



Урок 5

Работа с текстом. Границы в CSS3. Canvas

Работа с шрифтами. Веб-шрифты Google. Размещение текста в несколько колонок. Тени. Рамки. Технология рисования Canvas.

[Введение](#)

[Форматы веб-шрифтов](#)

[Наборы шрифтов](#)

[Веб-шрифты Google](#)

[Размещение текста в нескольких колонках](#)

[Добавление теней к тексту](#)

[Свойство text-overflow](#)

[Свойство word-wrap](#)

[Рамки в CSS3](#)

[Скругление углов с помощью border-radius](#)

[Рамки-изображения border-image](#)

[Ширина рамки-изображения border-image-width](#)

[Ресурс рамки-изображения border-image-source](#)

[Элементы рамки-изображения border-image-slice](#)

[Повтор рамки-изображения border-image-repeat](#)

[Смещение рамки-изображение border-image-outset](#)

[Смещение внешней рамки outline-offset](#)

[Градиентная рамка](#)

[Технология рисования Canvas](#)

[Основы использования Canvas](#)

[Canvas 2D API](#)

[Заливки и границы фигур](#)

[Окружность и круг](#)

[Кривые Безье](#)

[Контуры](#)

[Вставка изображений в Canvas](#)

[Практическое задание](#)

[Используемая литература](#)

Введение

Когда-то веб-мастерам приходилось работать с ограниченным набором шрифтов, пригодных для веб-страниц – речь идет о «безопасных» шрифтах, о которых заведомо известно, что они работают в разных браузерах и в разных операционных системах. Такой подход не годится для современных веб-страниц, где шрифты играют огромную роль в создании общего впечатления о документе.

В CSS3 поддержку сложных шрифтов обеспечивает правило `@font-face`, которое применяется следующим образом:

1. Требуемый шрифт (или, что более вероятно, несколько версий шрифта для поддержки разных браузеров) загружается на сайт.
2. Каждый шрифт регистрируется в таблице стилей с помощью команды `@font-face`.
3. Зарегистрированный шрифт в дальнейшем можно использовать в правилах стиля точно так же, как и обычные веб-шрифты – просто указывая его название.
4. Когда браузер обнаруживает таблицу стилей, в которой используется специальный веб-шрифт, он загружает этот шрифт с сервера в свой кэш для временного хранения страниц и изображений. После этого браузер использует этот шрифт только для данной страницы или сайта. Если этот же шрифт указывается на другой странице, он должен быть зарегистрирован на этой странице и загружен на ее сервер, откуда его может загрузить в кэш браузер.

Форматы веб-шрифтов

Хотя все современные браузеры поддерживают правило `@font-face`, не все они поддерживают одинаковые типы файлов шрифтов. Например, Internet Explorer, который обеспечивает использование `@font-face` в течение многих лет, поддерживает только файлы типа EOT (Embedded OpenType). Этот формат предоставляет ряд преимуществ: например, в нем используется сжатие для уменьшения объема файла шрифтов, а также применяется строгое лицензирование для веб-сайтов, чтобы шрифт нельзя было украсть с одного сайта и использовать на другом.

Но формат EOT никогда не пользовался большой популярностью и не используется никакими другими браузерами. Вместо него браузеры работают с более знакомыми стандартами шрифтов, применяемыми в компьютерных приложениях – TTF (TrueType) и OTF (OpenType PostScript). Кроме этого, существуют еще два типа отображения шрифтов – SVG и WOFF. В таблице ниже дано краткое описание всех этих форматов шрифтов:

Форматы внедряемых шрифтов:

Формат	Описание	Используется в
TTF (TrueType), OTF (OpenType PostScript)	Распространенные форматы шрифтов для настольных компьютеров	Firefox (до версии 3.6), Chrome (до версии 6), Safari и Opera
EOT (Embedded OpenType)	Формат, специфичный для продуктов корпорации Microsoft. Не завоевал популярности у браузеров, за исключением Internet Explorer	Internet Explorer (до версии IE 9)
SVG (Scalable Vector Graphics)	Универсальный графический формат, который можно использовать для	Safari Mobile (на iPhone и iPad до iOS 4.2) и мобильные устройства

	создания шрифтов. Дает хорошие, но не отличные результаты – медленно отображается и демонстрирует текст пониженного качества	под управлением операционной системы Android
WOFF (Web Open Font Format)	Возможно, единый формат шрифтов будущего. Поддерживается новыми версиями браузеров	Любой поддерживающий браузер, начиная с Internet Explorer 9, Firefox 3.6 и Chrome 6

Подытожим: если вы хотите использовать возможность @font-face и поддерживать широкий диапазон браузеров, вам нужно предоставлять ваш шрифт в нескольких разных форматах. Шрифт нужно предоставить как минимум в форматах TTF или OTF (любом из них), EOT и SVG. Хорошо (но не обязательно) также предоставить шрифт в перспективном формате WOFF, который может стать более популярным и получить лучшую поддержку в будущем. (Одним из достоинств этого формата является использование сжатых файлов, что сокращает время их загрузки.)

