Мировые информационные ресурсы

Преподаватель:

Попков Сергей Игоревич,
аспирант,
заведующий лабораторией

(rsteach0@gmail.com)

Лекция 4. CSS: стиль текста, списки, таблицы и другое.
HTML5: Canvas, SVG

Комбинаторы CSS

- Комбинатор задает отношения между селекторами
- Селектор CSS может содержать больше одного простого селектора. Между простыми селекторами в выражение вставляется комбинатор
- Существует 4 разных комбинатора в CSS3:
  - селектор-преемник (space)
  - селектор-потомок (>)
  - селектор-ближайший родственник (+)
  - селектор-родственник общего вида (~)
    - Примеры: div p{}, div > p {}, div + p{}, div ~ p{}
- Селектор-преемник соответствует всем элементам, вложенным в указанный
- Селектор-потомок находит все элементы, которые состоят в непосредственной дочерней связи с указанным
- Селектор-ближайший родственник указывает на соответствующую группу элементов-ближайших родственников; родственные элементы происходят от одного предка, а “ближайший” в данном контексте означает "незамедлительно следующий за указанным"
- Селектор-родственник общего вида определяет все элементы, которые являются родственными указанному

Счетчики CSS

- Счетчики CSS – это "выражения", поддерживаемые CSS, при этом значения выражений могут быть увеличены на основе правил CSS (чтобы отследить, сколько раз они были использованы).
- Счетчики позволяют видоизменить представление содержимого на основании положения в документе
- Для работы с счетчиками CSS применяются следующие свойства:
  - counter-reset – создает или сбрасывает счетчик
  - counter-increment – увеличивает значение счетчика
  - content – вставляет сгенерированное содержимое
  - counter() или counters() – функции для добавления значения счетчика элементу
- Чтобы воспользоваться счетчиком CSS, необходимо сначала создать его с помощью counter-reset
- Свойство content может быть:
  - normal → по умолчанию; устанавливает содержимое в нормальное состояние, по умолчанию "none" → “ничто”
  - counter → устанавливает содержимое равным счетчику
  - attr(attribute) → устанавливает содержимое равным одному из атрибутов соответствущего селектора
  - string → устанавливает содержимое в указанный текст [“text”]
  - open-quote/close-quote → устанавливает содержимое в открытую/закрытую кавычку
  - no-open-quote/no-close-quote → удаляет открытую/закрытую кавычку из содержимого при наличии
  - url(url) → Устанавливает содержимое в медиаконтент (изображение, звук, видео и т.п.)
  - initial → Устанавливает свойств в изначальное состояние;
  - inherit→ наследует свойство от элемента-предка

Реализация счетчиков

- Пример счетчиков:

