Разработка веб-сайта с играми.

Разработка интерактивных игр на сайте.

HTML+CSS=САЙТ

Мы с вами познакомились с языком программирования JavaScript, но все наши программы выглядили, мягко говоря, не очень красиво. Для того, чтобы сделать веб-страницу более насыщенной давайте познакомимся с языком разметки веб-страниц HTML и языком описания внешнего вида документа CSS. Начнем с HTML.

HTML

На самом-то деле, все программы, которые мы с вами писали до этого, были не совсем правильными веб-страницами. Но браузеры так устроены, что если даже веб-страница написана не правильно, он старается отобразить информацию, так как считает нужным. Давайте посмотрим, как мы можем указать браузеру, что и как выводить.

Наберите в текстовом редакторе следующий текст. Сохраните его под названием: index.html и откройте в браузере просто щелкнув на файле index.html два раза.

```
<!DOCTYPE html>
<html>
    <head></head>
    <body>
        Тело веб-страницы
    </body>
</html>
```

Возможно, что у вас выведется текст, а может быть выводятся непонятные символы. От чего это зависит читайте дальше.

Перед вами шаблон веб-страницы. Веб-страница состоит из тегов - ключевых слов заключенных в угловые скобки. Теги бывают *парные* и *одинарные*.

<!DOCTYPE html> - это указание браузеру, что тип документа, с которым ему придется работать html. Это одинарный тег. Он не требует закрывающего тега.

<html> </html> - это парные теги внутри которых лежит веб-страница

Сама веб страница состоит из двух частей: заголовка и тела.

Заголовок помещается между тегами <head> и </head>. Здесь описывается некоторая служебная информация, которая влияет на отображение веб-страницы.

Тело, то есть сама веб-страница, располагается между тегами <body> и </body>. Большинство текста по описанию веб-страницы помещается сюда.

Давайте добавим в нашу веб-страницу в заголовок указание в какой кодировке (тег <meta charset="utf-8">) вы создаете веб-страницу, заголовок, которые будет отображатся на вкладке браузера, и некоторый произвольный текст.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Иванов Иван</title>
</head>

<body>
    <strong>Папа у Васи силен в математике,</strong><br>
    <em>Учится папа за Васю весь год,</em><br>
    <i>Где это видано, где это слыхано,</i><br>
    <b>Папа решает, а Вася сдает?</b>
</body>
</html>
```

Папа у Васи силен в математике,
Учится папа за Васю весь год,
Где это видано, где это слыхано,
Папа решает, а Вася сдает?

Теги strong и b - делают шрифт жирным, а теги em и i - наклонным. Отличие тега strong от b в том что он не просто делает текст жирным, он с точки зрения поисковых систем придает выделенному тексту больший вес, это свойство отражает его семантическую природу - выделенный текст имеет бОльшую важность. Поначалу количество тегов в HTML может навести тоску, но они легко запоминаются, а специальные редакторы, такие как Sublime Text или Brackets имеют встроенную контекстную подсказку по тегам, что делает запоминание тегов еще более легким.

С HTML немного разобрались. Теперь давайте посмотрим, что такое CSS.

CSS

Для примера добавим простенький CSS прямо в заголовок нашей страницы. Для этого в между тегами head добавим парный тег style, а внутри него код CSS.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Иванов Иван</title>
        <style>
            strong {
                color: red;
            }
            .my_style {
                font-size: 30px;
                color: blue;
            }
        </style>
    </head>

    <body>
        <strong>Папа у Васи силен в математике,</strong><br>
        <em class="my_style">Учится папа за Васю весь год,</em><br>
        <i>Где это видано, где это слыхано,</i><br>
        <b>Папа решает, а Вася сдает?</b>
    </body>
</html>
```

Папа у Васи силен в математике,
Учится папа за Васю весь год,
Где это видано, где это слыхано,
Папа решает, а Вася сдает?

CSS задает стиль отображения информации. Здесь мы продемонстрировали два способа задания стиля. Первый, strong - мы описали без точки вначале, что означает, что этот стиль применяется для всех тегов strong на этой странице. Второй, .my_style с точной в начале - мы создали класс стиля, который потом можно назначать на веб-странице, что мы и сделали назначив этот стиль тексту заключенному между тегами em. Сами же стили довольно простые, в первом случаем мы указали, что нужно выводить красным цветом, во втором, что требуется задать размер шрифта в 30 пикселей и выводить информацию синим цветом.