Даже если вы последуете всем приведенным ранее правилам и предоставите все требуемые форматы веб-шрифтов, с ними могут возникнуть такие проблемы:

- Многие шрифты выглядят плохо в операционной системе Windows XP, так как в настройках многих компьютеров с этой операционной системой отключено сглаживание (anti-aliasing), а без применения сглаживания шрифты выглядят очень непривлекательно.
- От пользователей поступают жалобы о проблемах с печатью определенных внедренных шрифтов в некоторых браузерах или в операционных системах.
- Некоторые браузеры страдают проблемой FOUT – Flash of Unstyled Text, вспышка нестилизованного текста. Она происходит, когда внедренный шрифт не успевает загрузиться вовремя, и на странице сначала отображается резервный шрифт, а потом он заменяется встроенным. Эта проблема особенно заметна в старых версиях браузера Firefox. Если вас это беспокоит, можете воспользоваться библиотекой JavaScript от Google, позволяющей разработчику определить резервные стили, которые используются вместо незагруженных внедренных шрифтов, таким образом предоставляя ему полный контроль над воспроизведением текста в любое время.

Хотя эти небольшие проблемы иногда и возникают, большинство из них постепенно решается с выпуском новых версий браузеров. Например, браузер Firefox теперь сводит эффект FOUT к минимуму, ожидая до трех секунд, пока не загрузится внедренный шрифт, прежде чем использовать резервный шрифт.

Наборы шрифтов

На рисунке ниже показаны несколько шрифтов из предоставляемых на сайте [Font Squirrel](#):

Astonished AaBbCcDdEeFfGgHhIiJjKkLlMmNn

Astonished Misprinted Type 1 Style

DOWNLOAD TTF (OFFSITE)

BEARPAW ABCDEFGHIJKLMNOPQRSTUVWXYZ

Bearpaw Dennis Anderson 1 Style

DOWNLOAD TTF

Belligerent Madness AaBbCcDdEe

Belligerent Madness Font Monkey 1 Style

DOWNLOAD TTF

Black Casper AaBbCcDdEeFfGgHhIiJj

BlackCasper AllencHIU cHIU 1 Style

DOWNLOAD TTF

BLOODY ABCDEFGHIJKLMNOPQRSTUVWXYZ

Hot Today
Hot Web Fonts
Almost Free

Languages
Foundries

FIND FONTS

CLASSIFICATIONS

Display 38
Typewriter 11
Handdrawn 10
Stencil 8
Novelty 5
Serif 4
Script 2
Dingbat 2
Calligraphic 1

TAGS

Distressed 58
Rough 49
Display 44
Sans Serif 21
All Caps 20
Graffiti 14
Handwritten 13
Serif 12
Typewriter 11
Brushed 9
Stencil 8
Novelty 7
Decorative 7
Casual 7
Calligraphic 6
Paint 5
Industrial 5

Grunge
Show More Tags

Рисунок 1. Пример выбора шрифтов с сайта Font Squirrel

На сайте Font Squirrel можно скачать нескольких сотен высококачественных шрифтов, организованных в разделы по категориям (такие, как, например, Calligraphic, Grunge или Retro). Все эти шрифты бесплатны для использования как на персональном компьютере для создания документов, так и на веб-страницах.

Набор шрифтов загружается в виде ZIP-архива, который содержит несколько файлов шрифта в разных форматах. Чтобы использовать шрифт на своей веб-странице, файлы шрифта разных форматов нужно загрузить на веб-сервер в папку этой веб-страницы. После этого шрифт нужно зарегистрировать, чтобы он был доступным для использования в таблице стилей. Регистрация выполняется с помощью правила `@font-face` в начале таблицы стилей, которое выглядит следующим образом:

```
@font-face {  
  font-family: 'MetrophobicRegular';  
  src: url('Metrophobic-webfont.eot');  
  src: local('Metrophobic'),  
       url('Metrophobic-webfont.eot?#iefix') format('embedded-opentype'),  
       url('Metrophobic-webfont.woff') format('woff'),  
       url('Metrophobic-webfont.ttf') format('truetype'),  
       url('Metrophobic-webfont.svg#MetrophobicRegular') format('svg');  
  font-weight: normal;  
  font-style: normal;  
}
```

Что делает этот код и что здесь нужно учесть?

- Выражение `@font-face` регистрирует шрифт для его дальнейшего применения в таблице стилей.
- Шрифту можно присвоить любое название.

- Первым должен быть указан формат EOT, так как следующая часть правила сбивает с толку Internet Explorer, и тот не обращает внимания на остальные форматы. Функция таблицы стилей url() указывает браузеру загрузить файл с указанного URL. Если шрифт размещен в одной папке с веб-страницей, то здесь можно просто указать название файла.
- Функция local() указывает браузеру название шрифта, и если этот шрифт установлен на компьютере посетителя веб-страницы, браузер использует локальный шрифт. Но в редких случаях это может вызвать проблемы. Например, в зависимости от того, где установлен шрифт на компьютере посетителя, компьютеры Mac OS X могут вывести предупреждение о нарушении безопасности, или же может загрузиться другой шрифт с таким же названием, как и ваш. По этой причине веб-разработчики иногда указывают явно несуществующее имя файла, чтобы браузер не нашел локального шрифта. В качестве простого имени такого типа можно использовать какой-либо бессмысленный символ.
- В конце нужно сообщить браузеру о других файлах шрифтов, которые он может использовать. Если есть файл шрифта типа WOFF, укажите его первым, так как этот формат обеспечивает наилучшее качество шрифта. Следующим укажите файл шрифта формата TTF или OTF, а самым последним – файл формата SVG.

