка запроса, так как запрос асинхронный сценарий продолжит свое выполнение. Когда с сервера придет ответ сработает событие onreadystatechange

}

}

</script>

</head>

<body>

<input id="btnGet" type="button" value="Get Info" />

<p id="output"></p>

</body>

## HttpMethods

### GetHttpMethod

<script>

window.onload = function () {

var xhr = new XMLHttpRequest();

// настройка объекта запроса с указание метода отправи запроса и данных

xhr.open("GET", "GetHandler.ashx?a=hello&b=world");

xhr.onreadystatechange = function () {

if (xhr.readyState == 4 && xhr.status == 200) {

document.getElementById("output").innerHTML = xhr.responseText;

}

}

xhr.send();

}

</script>

</head>

<body>

<p id="output"></p>

</body>

GetHandler.ashx

<%@ WebHandler Language="C#" Class="GetHandler" %>

using System;

using System.Web;

public class GetHandler : IHttpHandler {

public void ProcessRequest (HttpContext context) {

context.Response.ContentType = "text/plain";

// Чтение данных из строки запроса.

string aParam = context.Request.QueryString["a"];

string bParam = context.Request.QueryString["b"];

// Формирование ответа.

context.Response.Write("<b>GET</b> параметры переданные с запросом: a=" + aParam + ", b=" + bParam);

}

public bool IsReusable {

get {

return false;

}

}

}

### PostHttpMethod

<script>

window.onload = function () {

var xhr = new XMLHttpRequest();

// настройка объекта запроса с указание метода отправи запроса и данных

xhr.open("POST", "PostHandler.ashx");

// Content-Type - HTTP заголовок, который указывает серверу, как интерпретировать тело запроса.

xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");

xhr.onreadystatechange = function () {

if (xhr.readyState == 4 && xhr.status == 200) {

document.getElementById("output").innerHTML = xhr.responseText;

}

}

// отправка запроса с указанием данных.

xhr.send("a=hello&b=world");

}

</script>

</head>

<body>

<p id="output"></p>

</body>

PostHandler.ashx

<%@ WebHandler Language="C#" Class="PostHandler" %>

using System;

using System.Web;

public class PostHandler : IHttpHandler {

public void ProcessRequest (HttpContext context) {

context.Response.ContentType = "text/plain";

// Чтение POST данных.

string aParam = context.Request.Form["a"];

string bParam = context.Request.Form["b"];

// Формирование ответа.

context.Response.Write("<b>POST</b> параметры переданные с запросом: a=" + aParam + ", b=" + bParam);

}

public bool IsReusable {

get {

return false;

}

}

## Samples

### TimeOutRequest

<title>AJAX Timeout</title>

<script>

window.onload = function () {

document.getElementById("buttonGet").onclick = function () {

var xhr = new XMLHttpRequest();

// время ожидания.

var timeout = 20000;

// запуск функции отмены запроса через указанный промежуток времени.

var timer = setTimeout(function () { xhr.abort(); alert("aborted"); }, timeout);

xhr.open("GET", "TestHandler.ashx");

xhr.onreadystatechange = function () {

if (xhr.readyState == 4 && xhr.status == 200) {

clearTimeout(timer); // отмена запуска abort() метода если пришел ответ от сервера.

document.getElementById("output").innerHTML = xhr.responseText;

}

}

xhr.send();

}

}

</script>

</head>

<body>

<input id="buttonGet" type="button" value="Get Data" />

<p id="output"></p>

</body>

TestHanler.ashx

<%@ WebHandler Language="C#" Class="TestHandler" %>

using System;

using System.Web;

public class TestHandler : IHttpHandler {

public void ProcessRequest (HttpContext context) {

context.Response.ContentType = "text/plain";

System.Threading.Thread.Sleep(3000); // Задержка на 3 секунды.

context.Response.Write("Hello World " + DateTime.Now.ToLocalTime() ); // Отправка ответа.

}

public bool IsReusable {

get {

return false;

}

}

### RequestProgress

<head>

<title>Индикатор загрузки</title>

<script>

window.onload = function () {

// фунцикя для отображения индикатора загрузки.

function show() {

document.getElementById("loader").style.display = "block";

}

// фунцикя для удаления индикатора загрузки.

function hide() {

document.getElementById("loader").style.display = "none";

}

var xhr = new XMLHttpRequest();

document.getElementById("buttonGet").onclick = function () {

xhr.open("GET", "TestHandler.ashx");

xhr.onreadystatechange = function () {

if (xhr.readyState == 4 && xhr.status == 200) {

hide(); // скрыть индикатор.

document.getElementById("output").innerHTML = xhr.responseText;

}

}

xhr.send();

show(); // отобразить индикатор.

}

}

</script>

</head>

<body>

<input id="buttonGet" type="button" value="Get Data" />

<p id="loader" style="display: none; color:red;">

<img src="ajax-loader.gif" /></p>

<p id="output"></p>

</body>

### JsonRequest

<head>

<title>Получение JSON объекта</title>

<script>

window.onload = function () {

var xhr = new XMLHttpRequest();

document.getElementById("buttonGet").onclick = function () {

xhr.open("GET", "JsonHandler.ashx");

xhr.onreadystatechange = function () {

if (xhr.readyState == 4 && xhr.status == 200) {

// JSON.parse - преобразование в объект строки полученной с сервера.

var data = JSON.parse(xhr.responseText);

document.getElementById("output").innerHTML = "Имя - " + data.firstName + ", Фамилия - " + data.lastName;

}

}

xhr.send();

}

}

</script>

</head>

<body>

<input id="buttonGet" type="button" value="Get Data" />

<p id="output"></p>

</body>

JsonHandler

<%@ WebHandler Language="C#" Class="JSONHandler" %>

using System;

using System.Web;

public class JSONHandler : IHttpHandler {

public void ProcessRequest (HttpContext context) {

// ContentType указывающий на наличие JSON объекта в ответе.

context.Response.ContentType = "application/json";

// отправка JSON объекта

// { "firstName":"Ivan", "lastName":"Ivanov" } - объект со свойтсвами firstName и lastName.

context.Response.Write("{\"firstName\":\"Ivan\", \"lastName\":\"Ivanov\"}");

}

public bool IsReusable {

get {

return false;

}

}