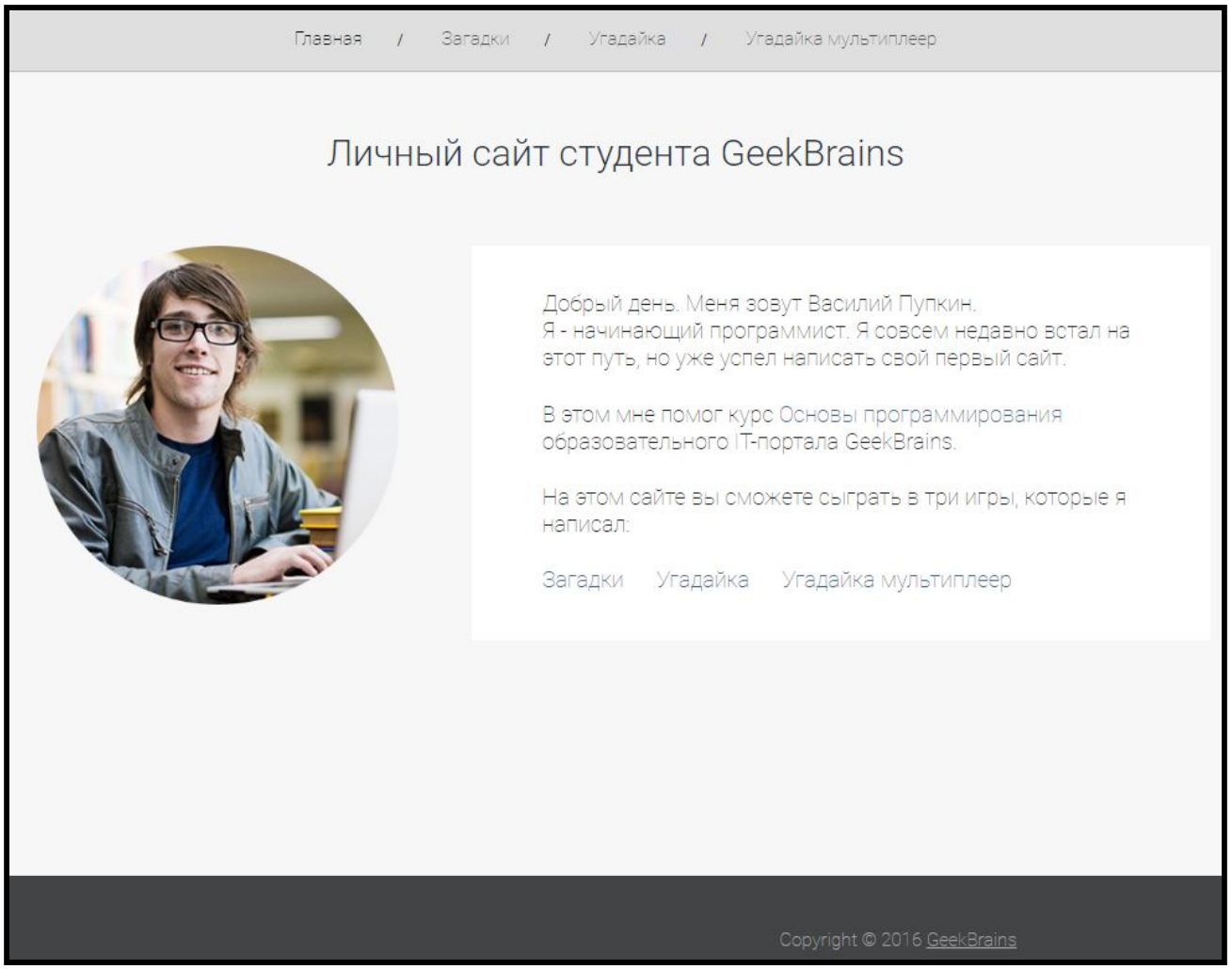
Личный сайт

Когда начинаешь изучать HTML, то наверное одним из самых сложных вопросов является: “Какой сайт создать?” Зачастую, так и не ответив на этот вопрос, обучение заканчивается. В лучшем случае пишется страничка в стиле:

"Красные подчеркнутые наклонные буквы на черном фоне"

Давайте не будем мучиться этим вопросом и создадим свой личный сайт, который будет вашим стартом в мире сайтостроительства.

Что примерно должно получиться:



Этот сайт состоит из нескольких частей: верхней части, центральной части и нижней части. Центральная часть разделена на две части. Интересно, что начинающие веб-строители не могут построить сайт, просто потому, что не очень понимают, из каких частей он будет состоять. Начнем создание.

Центральная часть

Подготовка и вставка картинки

Наш сайт будет состоять из нескольких файлов. Поэтому первым делом давайте создадим папку mysite, в которой эти файлы будут храниться. Для простоты можно разместить её на рабочем столе. Далее создайте в этой папке папку img. В этой папке будут храниться картинки. Поместите туда свою фотографию или другое изображение, которое будет выводиться на месте фотографии. Назовите её photo.png. Если вы умеете обрабатывать фотографии, то сделайте размер картинки 390x390. Если нет, ничего страшного, при вставке картинки мы заставим браузер задать нужный размер.

Для вставки картинки используется одинарный тег img:

```
<img src="img/photo.png" alt="фотография">
```

Атрибут src указывает расположение картинки. Атрибут alt нужен для того, чтобы показать какой текст будет выводиться, если вдруг у пользователя отключен вывод картинок, или картинка будет не доступна.

Должно получиться:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
    </head>
    <body>
        <img src="img/photo.png" alt="фотография">
    </body>
</html>
```

Заголовок

Добавим заголовок. Для этого используется парный тег <h1>. Добавим перед картинкой текст:

```
<h1>Личный сайт студента GeekBrains</h1>
```

Получится:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
    </head>
    <body>
        <h1>Личный сайт студента GeekBrains</h1>
        <img src="img/photo.png" alt="фотография">
    </body>
</html>
```

Личный сайт студента GeekBrains

Информация о себе

Добавим информацию о себе. Для этого используем парный тег <p> и одинарный
. Добавьте под тегом img следующий текст

```
<p>
    Добрый день. Меня зовут Василий Пупкин.
    <br>
    Я - начинающий программист. Я совсем недавно встал на этот путь, но уже успел
написать свой первый сайт.
</p>
<p>
    В этом мне помог курс Основы программирования образовательного IT-портала
GeekBrains.
</p>
<p>
    На этом сайте вы сможете сыграть в три игры, которые я написал:
</p>
```