Зарегистрировав веб-шрифт с помощью функции @font-face, вы можете использовать его в любой таблице стилей. Для этого используется уже знакомое нам свойство font-family, значение которого – название семейства шрифтов, зарегистрированного с помощью функции @font-face (в строке 2). Вот пример использования этого шрифта в правиле таблицы стилей:

```
body {
    font-family: Amerika;
}
```

Это правило применяет веб-шрифт ко всей странице, хотя область его применения, конечно же, можно ограничить определенными элементами или применить классы. Но шрифт нужно в обязательном порядке зарегистрировать до того, как использовать его в правиле таблицы стилей. Если выполнить эти шаги в обратном порядке, шрифт не будет работать как надо.

Веб-шрифты Google

Еще один источник бесплатных веб-шрифтов — это служба Google Fonts. Она определяет браузер посетителя страницы и автоматически отправляет файл шрифта нужного формата,

Чтобы использовать шрифт Google на своих страницах, выполните такую последовательность шагов:

1. Откройте в браузере страницу Google Fonts (<https://fonts.google.com/>). Откроется страница, содержащая длинный список имеющихся в наличии шрифтов.
2. Настройте опции поиска. Если вы знаете название нужного вам шрифта, введите его в поле поиска. В противном случае нужно будет просмотреть все шрифты, прокручивая страницу вверх. Латинских шрифтов много, и прокрутка списка займет много времени. Чтобы ускорить поиск, можно отсортировать и отфильтровать список.
3. Найдя подходящий шрифт, щелкните по знаку «+». Откроется новое окно, содержащее подробное описание шрифта.
4. Если вы решите использовать этот шрифт, то вы можете скачать его к себе и использовать его так, как показано в предыдущем примере, или использовать код из ссылки на таблицу стилей, которую нужно вставить в разметку вашей веб-страницы, и примера правила таблицы стилей, применяющего шрифт.
5. Вставьте на свою веб-страницу ссылку на таблицу стилей. Например:

```
<link href="https://fonts.googleapis.com/css?family=Underdog" rel="stylesheet">
```

Эта таблица стилей регистрирует шрифт с помощью функции `@font-face`, и вам уже не надо делать это самому. Но лучше всего то, что Google предоставляет сами файлы шрифтов, и загружать их на свой веб-сайт вам тоже не надо.

Используйте шрифт, обращаясь к нему по его названию, в любое время. Например, далее приводится правило для применения должным образом зарегистрированного шрифта *Seviche One* в заголовках первого уровня, предоставляющее резервный шрифт на случай, если браузер не сможет загрузить основной шрифт:

```
h1 {  
    font-family: 'Underdog', arial, serif;  
}
```

Размещение текста в нескольких колонках

В CSS3 был добавлен абсолютно новый модуль для размещения текста в нескольких колонках, чтобы более компактно и без потерь для читаемости разместить длинный текст. Создание нескольких колонок текста не требует почти никаких усилий и может быть выполнено двумя способами. Первый подход – установить количество требуемых колонок с помощью свойства `column-count`, как показано в следующем коде:

```
.Content {  
    text-align: justify;  
    column-count: 3;  
}
```

Для браузеров Firefox, Chrome и Safari это свойство используется с префиксом разработчика браузера:

```
.Content {  
    text-align: justify;  
    -moz-column-count: 3;  
    -webkit-column-count: 3;  
    column-count: 3;  
}
```

Указание точного числа колонок лучше всего подходит для веб-страниц с фиксированной версткой. Но если верстка резиновая (то есть горизонтальные размеры элементов меняются в соответствии с размерами окна браузера), ширина колонок может оказаться слишком большой, и текст в них будет трудно читать. Во избежание такой проблемы лучше не устанавливать точное число колонок, а дать указание браузеру, каким должен быть размер каждой колонки, используя свойство `column-width`, как показано далее:

```
.Content {
    text-align:justify;
    -moz-column-width: 10em;
    -webkit-column-width: 10em;
    column-width: 10em;
}
```

Таким образом браузер может создать такое количество колонок, которое необходимо для заполнения имеющегося пространства.

Размер колонок можно указывать в пикселах, но лучше использовать для этого em-единицы, которые адаптируются к текущему размеру шрифта. Таким образом, если посетитель веб-страницы увеличит размер шрифта в браузере, ширина колонки возрастет пропорционально размеру шрифта. В пикселах одна em-единица приблизительно вдвое больше размера шрифта. Например, для 12-пиксельного шрифта одна em-единица будет равна 24 пикселям.

С помощью CSS3 свойства `column-gap` Вы можете установить величину отступа между столбцами текста:

```
div {
    column-count:4;
    column-gap:50px;
}
```

С помощью свойства `column-rule` Вы можете задать ширину, цвет и стиль оформления пространства между столбцами:

```
div {
    column-count:4;
    column-rule:2px dotted #7F0055;
}
```

Добавление теней к тексту

С помощью нового CSS3 свойства `text-shadow` вы можете добавлять к тексту элементов тени (к тексту одного элемента может быть добавлено одновременно несколько теней).

При задании тени для текста необходимо указать величину смещения тени от текста по горизонтали и вертикали (она может быть отрицательной), а также радиус размытия и цвет тени.

```
#shadow1 {
    text-shadow:3px 2px #FFAE00;
}
#shadow2 {
    text-shadow:1px 1px 10px #FFAE00;
}
#shadow3 {
    text-shadow:2px 2px 2px #FFAE00, 2px 2px 15px #1435AD;
}
```


Свойство text-overflow

В CSS3 было добавлено новое свойство text-overflow, которое позволяет указать, что должно случиться с текстом, вышедшем за пределы границ элемента.

```
#wrap1 {  
  text-overflow:ellipsis;  
  overflow:hidden;  
}  
#wrap2 {  
  text-overflow:clip;  
  overflow:hidden;  
}
```

Свойство word-wrap

С помощью нового CSS3 свойства word-wrap Вы можете указать, что длинные слова, выходящие за пределы границ элемента, должны разделяться и переноситься на новую строку.

```
#wrap2 {  
  word-wrap:break-word;  
}
```

В таблице ниже приведены основные свойства CSS для текста.