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
<style>
body {counter-reset: section;}
p::before {
    counter-increment: section;
    content: "Paragraph " counter(section) ": ";
}
p::after {
    content: url('googlelogo.png');
}
</style>
</head>
<body>
<p>Text</p>
<p>Text</p>
<p>Text</p>
</body>
</html>
```

Paragraph 1: Text

Paragraph 2: Text

Paragraph 3: Text

Свойства для оформления текста

- Свойство color задает цвет текста
- Свойство text-align задает горизонтальное расположение текста (center, left, right, justify)
- Свойство text-decoration используется для оформления текста с помощью прямых (overline, line-through, underline, none)
- Свойство text-transform позволяет задавать верхний либо нижний регистр текста (uppercase, lowercase, capitalize).
- Свойство text-indent задает отступ первой строки текста
- Свойство letter-spacing применяется для указания межсимвольного расстояния в тексте
- Свойство line-height определяет расстояние между строками
- Свойство word-spacing определяет расстояние между словами в тексте
- Свойство text-shadow определяет тень текста (1px 2px red) ← гор, вер, цвет
- Свойство direction задает направление текста в элементе (rtl, ltr)
- “resize:none;” применяется в <textarea> для запрета процедуры изменения размера окна, доступной по умолчанию

Группа свойств font

- Свойства CSS группы font определяют семейство шрифтов, жирность, размер и стиль текста
- В CSS существуют два типа имен семейств шрифтов
  - общее семейство – группа однообразных семейств шрифтов (like "Serif" or "Monospace")
  - семейство шрифтов – специфическое ("Times New Roman", "Arial")
- Семейство шрифтов текста определяется свойством font-family
- Свойство font-family может содержать несколько имен шрифтов на случай сбоев: если браузер не поддерживает первый указанный шрифт, он попробует следующий – и т.д.
- Рекомендуется сначала указывать желаемый шрифт, а заканчивать общим семейством, чтобы позволить браузеру выбрать схожий шрифт из общего семейства, если другие перечисленные шрифты недоступны
- Если имя семейства шрифтов содержит более одного слова, необходимо заключать его в кавычки ("Times New Roman")
- Можно задать набор семейств шрифтов в списке через запятую
  - font-family: "Times New Roman", Times, serif;
- Свойство font-style чаще всего используется для задания курсива (normal, italic, oblique)
- Свойство font-weight задает вес шрифта (normal, bold)
- Свойство font-variant задает, должен ли шрифт отображаемого текста представляться в small-caps (normal, small-caps)
  - Шрифт, преобразованный в small-caps, преобразует все буквы в нижнем регистре к верхнему, но верхний при этом имеет меньший размер, чем в оригинальном тексте

Свойство font-size

- Свойство font-size задает размер текста
- Навык управления размером текста крайне важен в веб-дизайне, но не следует подгонять размер шрифта, чтобы сделать параграфы выглядящими как заголовки и наоборот; используйте корректные тэги
- Значение font-size может быть абсолютным и относительным
- Абсолютное:
  - задает тексту указанный размер
  - не позволяет пользователю изменять размер текста в браузерах (нехорошо с точки зрения доступности)
  - абсолютный размер полезен, когда известен физический размер вывода
- Относительное:
  - устанавливает размер относительно окружающих элементов
  - позволяет пользователю изменять размер текста в браузерах
- Если размер шрифта не указан, то по умолчанию для обычного текста (вроде параграфов) он равен 16px (16px=1em)
- Установка размера текста в пикселях дает полный контроль над размером текста: p{font-size: 14px;}