Ссылки

Парный тег <a> указывает браузеру, что содержимое элемента (это может быть и текст, и картинка) является ссылкой, и, при щелчке по ссылке(то есть по содержимому), браузер попытается перейти по адресу указанном в атрибуте href.

```
<p>
    В этом мне помог курс
```

```
    <a href="https://geekbrains.ru/courses/2">
        Основы программирования
    </a>
    образовательного IT-портала GeekBrains.
</p>
```

Тег <a> поддерживает атрибут target, который указывает в какой вкладке открыть страницу, указанную в ссылке. Например, target="_blank" - указание загрузить страницу в новой вкладке.

```
<p>
    В этом мне помог курс
    <a href="https://geekbrains.ru/courses/2" target="_blank">
        Основы программирования
    </a>
    образовательного IT-портала GeekBrains.
</p>
```

Верхняя часть

Добавьте перед тегом <h1> сразу после <body> следующий текст.

```
<a href="#">Главная</a>
<span>/</span>

<a href="puzzles.html">Загадки</a>
<span>/</span>

<a href="guess.html">Угадайка</a>
<span>/</span>

<a href="guess-2.html">Угадайка мультиплеер</a>
```

Атрибут href="#" - указание, что при щелчке по этой ссылке никуда переходить не нужно.

Парный тег - это, наверное, самый трудный для понимания тег. Он как бы говорит, что внутри заключен кусочек текста. На самом деле в HTML есть еще теги, которые позволяют задать некоторую логику страницы, span один из них. Он не влияет непосредственно на сам вывод, но позволит в дальнейшем элементам, заключенным в этот тег, придать стиль с помощью CSS. С помощью и CSS мы добавим промежутки между пунктами меню

Нижняя часть

В нижнюю часть добавим ссылки на будущие веб-страницы и информацию об авторском праве.

```
<a href="puzzles.html">Загадки</a>
<a href="guess.html">Угадайка</a>
<a href="guess-2.html">Угадайка мультиплеер</a>
Copyright © 2016 <a href="https://geekbrains.ru/" target="_blank">GeekBrains</a>
```

Итоговый код должен получиться примерно таким

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title></title>
    </head>
    <body>
        <a href="#">Главная</a>
        <span>/</span>

        <a href="puzzles.html">Загадки</a>
        <span>/</span>

        <a href="guess.html">Угадайка</a>
        <span>/</span>

        <a href="guess-2.html">Угадайка мультиплеер</a>

        <h1>Личный сайт студента GeekBrains</h1>
        <img src="img/photo.png" alt="фотография">
        <p>
            Добрый день. Меня зовут Василий Пупкин.
            <br>
            Я - начинающий программист. Я совсем недавно встал на этот путь, но
уже успел написать свой первый сайт.
        </p>
        <p>
            В этом мне помог курс
            <a href="https://geekbrains.ru/courses/2" target="_blank">
                Основы программирования
            </a>
            образовательного IT-портала GeekBrains.
        </p>
        <p>
            На этом сайте вы сможете сыграть в три игры, которые я написал:
        </p>

        <a href="puzzles.html">Загадки</a>
        <a href="guess.html">Угадайка</a>
        <a href="guess-2.html">Угадайка мультиплеер</a>
        Copyright © 2016 <a href="https://geekbrains.ru/"
target="_blank">GeekBrains</a>

    </body>
</html>
```

Главная/ Загадки/ Угадайка/ Угадайка мультиплеер

Личный сайт студента GeekBrains

Добрый день. Меня зовут Василий Пупкин.
Я - начинающий программист. Я совсем недавно встал на этот путь, но уже успел написать свой первый сайт.

В этом мне помог курс Основы программирования образовательного IT-портала GeekBrains.

На этом сайте вы сможете сыграть в три игры, которые я написал:

Загадки Угадайка Угадайка мультиплеер Copyright © 2016 GeekBrains

DIV - разделяй и властвуй

Пока сайт выглядит очень невзрачно. Для того, чтобы он стал более красивым, нужно задать стили оформления, но перед этим нужно задать логику сайта с помощью тегов и атрибутов id.

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Личный сайт студента GeekBrains</title>
</head>
<body>
        <div id="header">
            <a href="#">Главная</a>
            <span>/</span>

            <a href="#">Загадки</a>
            <span>/</span>

            <a href="#">Угадайка</a>
            <span>/</span>

            <a href="#">Угадайка мультиплеер</a>
        </div>

        <div id="content">
            <h1>Личный сайт студента GeekBrains</h1>
            <div id="center">
                <img src="img/photo.png" alt="фотография">

                <div id="box_text">
                    <p>
                        Добрый день. Меня зовут <b>Василий Пупкин</b>.
                        Я - <i>начинающий программист</i>. Я совсем недавно встал
на этот путь,
                        но уже успел написать свой первый сайт.
                    </p>

                    <p>
                        В этом мне помог курс <a
href="https://geekbrains.ru/courses/2">Основы программирования</a>
                    </p>

                    <p>
                        На этом сайте вы сможете сыграть в три игры, которые я
написал:
                    </p>
                    <a href="#">Главная</a>
                    <a href="#">Загадки</a>
                    <a href="#">Угадайка</a>
                    <a href="#">Угадайка мультиплеер</a>
                    <br><br>

                </div>
            </div>
        </div>

    <div id="footer">
```

```
        Copyright © 2016 <a href="https://geekbrains.ru/">GeekBrains</a>
    </div>
</body>
</html>
```

Сам тег div, так же как и , не влияет на вывод информации на экран. Но он позволяет разделить веб-страницу на разделы, на которые можно будет потом влиять с помощью CSS кода. Атрибут ID дает название разделам, чтобы по ID можно было обратиться к нужному разделу и изменить его стиль оформления.