Свойство	Описание	Введено в
@font-face	Позволяет подключить к веб-страницам произвольные шрифты	CSS3
font	Позволяет установить все свойства шрифта (font-family, font-size, font-style, font-variant, font-weight) за одно определение	CSS1
font-family	Позволяет установить шрифт текста	CSS1
font-size	Позволяет установить размер текста	CSS1
color	Позволяет установить цвет текста	CSS1
text-shadow	Позволяет привязать тень (или несколько теней) к тексту элемента	CSS3
text-decoration	Позволяет оформить текст элемента	CSS1
text-align	Позволяет определить горизонтальное выравнивание текста	CSS1
letter-spacing	Позволяет определить расстояние между символами текста	CSS1
word-spacing	Позволяет определить расстояние между словами текста	CSS1

<u>line-height</u>	Позволяет установить высоту строк	CSS1
<u>font-style</u>	Позволяет установить стиль шрифта элемента	CSS1
<u>font-variant</u>	Позволяет отобразить текст элемента капителью	CSS1
<u>font-weight</u>	Позволяет установить толщину шрифта	CSS1
<u>text-overflow</u>	Позволяет указать как должен отображаться текст вышедший за пределы границ элемента	CSS3
<u>vertical-align</u>	Позволяет установить вертикальное выравнивание текста	CSS1
<u>text-transform</u>	Позволяет управлять регистром символов в тексте	CSS1
<u>text-indent</u>	Позволяет установить величину отступа первого символа текста	CSS1
<u>text-justify</u>	Позволяет установить алгоритм выравнивания для свойства "text-align:justify"	CSS3
<u>word-wrap</u>	Позволяет указать, должны ли длинные слова, выходящие за пределы родительского элемента, разбиваться и переноситься на новую строку или нет	CSS3
<u>white-space</u>	Позволяет установить, как должны оформляться пробелы в тексте элемента	CSS1
<u>quotes</u>	Позволяет установить, как должны оформляться кавычки вставленные тэгом <q>	CSS1
<u>direction</u>	Позволяет установить направление письма текста	CSS1

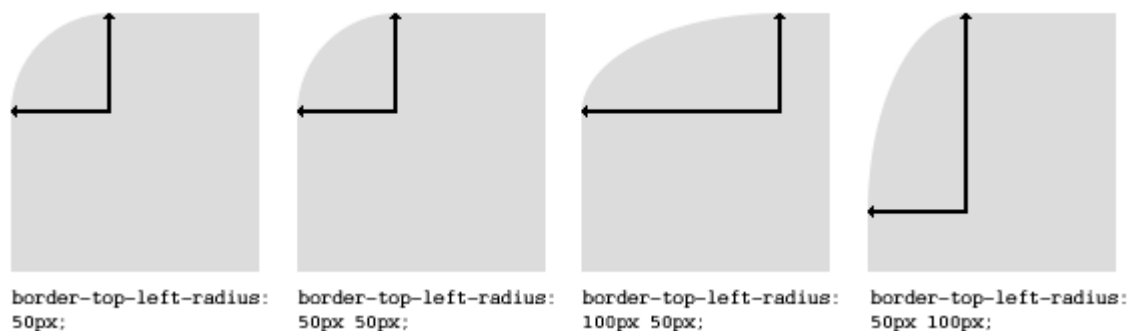
Рамки в CSS3

CSS3-рамка дополняет возможности форматирования границ элементов с помощью свойств, позволяющих скруглить углы элемента, а также использовать изображения для оформления границ.

Возможностей для управления внешним видом границ и рамок в CSS3 довольно много:

1. Скругление углов с помощью border-radius.
2. Рамки-изображения border-image.
 - a. Ширина рамки-изображения border-image-width.
 - b. Источник изображения для рамки border-image-source.
 - c. Элементы рамки-изображения border-image-slice.
 - d. Повтор рамки-изображения border-image-repeat.
 - e. Смещение рамки-изображения border-image-outset.
3. Смещение внешней рамки outline-offset.
4. Градиентная рамка.

Скругление углов с помощью border-radius



Свойство позволяет скруглить углы строчных и блочных элементов. Кривая для каждого угла определяется с помощью одного или двух радиусов, определяющих его форму – круга или эллипса. Радиус распространяется на весь фон, даже если элемент не имеет границ. Точное положение текущей определяется с помощью свойства background-clip.

Свойство border-radius позволяет скруглить все углы одновременно, а с помощью свойств border-top-left-radius, border-top-right-radius, border-bottom-right-radius, border-bottom-left-radius можно скруглить каждый угол отдельно.

Если задать два значения для свойства border-radius, то первое значение скруглит верхний левый и нижний правый угол, а второе – верхний правый и нижний левый.