  - Даже при использовании пикселей можно масштабировать страницу для увеличения отдельных элементов
- Чтобы пользователь мог более гибко изменять размер текста на странице при просмотре, разумно использовать размерность em вместо px
- Единица em рекомендована W3C: 1 em = текущему размеру шрифта (по умолчанию – 16px). Исходя из умолчания, размер шрифта определяется по формуле: пиксели/16=em
  - p {font-size: 0.875em; /* 14px/16=0.875em */}
  - В примере выше размер текста в em тот же, что и предыдущий в пикселях. Однако, единицы em позволяют подогнать размер текста в браузере
- Существует проблема со старыми версиями браузеров IE, когда текст не изменяет размер пропорционально. Решением является задание размера шрифта по умолчанию для тэга <body> в процентах (используя комбинацию % и Em)
  - body {font-size: 100%;}p {font-size: 0.875em;}

Оформление таблиц в CSS

- Для задания границ таблиц в CSS применяется свойство border. Если нужна граница только вокруг таблицы, следует указать свойство border только для <table>
- Свойство border-collapse определяет, будут ли границы таблицы на стыке элементов соединяться воедино
- Ширина и высота таблицы определяются свойствами width и height соответственно
- Свойство text-align определяет горизонтальное положение содержимого (left, right, center) в <th> или <td>
- Свойство vertical-align определяет вертикальное положение содержимого (top, bottom, middle) в <th> или <td>
- Для управления пространством между границей и содержимым таблицы используется свойство padding элементов <td> и <th>
- Свойство border-bottom для <th> и <td> задает горизонтальные разделители
- Селектор :hover у <tr> позволяет выделять строки таблицы при наведении курсора мыши на них
- Чересполосные таблицы применяют селектор nth-child() с установленным свойством background-color для выделения четных (even) или нечетных (odd) строк таблицы
  - tr:nth-child(even) {background-color: lightgray}
- Свойства background-color и color применяются для указания цвета фона и текста соответственно
- Чтобы избежать отображения горизонтальной полосы прокрутки у таблицы, когда экран слишком мал для отображения всего содержимого, сделайте предком таблицы блочный элемент (например,) со свойством overflow-x:auto – таблица станет гибче размещать содержимое
- <table> <!-- <...> --> </table>

Списки в CSS

- Неупорядоченный список задается тэгом
- Каждый пункт списка задается тэгом <li> (справедливо и для упорядоченного списка)
- По умолчанию, элементы списка выделяются малыми черными кружочками
- Свойство list-style-type определяет вид маркера для выделения элемента списка
  - disc → по умолчанию
  - circle → окружность
  - square → прямоугольник
  - none → отсутствие маркера
- Упорядоченный список задается тэгом
- Пункты в упорядоченном списке выделяются цифрами по умолчанию
- Атрибут “type” тэга <ol> задает тип маркера в упорядоченном списке
  - type="1" → по умолчанию
  - type="A" → латинские буквы верхнего регистра
  - type="a" → латинские буквы нижнего регистра
  - type="I" → римские цифры, верхний регистр
  - type="i" → римские цифры, нижний регистр
- Списки могут быть вложенными
- HTML также поддерживает описательные списки: набор – понятий с описанием каждого из них. Тэг <dl> задает описательный список, тэг <dt> – понятие (имя), а тэг <dd> – описание
  - <dl><dt>term</dt><dd>- description</dd></dl>

Свойства для определения стилей списков

- Свойство list-style-image определяет изображение в качестве маркера списка
  - ul {list-style-image: url('picture.png');}
- Свойство list-style-position определяет, должны ли маркеры списка появляться внутри или снаружи потока содержимого