Подключим CSS

Добавьте в <head> тег загрузки стиля

```
<link rel="stylesheet" href="style.css">
```

Должно получиться примерно следующее:

```
<head>
    <meta charset="utf-8">
    <title>Личный сайт студента GeekBrains</title>
    <link rel="stylesheet" href="style.css">
</head>
```

Теперь перейдем к созданию файла, который будет влиять на стиль отображения информации на экране

Его величество CSS

Теперь мы подошли к одновременно самой сложной и самой простой части урока. Сложность заключается в большом количестве новой информации, которая на вас обрушится. Простота же заключатся в том, что на первых порах не обязательно понимать всё, что вы увидите. CSS не так сложен, как может показаться на первый взгляд. Для начала поймите, что он будет влиять на отображения страницы.

CSS (Cascading Style Sheets) — каскадные таблицы стилей. Для начала слова “каскадные” и “таблицы” можно не запоминать. Важно только “стили”. Действительно, CSS подменяет ранее созданную логику создания сайтов. Первое время создатели сайтов использовали различные теги и их атрибуты, чтобы указать браузеру, как выводить информацию на экран. Например, раньше только тег <h1> позволял создать заголовок, а тег <font> - задать размер шрифта. Теперь же с помощью тега <h1> указывается браузеру, что текст внутри будет заголовком, а как будет выглядеть этот заголовок мы можем задать с помощью таблицы стилей. Можно сказать, что сейчас основное предназначение тегов - создать логику сайта, а уже CSS управляет тем, как этот текст будет выглядеть в браузере.

На заметку. Если же для каких-то тегов мы не задали стиль, то браузер использует свою внутреннюю таблицу стилей.

Структура CSS

CSS - это уже не HTML, хотя и используется совместно. При описании CSS чаще всего используется следующая структура:

```
Название_стиля
{
    описание1;
    описание2;
...
}
```

Название_стиля - грамотно называть *селектором*, а описание - *свойством.*

body и #header

Перед названием стиля могут стоять значки . или #, которые указывают, к чему требуется применить этот стиль.

Например, для тегов можно задать стиль таким образом:

```
body {
    background-color: #555;
    font-family: sans-serif;
}
```

Это указание, что для тега body сделать фон цветом #555, а шрифт sans-serif

Таким образом можно указать стиль для тегов у которых атрибут id=”header”:

```
#header {
    width: 1300px;
    margin: 0 auto;
    height: 50px;
    text-align: center;
    padding-top: 18px;
}
```

*. padding и margin

* - звездочка - это указание применить стиль для всех тегов

```
* {
    margin: 0;
    padding: 0;
}
```

padding и margin - это отступы. Первое создаёт отступы вокруг содержимого, второе расширяет поле до границы элемента.

Стилизация вложенных тегов

```
#header a {
    color: #4d4d4d;
    font-size: 20px;
    font-weight: 100;
    text-align: left;
    text-decoration: none;
    height: 30px;
    line-height: 30px;
    display: inline-block;
}
```

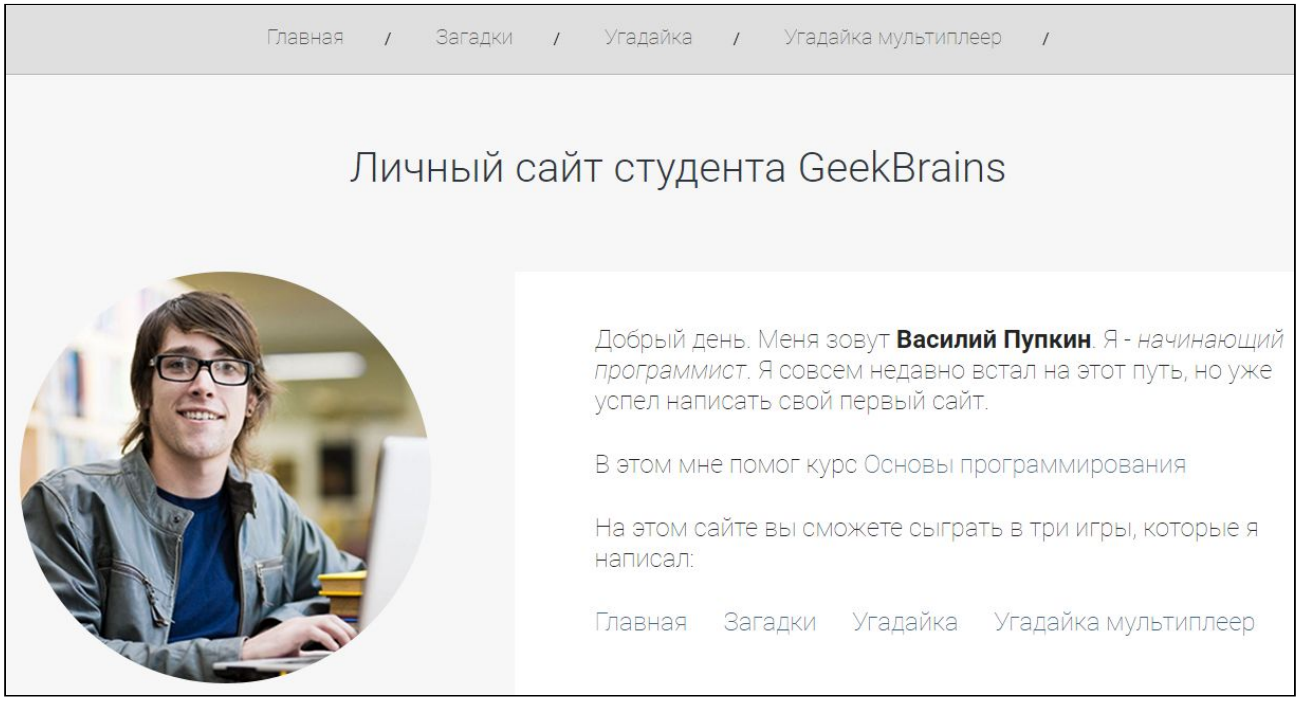
Здесь мы как бы говорим, “Эй! Все теги <a>, которые имеют id=”header”, будьте со своим стилем”