Значения, заданные через /, определяют горизонтальные и вертикальные радиусы. Свойство не наследуется.

border-radius (border-top-left-radius, border-top-right-radius, border-bottom-right-radius, border-bottom-left-radius)	
Значения	Описание
Длина	Значения задаются в единицах длины
%	Значения задаются в процентах от длины и ширины сторон элемента
initial	Задаёт свойству значение по умолчанию
inherit	Наследует значение свойства от родительского элемента

```
div {width: 100px; height: 100px; border: 5px solid;}
.r1 {border-radius: 0 0 20px 20px;}
.r2 {border-radius: 0 10px 20px;}
.r3 {border-radius: 10px 20px;}
.r4 {border-radius: 10px/20px;}
.r5 {border-radius: 5px 10px 15px 30px/30px 15px 10px 5px;}
.r6 {border-radius: 10px 20px 30px 40px/30px;}
.r7 {border-radius: 50%;}
.r8 {border-top: none; border-bottom: none; border-radius: 30px/90px;}
.r9 {border-bottom-left-radius: 100px;}
```

```
.r10 {border-radius: 0 100%;}
.r11 {border-radius: 0 50% 50% 50%;}
.r12 {border-top-left-radius: 100% 20px; border-bottom-right-radius: 100% 20px;}
```

Какие в результате получатся скругления, можно увидеть на рисунке 1:

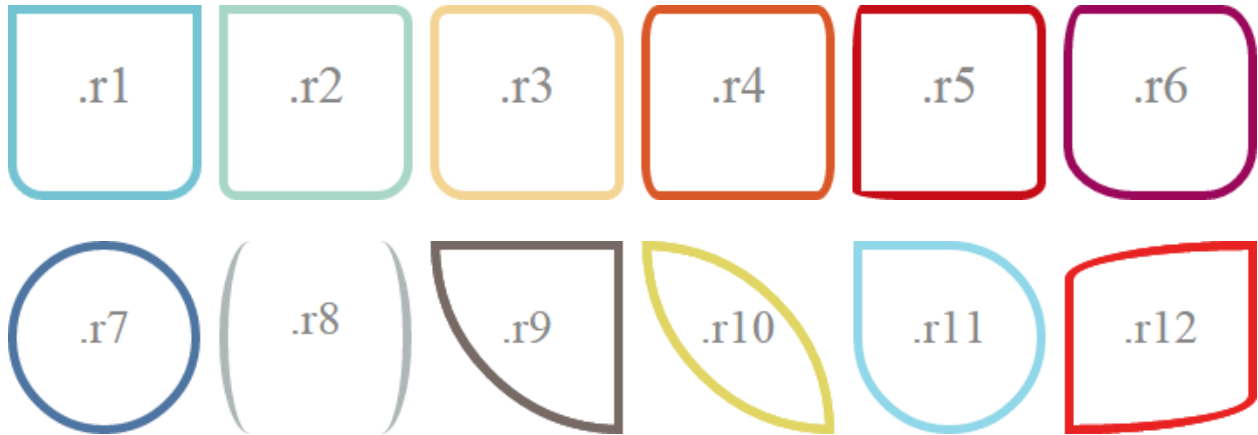
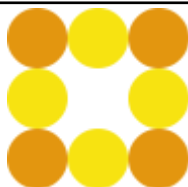


Рисунок 1. Примеры скругления углов блока

Рамки-изображения border-image

Свойство позволяет устанавливать изображение в качестве рамки элемента. Основное требование к изображению — оно должно быть симметричным. Свойство включает в себя следующие значения: {border-image: width source slice repeat outset;}.

border-image	
Значения	Описание
Краткая запись	Устанавливает рамку-изображение с помощью одного свойства, которое представляет собой краткую запись свойств border-image-source, border-image-slice, border-image-width, border-image-outset и border-image-repeat. Значения по умолчанию: {border-image: 1 none 100% stretch 0;}
initial	Задаёт свойству значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.



При помощи данного изображения можно получить такие рамки для элемента:

```
/* пример 1 */
div {
width: 260px; height: 100px;
border-image-width: 15px;
```

```
border-image-source: url(http://html5book.ru/images/border_round.png);
border-image-slice: 30;
border-image-repeat: stretch;
}
/* Пример 2 */
div {
width: 260px; height: 100px;
border-image-width: 15px;
border-image-source: url(http://html5book.ru/images/border_round.png);
border-image-slice: 30;
border-image-repeat: round;
}
```