  - ul {list-style-position: inside;}
- Свойство list-style – сокращенное свойство, применяемое для установки всех свойств списка в одном объявлении (list-style-type list-style-position list-style-image)
  - Если list-style-image определен, значение list-style-type property будет отображено вместо соответствующего изображения, если оно по какой-либо причине не сможет отобразиться
- Существует возможность определить стиль списков с помощью цветов. Все, что добавляется к тэгам <ol> or <ul>, влияет на весь список, а свойства, добавляемые к тэгу <li>, влияют на отдельные пункты списка
  - ol {background: #ff0000;} /* как rgb(255,0,0) → 16-ричное значение */
  - ul {background: #ff00ff;}
  - ol li {background: #7f7f7f;}
  - ul li {background: #0000ff;}

Свойство position

- Свойство position определяет род способа позиционирования элемента (static, relative, fixed, absolute)
- Элементы размещаются на основе свойств top, bottom, left, right. Однако, эти свойства не будут действовать пока не будет установлено свойство position. Они также обрабатываются по-разному в зависимости от значения свойства position
- Элементы HTML по умолчанию размещаются статично (static). Такой элемент просто следует обычному потоку вывода содержимого страницы
- “position: relative;” задает положение элемента относительным по отношению к обычному положению. Изменение свойств top, right, bottom, left такого элемента задает смещение относительно обычного положения, а прочее содержимое не будет смещено на место образовавшегося пространства
- “position: fixed;” задает положение элемента относительным по отношению к области просмотра, то есть всегда стоит на месте и не прокручивается. Свойства top, right, bottom, left применяются для задания точного местонахождения элемента. Такой элемент не образует пространства на месте своего обычного положения в потоке вывода содержимого страницы
- “position: absolute;” задает положение элемента относительным по отношению к ближайшему предку “с размещением”. Если же предка “с размещением” нет, то в этом качестве используется тело документа, что позволяет искомому элементу реагировать на прокрутку
- Элемент "с размещением" – такой, у которого свойство “position” не установлено в “static”
- Элементы "с размещением" могут перекрывать другие. Свойство “z-index” задает порядок отображения элемента (размещение поверх или позади прочих). Может быть как положительным, так и отрицательным
  - Элемент с большим значением “z-index” всегда впереди элементов с меньшим значением
  - Если два элемента “с размещением” перекрывают друг друга, не обладая свойством “z-index”, сверху оказывается элемент, объявленный позднее в документе HTML

Обрезка содержимого в CSS

- Свойство overflow решает, следует ли обрезать содержимое (или добавить полосы прокрутки), если содержимое элемента слишком велико для того, чтобы поместиться в отведенное пространство
- Возможные значения свойства overflow:
  - visible – по умолчанию. Содержимое не обрезается, “переливаясь” за пространство контейнера
  - hidden – содержимое обрезается, избыток не отображается
  - scroll – содержимое обрезается, но добавляются полосы прокрутки
  - auto – при обрезке содержимого появляются полосы прокрутки
- По умолчанию, никаких ограничений на представление содержимого заходящего за пространство элемента не накладывается
- Значение “hidden” “скрывает” содержимое, выходящее за пространство – оно просто не отображается
- Значение “scroll” задает полосы прокрутки, вертикальные и горизонтальные, для содержимого, вне зависимости от того, требует оно обрезки или нет