@import

Позволяет импортировать содержимое CSS-файла в текущую стилевую таблицу. В нашем примере с помощью этой команды мы подключим набор шрифтов.

Магия CSS

На этом теоретическое введение закончим и рассмотрим CSS файл, который нам любезно предоставил преподаватель курса "HTML&CSS". Скопируйте текст со страницы, и сохраните его в файл с именем style.css в папку site. После этого обновите в браузере страницу с html файлом, и она магическим образом преобразится

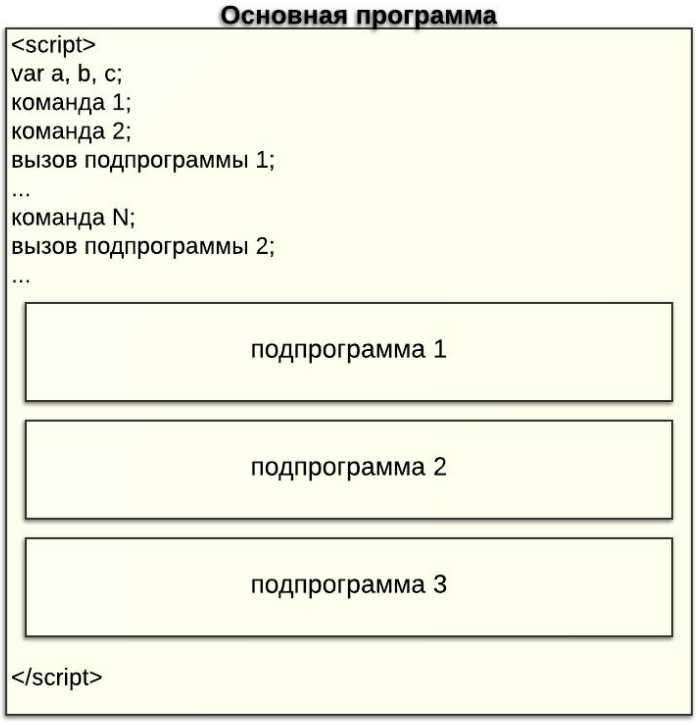


Функции

Функции — это рабочие лошадки JavaScript. Функции сами по себе играют те роли, которые в других языках исполняются самыми разными средствами: процедурами, методами, конструкторами, классами и модулями. После освоения функций вы овладеете существенной частью JavaScript.

Функция - это подпрограмма

Программисты при написании своих программ постоянно используют функции. Одной из главных целей использования функций является упрощение программирования. Действительно, ведь при написании большой программы часто возникают случаи, когда один и тот же код нужно выполнять много раз. В этом случае повторяющийся код может быть оформлен в виде подпрограммы внутри программы только со своим собственным именем. Такие программы внутри программы в общем случае называются подпрограммами, а в разных языках программирования эти подпрограммы могут носить разное название: функции, процедуры, методы - в зависимости от назначения подпрограммы.



На самом деле, когда мы писали alert или document.write и другие команды, мы уже использовали стандартные функции JavaScript.

Скажем привет функциям

Наберите программу, которая показывает, как описать функцию, и как её вызвать. В отличие от некоторых языков программирования, описание функции может идти после её вызова, но лучше старайтесь сначала декларировать функцию, а потом вызывать ее.

```
<script>
    function SayHello() {
        alert("Hello!");
    }
    SayHello();
</script>
```

SayHello() - это вызов функции. Отличительной особенностью функций от переменных является наличие скобок после её названия.

```
function SayHello() {
    alert("Hello!");
}
```

Это описание функции. Описание функции начинается с ключевого слова function, за которым идет название функции, скобки, в которых могут быть параметры функции, и фигурные скобки, в которых заключено тело функции.

Параметры функции

Мы написали простую функцию, которая выводит на экран приветствие. Но что если нам нужно, чтобы в приветствии так же было имя человека с которым мы хотим поздороваться? Писать функции с разными названиями? Для этих целей служат параметры функции.

```
<script>
    function SayHello(name)
    {
        document.write("Hello!"+name+"<br>");
    }
    SayHello("Phil");
    SayHello("Fred");
</script>
```

Если мы хотим, чтобы была возможность передать данные из основной программы в функцию, то можно при описании функции в скобках перечислить названия параметров функции. Теперь при вызове функции мы можем в скобках писать данные, которые будут переданы через параметры внутрь функции. Можно воспринимать параметры как специальные переменные, необходимые для передачи данных внутрь функции. Можно описать сколько угодно много параметров, перечисляя их через запятую.