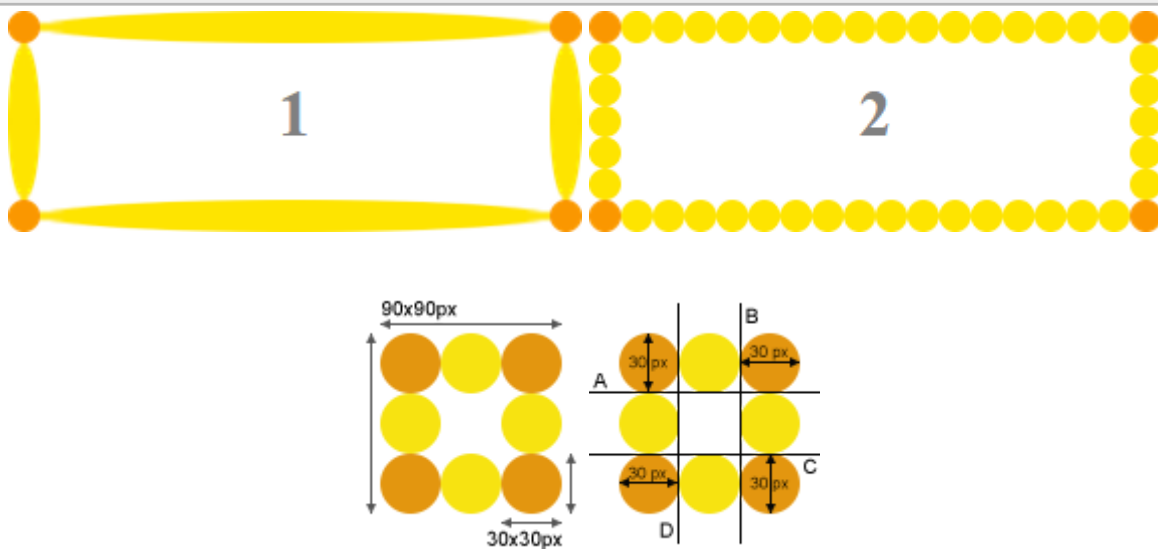


Рисунок 2. Пример оформления блоков с помощью изображения

Срезы А – В – С – D образуют углы рамки, а часть рисунка, расположенная между ними, заполняет оставшееся пространство рамки в соответствии с заданным значением свойства border-image-repeat.

Размер угловой части (в данном примере это число 30) задается с помощью значения свойства border-image-slice.

Ширина рамки-изображения border-image-width

Свойство задает ширину изображения для границы элемента. Если ширина не задана, то по умолчанию она равна 1.

border-image-width	
Значения	Описание
Длина	Устанавливает ширину рамки в единицах длины – px/em. Можно задавать от одного до четырех значений одновременно. Если задано одно значение, то ширина всех сторон рамки одинакова, два значения задают ширину верхней/нижней и правой/левой и т. д.
Число	Числовое значение, на которое умножается значение border-width

%	Ширина рамки элемента вычисляется относительно размера изображения: горизонтальные – относительно ширины, вертикальные – относительно высоты
auto	Соответствует значению border-image-slice
initial	Задаёт свойству значение по умолчанию
inherit	Наследует значение свойства от родительского элемента

Синтаксис:

```
div {
border-image-source: url(border.png);
border-image-width: 30px;
}
```

Ресурс рамки-изображения border-image-source

Свойство задаёт путь к изображению, которое будет использоваться для границы блока.

border-image-source	
Значения	Описание
none	Отсутствие изображения для рамки. Значение по умолчанию
url(url)	Относительный или абсолютный путь к изображению
initial	Задаёт свойству значение по умолчанию
inherit	Наследует значение свойства от родительского элемента

Синтаксис:

```
div {border-image-source: url(border.png);}
```

Элементы рамки-изображения border-image-slice

Свойство определяет размер частей изображения, используемых для оформления границ элемента, и делит изображение на девять частей: четыре угла, четыре края между углами и центральную часть.

border-image-slice	
Значения	Описание
Число	<p>Размер частей рамки можно задавать с помощью одного, двух, трех или четырех значений.</p> <p>Одно значение устанавливает границы одинакового размера для каждой стороны элемента.</p> <p>Два значения: первое определяет размер верхней и нижней границ, второе – правой и левой.</p>

	Три значения: первое определяет размер верхней границы, второе – правой и левой, а третье – нижней. Четыре значения: определяет размеры верхней, правой, нижней и левой границ. Числовое значение – это количество px
%	Размеры границ рассчитываются относительно размера изображения. Горизонтальные относительно ширины, вертикальные – относительно высоты
fill	Значение указывается вместе с числом или процентным значением. Если оно задано, изображение не обрезается внутренним краем рамки, а заполняет также область внутри рамки
initial	Задаёт свойству значение по умолчанию
inherit	Наследует значение свойства от родительского элемента

Синтаксис:

```
div {border-image-slice: 50 20;}
```



border.png
размер 200x200px

```
div {
width: 200px;
height: 200px;
border-image-width: 40px;
border-image-source:
url(border.png);
border-image-slice: 40;
border-image-repeat: round;
}
```



<div></div>

Рисунок 3. Пример задания срезов изображения

Повтор рамки-изображения border-image-repeat

Свойство управляет заполнением пространства между углами рамки фоновым изображением. Можно задавать его как с помощью одного, так и с помощью пары значений.

border-image-repeat	
Значения	Описание
stretch	Растягивает изображение на все пространство. Значение по умолчанию
repeat	Повторяет заполняющую часть изображения, при этом видны места стыков с угловой частью, и если длины образца не хватает, он растягивается
round	Наиболее точно заполняет промежуток между углами рамки, дублируя заполняющую часть изображения, при этом может образовывать стыки посередине стороны рамки

space	Действует аналогично repeat
initial	Задаёт свойству значение по умолчанию
inherit	Наследует значение свойства от родительского элемента

Синтаксис:

```
div {
border-image-source: url(border.png);
border-image-repeat: repeat;
}
```



Рисунок 4. Пример повтора центральной рамки изображения

Смещение рамки-изображение border-image-outset

Свойство позволяет переместить изображение-рамку за пределы границ элемента на указанную длину. Может задаваться с помощью как одного, так и четырёх значений.

border-image-outset	
Значения	Описание
длина	Отступ рамки-изображения задается с помощью любого положительного числа, указанного в px или em.
число	Числовое значение, на которое умножается border-width.
initial	Задаёт свойству в значение по умолчанию.
inherit	Наследует значение свойства от родительского элемента.