- Значение “auto” аналогично значению “scroll”, но добавляет полосы прокрутки только по необходимости
- Свойства “overflow-x” и “overflow-y” задают, следует ли контролировать обрезку содержимого лишь горизонтально или вертикально, соответственно (или оба значения одновременно)
- Особый пример – обрезка рисунка (не работает для “overflow:visible”)
  - img {position: absolute; clip: rect(0px,60px,200px,0px);}

Обтекание в CSS

- Свойство float указывает, должен ли элемент быть обтекаемым (none,left,right)
  - Простейшее применение: для задания обтекания текста вокруг изображений
- Свойство clear управляет поведением обтекающих элементов (none,left,right,both)
  - Элементы после плавающего будут размещаться вокруг его положения; во избежание этого эффекта и применяется свойство clear
- Если обтекаемый элемент длиннее элемента-предка, он выльется за пределы пространства предка. Задание свойства “overflow: auto;” для предка позволит этого избежать
  - .clearfix {overflow: auto;}
- Современная версия:
  - .clearfix::after {content: "";clear: both;display: table;}
- Сетку прямоугольных областей можно было задать издавна таким образом, чтобы она заполняла ширину окна браузера и растягивалась при необходимости (при изменении размеров браузера) благодаря свойству float
- Значение свойства display “inline-block” позволяет упростить этот процесс
- Элементы такого типа задаются как элементы “inline”, но с шириной и высотой
- Старый стиль:
  - .floating-box {float: left;/*<...>*/}
  - .after-box {clear: left;}
- Новый стиль:
  - .floating-box {display: inline-block;/*<...>*/}

Управление размещением содержимого элемента

- В CSS существует несколько свойств для задания выравнивания элементов по горизонтали и вертикали
- Чтобы отцентрировать блочный элемент по горизонтали (например,), применяется “margin: auto;”. Задание ширины (width) элемента не даст ему растянуться до границ своего предка. Элемент займет указанную ширину, а оставшееся пространство будет разделено поровну между двумя margin-отступами
  - Такое размещение не будет действовать без задания свойства width (в любое значение, кроме “100%”)
- Для централизации текста внутри элемента применяется “text-align: center;”
- Для центрирования изображения применяется “margin: auto;” с преобразованием вывода в блочный элемент:
  - img {display: block; margin: auto; width: 50%;}
- Выравнивание элементов (по левому или правому краю) осуществляется с помощью “position: absolute;”, Но такие элементы, как уже говорилось, могут перекрывать прочие элементы, выбиваясь из стандартного потока вывода содержимого документа
  - При выравнивании элементов с помощью свойства “position” нужно всегда определять отступы margin и padding элемента <body>, чтобы избежать разницы в представлении страницы браузерами
- Другой способ выравнивания – воспользоваться свойством float
  - При выравнивании элементов с помощью свойства “float” нужно придерживаться тех же правил, что и при выравнивании элементов с помощью свойства “position”
- Объявление !DOCTYPE всегда устанавливается, хотя бы чтобы избежать различных проблем с применением свойств “position” и “float” в браузерах IE версии 8 и ниже
- Для вертикального центрирования существует ряд способов, например, через отступы padding (top и bottom). Другой прием – свойство line-height со значением, равным свойству height. Если эти два способа нельзя применить по каким-либо причинам, третье решение – элемент “с размещением” и свойством “transform” (преобразующим положение):
  - transform: translate(-50%, -50%);

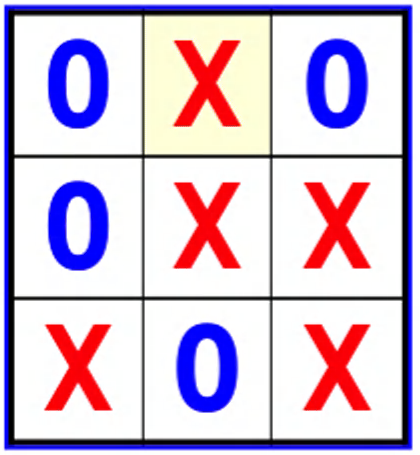
Обновление игры X/O

- Листинги