Функции вычисляют и возвращают значения

Довольно часто необходимо, чтобы главная программа могла получить результат работы функции. В этом случаем можно использовать механизм возврата значения из функции. Тогда сама функция в конце своей работы возвращает какой-то результат в то место где она была вызвана, и мы можем использовать её возвращаемое значение.

Пример функции, возвращающей квадрат числа x:

```
<script>
    function degree2(x) {
        return x * x;
    }
```

```
    alert(degree2(5));
</script>
```

Упрощение логики программы

Как уже было сказано, функции позволяют сделать программу более читаемой. Давайте рассмотрим это на примере решения следующей задачи.

Написать программу: Определить значение z=max(a,2*b)*max(2*a-b,b), где max(x,y) – максимальное значение из чисел x,y.

Решение без функции	Решение с функцией

<script>
a = parseInt(prompt("a:"));
b = parseInt(prompt("b:"));
 	
max1 = a;
if (2 * b > max1)
max1 = 2 * b;
 	
max2 = 2 * a - b;
if (b > max2)
max2 = b;
 	
z = max1 * max2;
alert(z);
</script>
	<script>
	a = parseInt(prompt("a:"));
	b = parseInt(prompt("b:"));
	
	function Max(x, y) {
	if (x > y)
	return x;
	else
	return y;
	}
	
	z = Max(2 * b, a) * Max(2 * a - b, b);
	alert(z);
	</script>

Не знаю как вам, но нам кажется, что с функцией программа становится более простой для понимания.

Рекурсии

Рекурсия - это вызов функции самой себя.

Рассмотрим простой пример:

```
<script>
    function Print10(n) {
        document.write(n + "<br>");
        if (n < 10)
            Print10(n + 1);
    }
    Print10(1);
</script>
```

Здесь функция вызывает саму себя, пока n<10. Таким образом на экран выведется последовательность чисел от 1 до 10.

Рекурсия является альтернативой циклам, и с её помощью часто можно написать весьма элегантные алгоритмы решения задач.

Задание

Попробуйте переместить document.write после условия. Объясните получившийся результат.

Рекурсивный факториал

Факториал F(n) это произведение чисел от 1 до n. Например, F(5)=1*2*3*4*5=120.

Факториал хорошо решается с помощью рекурсии, так как факториал можно описать рекуррентно:

F(n)=F(n-1)*n, n>1
F(0)=1, n=0. Где n - целые числа

Если мы можем построить рекуррентное соотношение, то это рекуррентное соотношение легко реализуется с помощью рекурсии

```
<script>
    function factorial(n) {
        if (n == 0) {
            return 1;
        } else if(n < 0) {
            alert('Факториал рассчитывается только для натуральных чисел.');
        } else {
            return factorial(n - 1) * n;
        }
    }
    alert(factorial(5));
</script>
```

При этом вычисления происходят в следующем порядке:

1. F(5)=F(4)*5 2. F(4)=F(3)*4 3. F(3)=F(2)*3 4. F(2)=F(1)*2 5. F(1)=F(0)*1 6. F(0)=1	7. F(1)=1*1=1 8. F(2)=1*2=2 9. F(3)=2*3=6 10. F(4)=6*4=24 11. F(5)=24*5=120

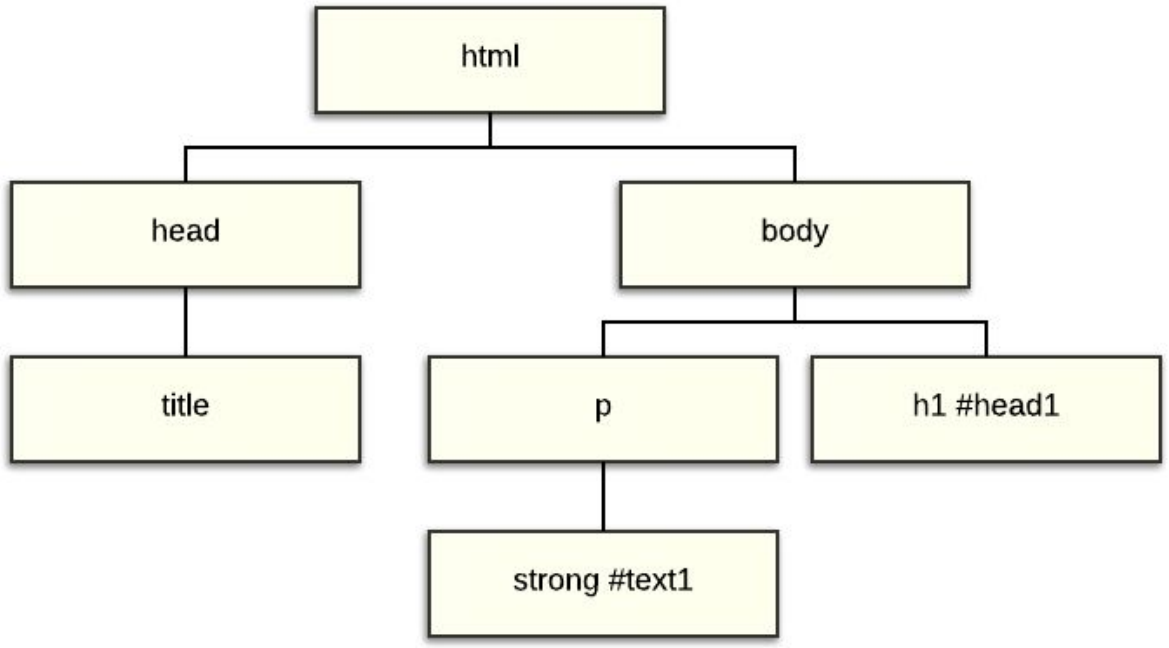
DOM, который построим мы

Рассмотрим небольшой html документ:

```
<!DOCTYPE HTML>
<html>
<head>
    <meta charset="utf-8">
    <title>Пример документа</title>
</head>
```

```
<body>
    <h1 id="head1">Заголовок</h1>
    <p>
        Параграф с <strong id="text1">очень важным </strong>текстом
    </p>
</body>
</html>
```

Когда браузер считывает веб-страницу, в памяти строится схема. Для нашей страницы схема будет выглядеть так. Если у элемента есть идентификатор, то он начинается с решетки.