Синтаксис:

```
div {
border-image-source: url(border.png);
border-image-outset: 10px;
}
```

Смещение внешней рамки outline-offset

Свойство задаёт расстояние между границей элемента border и внешней границей, созданной с помощью свойства outline.

outline-offset	
Значения	Описание
Длина	Задаёт расстояние с помощью единиц длины – px/em. Отрицательное значение отображает рамку внутри элемента, положительное – снаружи элемента
initial	Задаёт свойству значение по умолчанию
inherit	Наследует значение свойства от родительского элемента

Синтаксис:

```

/*Рисунок 1:*/
img {
border: 1px solid pink;
outline: 1px dashed grey;
outline-offset: 3px;
}
/*Рисунок 2:*/
img {
border-width: 1px 10px;
border-style: solid;
border-color: pink;
outline: 1px dashed grey;
outline-offset: 3px;
}
/*Рисунок 3:*/
img {
border: 3px inset pink;
outline: 1px dashed grey;
outline-offset: 1px;
}

```

Градиентная рамка

Значением border-image может выступать не только изображение, но и градиентная заливка.

В качестве одного из цветов выступает transparent. Таким способом можно задавать границы сразу для всех сторон элемента или по отдельности для каждой стороны. Толщина рамки регулируется свойством border-width.

Синтаксис HTML:

```
<div class="wrap"><div class="gradient"></div></div>
```

Синтаксис CSS:

```
.wrap {
```

```

height: 200px;
padding: 25px;
background: #00E4F6;
}
.gradient {
height: 150px;
width: 50%;
margin: 0 auto;
border-width: 10px;
border-image: linear-gradient(to right, transparent 0%, #ADF2F7 100%);
border-image-slice: 1;
}

```

Есть и другой вариант задания градиента.

Синтаксис HTML:

```

<div class="wrap"><div class="gradient"></div></div>

```

Синтаксис CSS:

```

{box-sizing:border-box;}
.wrap {
height: 200px;
padding: 25px;
}
.gradient {
height: 150px;
width: 50%;
margin: 0 auto;
border: 10px solid transparent;
border-image: 10 repeating-linear-gradient(45deg, #A7CECC, #A7CECC 10px,
transparent 10px, transparent 20px, #F8463F 20px, #F8463F 30px, transparent
30px, transparent 40px);
}

```



Технология рисования Canvas

HTML5 Canvas предоставляет способ вывода графики и рисования с использованием JavaScript. Для каждого элемента canvas можно использовать контекст, в котором нужно вызвать команды JavaScript для рисования на Canvas. Браузеры могут реализовывать несколько контекстов элемента canvas и предоставлять различные API для рисования. Следует также помнить, что рисование происходит в растровой форме

На этом уроке вы узнаете, как использовать контекст элемента canvas, основные функции для работы с Canvas, включая линии, примитивы фигур, изображения, текст и другие возможности. Предполагается, что вы владеете основами JavaScript.

Основы использования Canvas

Чтобы создать Canvas-контекст, достаточно просто добавить элемент `<canvas>` в HTML-документ:

```
<canvas id="myCanvas" width="300" height="150">  
Альтернативное содержимое, которое будет показано, если браузер не поддерживает  
Canvas. В данном случае можно написать сообщение «Ваш браузер не поддерживает  
CANVAS»  
</canvas>
```

Нужно добавить идентификатор к элементу canvas, чтобы потом обратиться к нему в JavaScript, также необходимо задать атрибуты `width` и `height` для определения ширины и высоты элемента canvas. Размер холста по умолчанию – 150×300 пикселей.

Для рисования внутри элемента canvas, нужно использовать JavaScript. Сначала нужно найти созданный тег canvas с помощью функции `getElementById`, а потом инициализировать нужный контекст. Как только это будет сделано, можно начинать рисование на холсте, используя доступные API-команды выбранного контекста. Следующий скрипт рисует простой прямоугольник на холсте:

```
<script>  
// Получаем ссылку на элемент canvas по идентификатору  
var elem = document.getElementById('myCanvas');  
// Проверка свойства и метода на доступность для обратной совместимости со  
старыми браузерами  
if (elem && elem.getContext) {  
// Получаем 2D-контекст  
// Для каждого элемента можно инициализировать только один контекст  
var context = elem.getContext('2d');  
if (context) {  
// Рисуем прямоугольник, задав координаты (x,y), а также его ширину и высоту  
context.fillRect(0, 0, 150, 100);  
}  
}  
</script>
```

Canvas 2D API

Заливки и границы фигур

С помощью свойств `fillStyle` и `strokeStyle` настраиваются цвета, используемые для заливки и линий объектов. Значения цветов, используемые в этих методах, такие же, как и в CSS: шестнадцатеричные коды (`#F5E6AB`), `rgb()`, `rgba()` или даже `hsla()`, если браузер поддерживает такой способ задания цвета.

При помощи метода `fillRect` рисуется прямоугольник с заливкой. С помощью метода `strokeRect` рисуется прямоугольник только с границами, без заливки. Для очистки части холста используется метод `clearRect`. Три этих метода используют одинаковый набор аргументов: `x`, `y`, `width`, `height`. Первые два аргумента задают координаты (`x,y`), а следующие два – ширину и высоту прямоугольника.

Для изменения толщины линий можно использовать свойство `lineWidth`.

Окружность и круг

```
context.beginPath();
context.arc(75, 75, 40, 0, Math.PI*2, true);
context.closePath();
context.fillStyle = "#FFA500";
context.fill(); // Если нужен круг, можно залить окружность
```

Кривые Безье

Для создания кривых Безье в HTML5 Canvas используется метод `bezierCurveTo()`. Кривые Безье задаются с помощью начальной, двух контрольных и конечной точек. В отличие от квадратичных кривых, кривые Безье в HTML5 Canvas определяются двумя контрольными точками вместо одной, позволяя создавать кривые с более сложным очертанием.