```html
<!DOCTYPE html><!--Модель поля для игры в Крестики-Нолики 3.0-->
<html>
<head>
<title>Page Title</title>
<link rel="stylesheet" type="text/css" href="theme.css">
<script src="jscript.js"></script>
</head>
<body onload="filltable()">
<div class="aligneDiv">
<pre>
<table class="tableClass tableClassX" id="TableToFill"></table>
</pre>
</div>
</body>
</html>
```

```css
/*theme.css*/
table, td {
    border: 1px solid;
    border-collapse: collapse;
}
div.aligneDiv{
    position: absolute;
    left:50%;
    top:50%;
    transform: translate(-50%, -50%);
}
td:hover {background-color: lightyellow;}
.X {color: red;}
.O {color: blue;}
pre {font-size:4.5em;}
pre > table { width:200px; height:200px; text-align:center;}
table.tableClass  {border: 3px solid;outline:2px solid black;}
table.tableClassX {outline-color:red;}
table.tableClassO {outline-color:blue;}
```

```javascript
//jscript.cs
var stepdraw="X";
function tdc(e){
    if(e.innerHTML!=" ")return; //Работа над ошибками ;) Кто заметил - молодец...
    e.innerHTML="<b class='"+stepdraw+"'>"+stepdraw+"</b>";
    stepdraw=stepdraw=="X"?"O":"X";
    var classch="tableClass tableClass"+stepdraw;
    document.getElementById("TableToFill").setAttribute("class",classch);
}
function filltable()
{
    var s='';
    for(i=0;i<3;i++){
        s+='<tr>';
        for(j=0;j<3;j++){
            s+='<td onclick="tdc(this);"> </td>';
        }
        s+='</tr>';
    }
    document.getElementById("TableToFill").innerHTML=s;
}
```



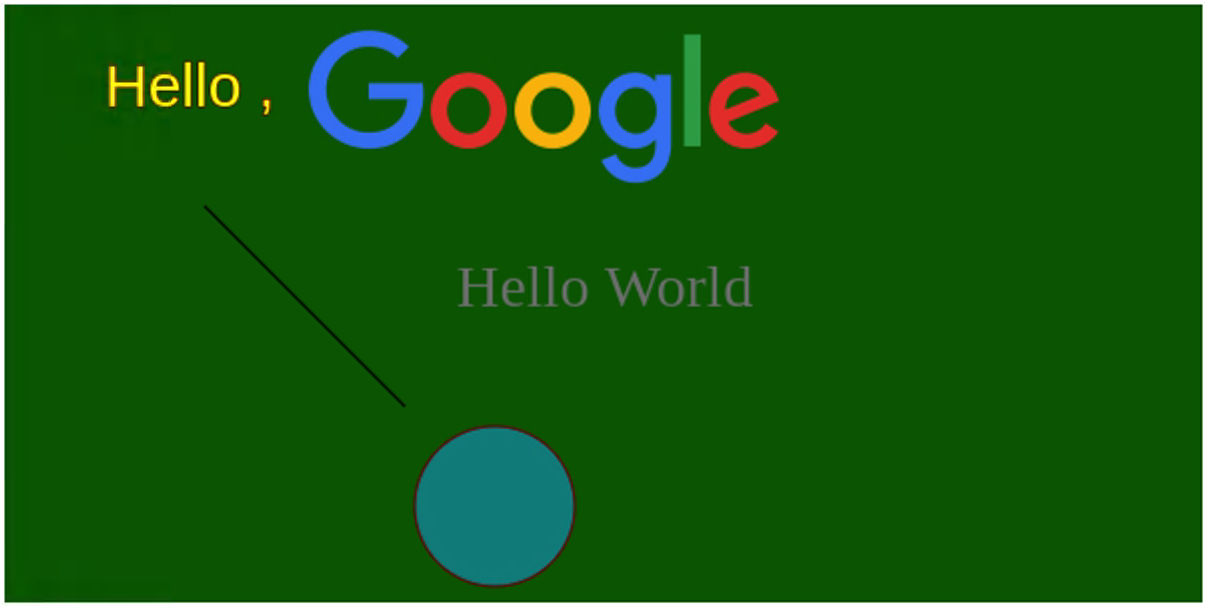
Полотно HTML5

- HTML-элемент <canvas> применяется для рисования изображений в реальном времени с помощью JavaScript. Элемент <canvas> представляет собой лишь контейнер для графики, для его наполнения требуется код на JavaScript
- Для полотна (canvas) существует несколько методов для отрисовки маршрутов, прямоугольных и круговых областей, текстов и загружаемых изображений. Полотно представляет собой прямоугольное пространство на HTML-странице, по умолчанию - без содержимого и границ
- Пример разметки полотна:
  - <canvas id="myCanvas" width="600" height="300"></canvas>
- Всегда указывайте id для обращения к полотну из JavaScript, а также ширину и высоту (width и height, соответственно) для задания размеров полотна. Для задания рамки средствами CSS можно воспользоваться атрибутом style
- Во-первых, необходимо определить искомый элемент <canvas>
- Во-вторых, получить объект для вывода изображения
  - Такой встроенный объект в HTML5 возвращается методом getContext(), со всеми необходимыми свойствами и методами для рисования
- Наконец, можно строить изображение на полотне. Можно установить стиль заливки (например, красный цвет). Пример построения изображения: метод fillRect(x,y,width,height) прорисовывает закрашенный прямоугольник на полотне
  - Свойство fillStyle может быть цветом CSS, градиентом, или графическим шаблоном (по умолчанию свойство выставлено в черный цвет)

Пример полотна HTML5

- Листинг и результат:

```
<!DOCTYPE html><html>
<head><title>Canvas lesson</title></head>
<body><canvas id="myCanvas" width="600" height="300"></canvas>
<img id="glogo" style="display:none"; src="googlelogo.png"></body>
<script>
var canvas = document.getElementById("myCanvas");
var ctx = canvas.getContext("2d");
ctx.fillStyle = "darkgreen";
ctx.fillRect(0,0,canvas.width,canvas.height);
ctx.moveTo(100,100);
ctx.lineTo(200,200);
ctx.stroke();
ctx.fillStyle = "darkcyan";
ctx.strokeStyle = "darkred";
ctx.beginPath();
ctx.arc(245,250,40,0,2*Math.PI);
ctx.fill();
ctx.stroke();
ctx.font = "30px Arial";
ctx.fillStyle = "yellow";
ctx.strokeText("Hello ,",50,50);
ctx.fillText("Hello ,",50,50);
ctx.font = "30px Comic Sans MS";
ctx.fillStyle = "gray";
ctx.textAlign = "center";
ctx.fillText("Hello World", canvas.width/2, canvas.height/2);
ctx.drawImage(document.getElementById("glogo"), 150, 10);
</script></html>
```



Градиент

- Градиенты применяются для заливки различных фигур (включая текст) на полотне. Это позволяет не ограничиваться базовыми цветами при построении фигур
- Существует два типа градиентов
  - createLinearGradient(x,y,x1,y1) – линейный градиент
  - createRadialGradient(x,y,r,x1,y1,r1) – радиальный (круговой) градиент
- После создания объекта градиента, необходимо задать 2 или более переходов цвета
  - Переходы и их позиции вдоль градиента формируются вызовом метода addColorStop(position,color). Позиции градиента принадлежат отрезку [0;1]
  - Чтобы воспользоваться градиентом, установите свойство fillStyle или strokeStyle равным градиенту, после чего постройте требуемую фигуру
- Пример:

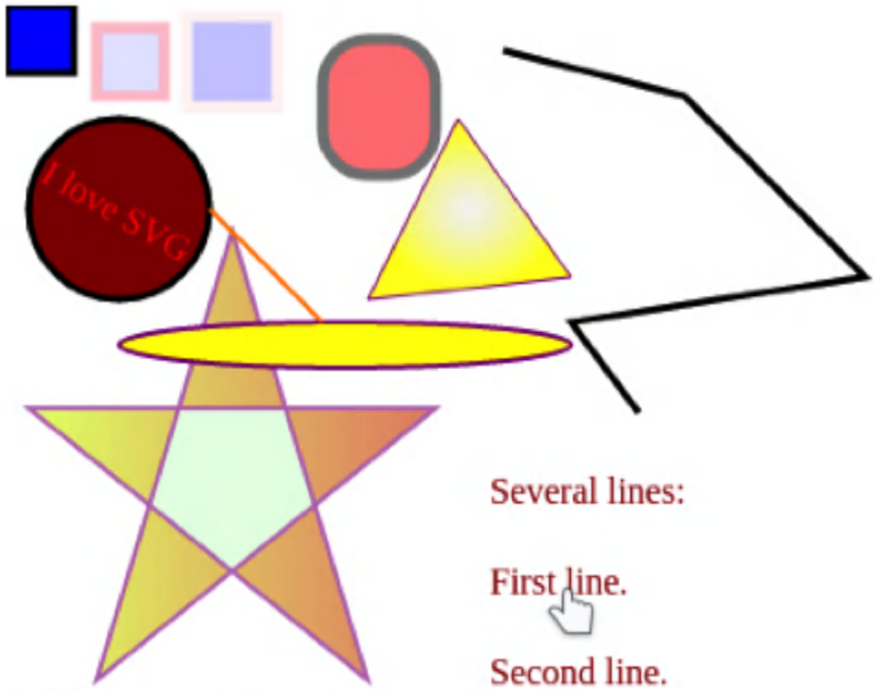
```
var c=document.getElementById("myCanvas");
var ctx=c.getContext("2d");
// Создадим линейный ...
var grd=ctx.createLinearGradient(0,0,200,0);
grd.addColorStop(0,"red");
grd.addColorStop(1,"white");
//… и радиальный граденты:
var grd=ctx.createRadialGradient(75,50,5,90,60,100);
grd.addColorStop(0,"red");
grd.addColorStop(1,"white");
// Зальем прямоугольную область градиентом
ctx.fillStyle=grd;
ctx.fillRect(10,10,150,80);
```

SVG

- Формат SVG – Scalable Vector Graphics – “Масштабируемая векторная графика”
- SVG применяется для описания графики внутри веб-контента
- SVG рекомендован W3C
- HTML-элемент <svg> - контейнер для графики SVG. В SVG существует множество методов для рисования маршрутов, круглых и прямоугольных областей, текстов, графических изображений и т.п.
- SVG – язык для описания двумерной графики в формате XML. В этом его отличие от полотна, на котором двумерная графика рисуется во время выполнения (благодаря JavaScript)
- То, что SVG основан на XML, означает доступность каждого элемента в рамках SVG DOM. Таким образом, есть возможность присоединить обработчики событий JavaScript к конкретному элементу
- Каждая отрисованная фигура в SVG запоминается как объект. При изменении атрибутов такого объекта браузер способен автоматически перерисовать соответствующую фигуру. Полотно же формируется попиксельно; как только изображение нарисовано, браузер забывает его структуру. Если положение полотна должно быть изменено, вся соответствующая область должна быть перерисована, включая все объекты, которые могли быть каким-либо образом перекрыты графической областью полотна
- Преимущества графического формата SVG перед другими (такими, как JPEG, PNG или GIF):
  - SVG можно создавать и модифицировать в любом текстовом редакторе
  - SVG можно индексировать, подвергать поиску, сжимать и обрабатывать с помощью кода JavaScript
  - SVG можно масштабировать
  - SVG можно печатать с высоким качеством для любого разрешения
  - SVG позволяет увеличивать изображение в браузере без искажений
  - SVG является открытым стандартом
  - SVG может храниться как содержимое файлов, представимое в качестве чистого XML

Пример работы с SVG

- Листинг и результат:




```
<!DOCTYPE html><html><head><title>SVG lesson</title></head><body>
<svg width="600" height="800">
<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>
<radialGradient id="grad2" cx="50%" cy="50%" r="50%" fx="50%" fy="50%">
<stop offset="0%" style="stop-color:rgb(227,227,227); stop-opacity:0.4" />
<stop offset="90%" style="stop-color:rgb(255,255,0); stop-opacity:0.9" />
</radialGradient>
</defs>
<rect width="30" height="30" style="fill:rgb(0,0,255);stroke-width:3;stroke:rgb(0,0,0)" />
<rect x="40" y="10" width="30" height="30" style="fill:blue;stroke:pink;stroke-width:5;fill-opacity:0.1;stroke-opacity:0.9" />
<rect x="80" y="5" width="40" height="40" style="fill:blue;stroke:pink;stroke-width:5;opacity:0.2" />
<rect x="140" y="15" rx="20" ry="20" width="50" height="60" style="fill:red;stroke:black;stroke-width:5;opacity:0.5" />
<polygon points="100,100 40,298 190,178 10,178 160,298" style="fill:url(#grad1);stroke:purple;stroke-width:2;opacity:0.5;fill-rule:evenodd;" />
<polygon points="100,100 40,298 190,178 10,178 160,298" style="fill:lime;stroke:purple;stroke-width:2;opacity:0.1;fill-rule:nonzero;" />
<circle cx="50" cy="90" r="40" stroke="black" stroke-width="3" fill="darkred" />
<ellipse cx="150" cy="150" rx="100" ry="10" style="fill:yellow;stroke:purple;stroke-width:2" />
<line x1="90" y1="90" x2="140" y2="140" style="stroke:rgb(255,127,0);stroke-width:2" />
<polygon points="200,50 250,120 160,130" style="fill:url(#grad2);stroke:purple;stroke-width:1" />
<polyline points="220,20 240,25 300,40 380,120 250,140 280,180" style="fill:none;stroke:black;stroke-width:3" />
<text x="34" y="75" fill="red" transform="rotate(30 20,40)">I love SVG</text>
<a xlink:href="http://www.ya.ru" target="_blank"><text x="214" y="220" style="fill:darkred;">
Several lines:<tspan x="214" y="260">First line.</tspan><tspan x="214" y="300">Second line.</tspan></text></a>
</svg></body>
```