Эта структура называется DOM (Document Object Model - объектная модель документа). Мы можем обращаться к этим элементам с помощью Javascript и считывать или изменять содержимое веб-страницы.

Например:

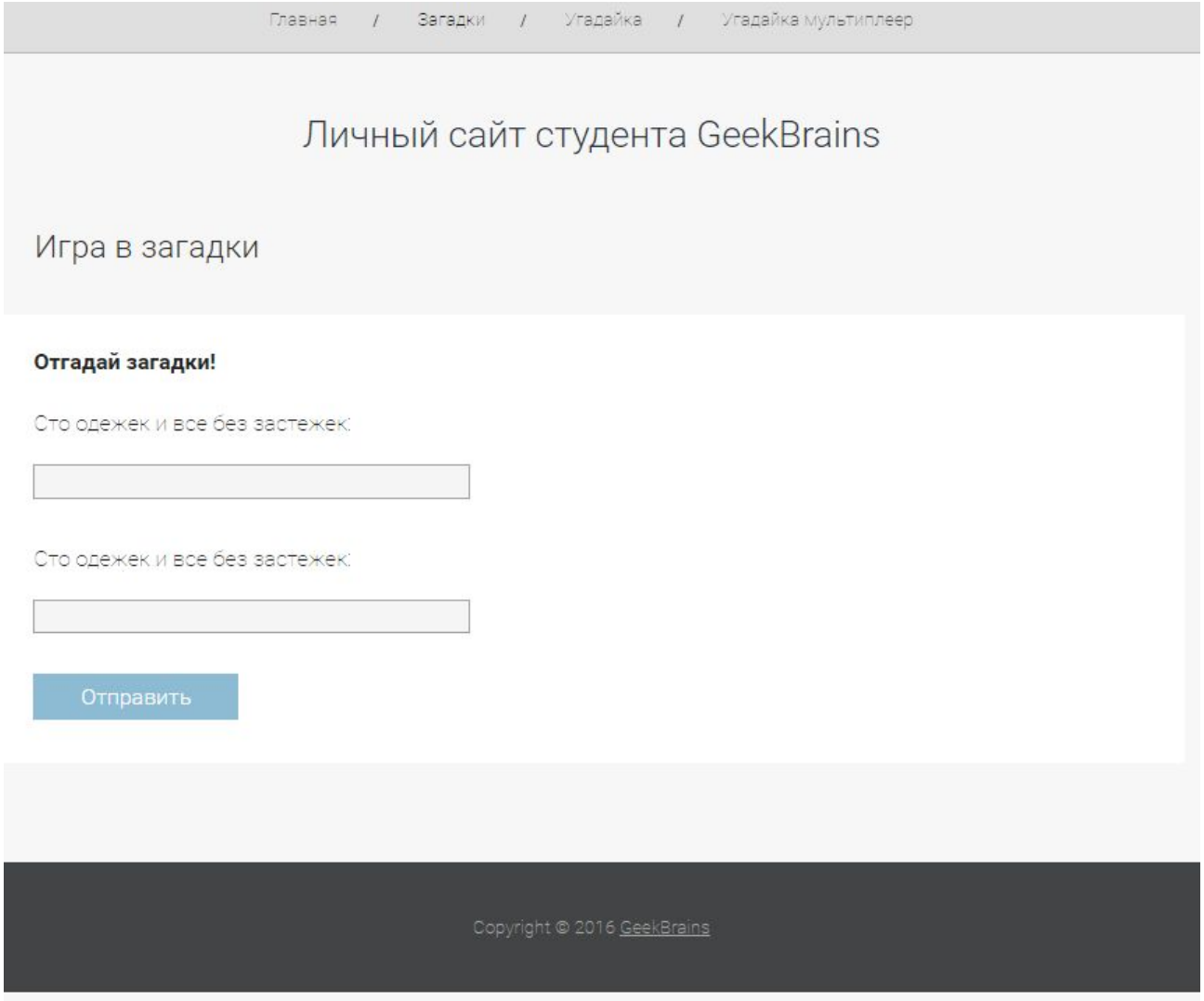
```
<html>
<head>
    <meta charset="utf-8">
    <title>Пример документа</title>
</head>
<body>
    <h1 id="head1">Заголовок до изменения</h1>
    <p>
        Параграф с <strong id="text1">очень важным </strong>текстом
    </p>
    <script>
        var head1 = document.getElementById("head1");  // Находим элемент в DOM
        head1.innerHTML = "Измененный заголовок";      // Обращаемся к свойству
```

```
элемента
    </script>
</body>
</html>
```

Здесь мы находим с помощью специальной функции getElementById элемент с идентификатором head1. У большинства элементов есть свойства (переменные, которые описывают состояние элемента), и мы используем свойство innerHTML, чтобы изменить внутренний текст элемента.