Метод `bezierCurveTo()` выглядит следующим образом:

```
context.bezierCurveTo(controlX1, controlY1, controlX2, controlY2, endX, endY);
```

Пример рисования кривой Безье в HTML5 Canvas:

```
<canvas id="myCanvas3" width="500" height="250">
  <p>&nbsp;</p>
  <p>Альтернативное содержимое, которое будет показано, если браузер не
поддерживает Canvas. В данном случае можно написать сообщение «Ваш браузер не
поддерживает Canvas».</p>
</canvas>
<script>
  var canvas = document.getElementById("myCanvas3");
  var context = canvas.getContext("2d");
  context.moveTo(188, 130);
  var controlX1 = 0;
  var controlY1 = 0;
  var controlX2 = 480;
  var controlY2 = 50;
```

Контуры

Вставка изображений в Canvas

20

- Три аргумента. Базовое использование метода `drawImage` включает один аргумент для указания изображения, которое необходимо вывести на холсте, и два аргумента для задания координат.
- Пять аргументов. Используются предыдущие три аргумента и еще два, задающие ширину и высоту вставляемого изображения (в случае, если вы хотите изменить размеры изображения при вставке).
- Девять аргументов. Используются предыдущие пять аргументов и еще четыре: два для координат области внутри исходного изображения и два для ширины и высоты области внутри исходного изображения для обрезки изображения перед вставкой в `Canvas`.

Чтобы передать объект изображения, его сначала нужно получить. В HTML5 есть три разных способа получения объекта изображения. Первый подход – создать его самостоятельно попиксельно с помощью метода `createImageData()`. Но этот подход очень трудоемкий и медленный.

Второй подход – использовать уже имеющийся в разметке элемент ``. Например, у нас есть следующая разметка:

```

```

Тогда изображение можно вставить на холст с помощью кода:

```
var img = document.getElementById("photo");
context.drawImage(img, 10, 10);
```

Можно создать объект изображения и загрузить изображение из отдельного файла. Недостаток этого подхода состоит в том, что изображение нельзя использовать с методом `drawImage()` до тех пор, пока оно полностью не загрузится. Чтобы не было проблем нужно подождать, пока не выполнится событие загрузки изображения `onLoad`, прежде чем пытаться выполнять какие-либо операции с ним.

Нам нужно отобразить на холсте изображение `img.jpg`. Теоретически это можно сделать такой последовательностью операций:

```
// Создаем объект изображения
var img = new Image();
// Загружаем файл изображения
img.src = "face.jpg";
// Прорисовываем изображение (этот шаг может не выполниться,
// т.к. изображение, возможно, еще не загрузилось)
context.drawImage(img, 10, 10);
```

Проблема в том, что установка атрибута `src` начинает загрузку изображения, но код продолжает выполняться дальше, не ожидая завершения загрузки. Правильно будет использовать следующий код:

```
// Создаем объект изображения
var img = new Image();
// Привязываем функцию к событию onload
// Это указывает браузеру, что делать, когда изображение загружено
img.onload = function() {
    context.drawImage(img, 10, 10);
};
```

```
// Загружаем файл изображения
img.src = "img.jpg";
```

Пример вставки изображения:

```
<canvas id="myCanvas5" width="600" height="600">
  <p>&nbsp;</p>
  <p>Альтернативное содержимое, которое будет показано, если браузер не
поддерживает Canvas. В данном случае можно написать сообщение «Ваш браузер не
поддерживает Canvas».</p>
</canvas>
<script>
  var canvas = document.getElementById("myCanvas5");
  var context = canvas.getContext("2d");
  // Создаем объект изображения
  var img = new Image();
  // Привязываем функцию к событию onload
  // Это указывает браузеру, что делать, когда изображение загружено
  img.onload = function() {
  // Три аргумента: элемент img или canvas, координаты для вывода на холст
  // (x,y).
  context.drawImage(img, 10, 10);
  // Пять аргументов: элемент img или canvas, координаты для вывода на холст
  // (x,y), ширина и высота для изменения размеров перед выводом
  context.drawImage(img, 10, 315, 500, 100);
  // Девять аргументов: элемент img или canvas, координаты, ширина и высота для //
  обрезки изображения, координаты для вывода на холст (x,y), ширина и высота
  // для изменения размеров перед выводом
  context.drawImage(img, 120, 120, 100, 100, 10, 420, 150, 100);
  };
  // Загружаем файл изображения
  img.src = "image/5.png";
</script>
```

Практическое задание

1. Приступаем к верстке нового макета ([Макет в Figma](#)).
2. Не забываем про семантическую структуру (данный макет сложнее чем предыдущий).
3. Сверстать, сколько успеете (это финальная работа).
4. Добавить все шрифты, использованные в макете.
5. * Использовать оба представленных в лекции способа подключения шрифтов.

Используемая литература

1. <http://www.wisdomweb.ru/>.
2. <https://html5book.ru/>.
3. Гоше Х. HTML5. Для профессионалов. СПб.: Питер, 2013. – 496 с.
4. Хоган Б. HTML5 и CSS3. Веб-разработка по стандартам нового поколения. СПб.: Питер, 2012.

5. Макфарланд Д. Большая книга CSS3. СПб.: Питер, 2014.