Страница с загадками

Давайте закрепим полученные знания. Для этого создадим интерактивную страничку с загадками.



input

Познакомимся с новым тегом <input>. Этот тег позволяет выводить на страницу различные элементы ввода информации.

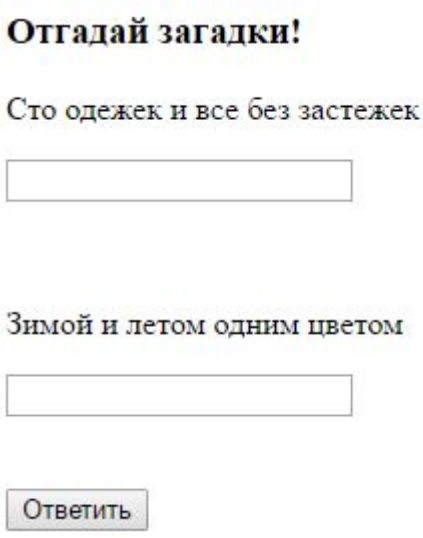
Например:

```
<html>
<head>
    <title>Пример документа</title>
    <meta charset="utf-8">
</head>
<body>
    <h3>Отгадай загадки!</h3>
    <p>
        Сто одежек и все без застежек
    </p>
    <input type="text" id="puzzle1">
    <br><br><br>

    <p>
        Зимой и летом одним цветом
    </p>
    <input type="text" id="puzzle2">
    <br><br><br>

    <button>Ответить</button>
</body>
</html>
```

Программа выведет на экран текст загадки, поля, куда пользователь может вводить ответы и кнопку с надписью “Ответить”.



Мы сделали возможность вводить ответы, теперь давайте сделаем возможность их проверять. Для этого нам потребуется познакомиться с событиями.

События или events

В Javascript функции играют ещё большую роль, чем в других языках. Дело в том, что они позволяют реализовать механизм событий.

События - это действия, происходящие в браузере: щелчки мышью, нажатия клавиш, перемещения указателя мыши, загрузка рисунка и др. Самые различные события могут влиять на дальнейшее поведение браузера.

Программы, которые позволяют определять события и выполнять соответствующие им действия, называются **обработчиками событий**.

Заставим кнопку реагировать на нажатие.

```
<button onclick="alert('Событие!');">Ответить</button>
```

Обратите внимание, что нужно использовать разные кавычки в тексте:

```
onclick="alert('Событие!');"
```

Теперь при нажатии на кнопку (возникновения события click) будет выполняться команда alert.

Но чаще при возникновении событий нужно выполнить несколько команд. В этом случае лучше использовать функции:

```
<button onclick="click1();">Ответить</button>

<script>
    function click1()
    {
        alert("События удобнее обрабатывать");
        alert("с помощью функций!");
    }
</script>
```

Скрипт с функцией обработки события может быть описан как до, так и после элемента, в котором назначается обработчик события.

Итоговая программа

Функция checkAnswer

Функция checkAnswer позволит упростить проверку значений. В функцию мы передаем id элемента и правильный ответ. Функция находит элемент по id и запоминает значение элемента в переменной userAnswer. Далее мы сравниваем userAnswer с правильным ответом. Если значения переменных совпадают, то функция возвращает истину, иначе функция возвращает ложь.

```
function checkAnswer(id, trueAnswer) {
    var userAnswer = document.getElementById(id).value;
    return userAnswer == trueAnswer;
}
```

Функция checkAnswers

В этой функции мы множество раз обращаемся к функции checkAnswer и передаем ей id элемента и правильный ответ. Каждый раз, когда checkAnswer возвращает истину, то переменная correctAnswers увеличивается на единицу. В конце мы выводим значение correctAnswers.

```
function checkAnswers()
{
    var correctAnswers = 0;

    if (checkAnswer('puzzle1', 'капуста'))
```

```
        correctAnswers++;

    if (checkAnswer('puzzle2', 'елка'))
        correctAnswers++;

    alert('Количество правильных ответов: ' + correctAnswers);
}
```

Текст всей программы веб-страницы:

```
<html>
<head>
    <title>Пример документа</title>
    <meta charset="utf-8">
    <script>
        function checkAnswer(id, trueAnswer) {
            var userAnswer = document.getElementById(id).value;
            return userAnswer == trueAnswer;
        }

        function checkAnswers()
        {
            var correctAnswers = 0;

            if (checkAnswer('puzzle1', 'капуста'))
                correctAnswers++;

            if (checkAnswer('puzzle2', 'елка'))
                correctAnswers++;

            alert('Количество правильных ответов: ' + correctAnswers);
        }
    </script>
</head>
<body>
<h3>Отгадай загадки!</h3>
<p>
    Сто одежек и все без застежек
</p>
<input type="text" id="puzzle1">
<br><br><br>

<p>
    Зимой и летом одним цветом
</p>
<input type="text" id="puzzle2">
<br><br><br>

<button onclick="checkAnswers();">Ответить</button>

</body>
</html>
```

Домашнее задание

1. Разместить на сайте свою фотографию, и написать о себе
2. Разместить на сайте свои загадки.
3. Разместить на сайте игру “Угадайка” для двух игроков.
4. * Разместить на сайте Генератор случайных паролей. Пользователь указывает в текстовом поле длину желаемого пароля, и получает случайный пароль, состоящий из латинских букв, а также цифр.